Информационные технологии и языки программирования

Лашин Сергей Александрович
Кафедра Информационной Биологии ФЕН НГУ
Электронно-лекционный курс
Разработан в рамках реализации Программы развития НИУ-НГУ, 2012 год

Лекция 1

Предмет программирования

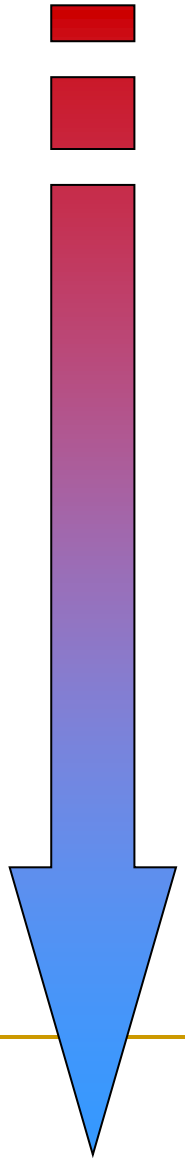
- Программирование – процесс создания компьютерных программ (программного обеспечения)
- Программа – последовательность инструкций, предназначенных для выполнения компьютером

Основные этапы разработки программ

- Анализ
- Проектирование
- Реализация
- Тестирование
- Внедрение
- Сопровождение

История развития программирования

Машинные коды	1940-1946
Ассемблер	1949-1957
Процедурное программирование	1954-1957
Структурное программирование	1960
Объектно-ориентированное программирование	1986-…

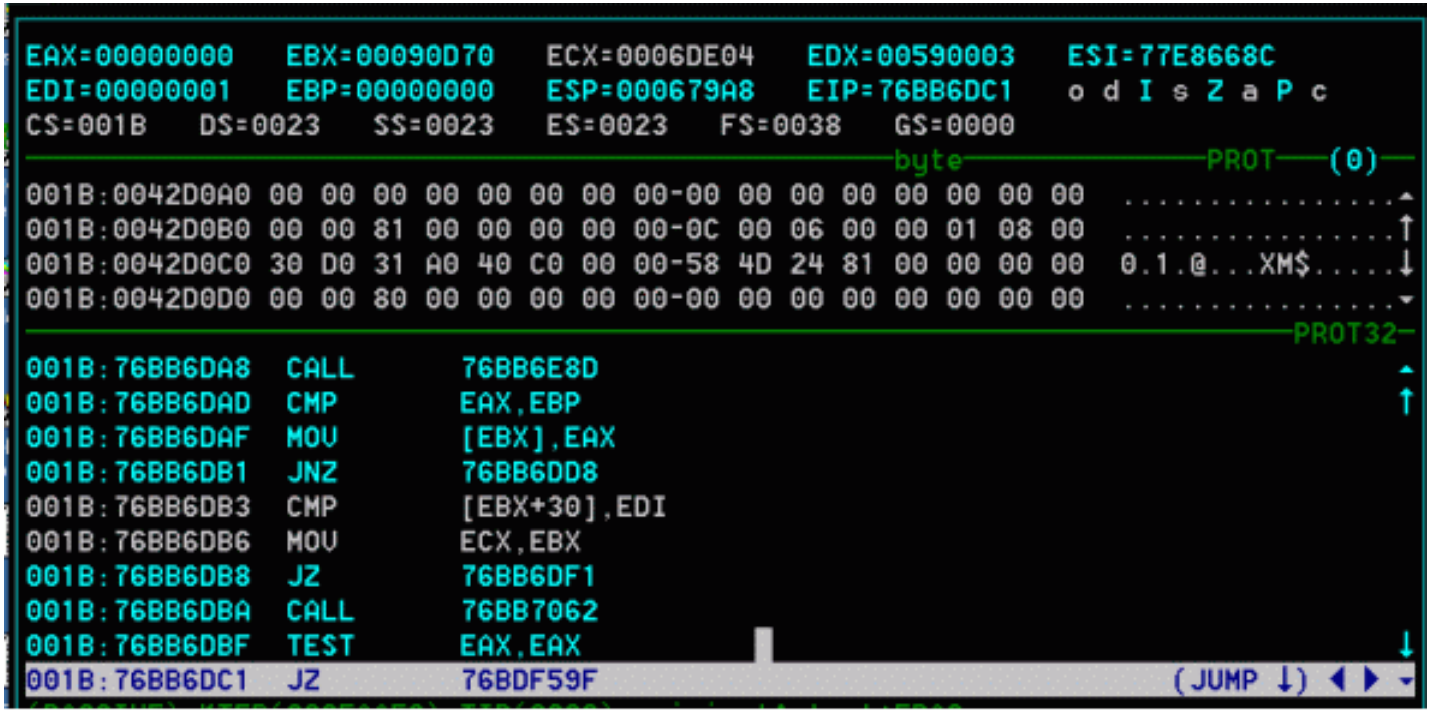


Этапы развития программирования: Машинные коды

```
Break due to BPX KERNEL32!CreateFileA DO "? PID; D esp->4 L 20; P RET; ? EAX; x;«
000001DC 0000000476 " -" ; PID
0010:0012138C 64 65 6D 6F 2E 70 72 6F-74 65 63 74 65 64 2E 63 demo.protected.c
0010:0012139C 72 6B 2E 65 78 65 00 00-00 00 00 00 00 00 00 00 rk.exe.........
00000074 0000000116 "t" ; код возврата
```

- Трудность разработки и сопровождения
- Сильно ограниченный объём понимания кода
- Непереносимость кода на другую платформу

Этапы развития программирования: Ассемблер



Компиляция – перевод текста программы с языка (высокого) уровня в машинные коды

- Трудность разработки и сопровождения
- Непереносимость кода на другую платформу

Этапы развития программирования: Процедурные языки - Фортран

```
INTEGER A,B,C
READ(5,501) A,B,C
501 FORMAT(3I5)
IF(A.EQ.0 .OR. B.EQ.0 .OR. C.EQ.0) STOP 1
S = (A + B + C) / 2.0
AREA = SQRT( S * (S - A) * (S - B) * (S - C))
WRITE(6,601) A,B,C,AREA
601 FORMAT(4H A= ,I5,5H B= ,I5,5H C= ,I5,8H AREA= ,F10.2,12HSQUARE UNITS)
STOP
END
```

- Хорошая читаемость кода
- Переносимость кода на другую платформу
- Сложности программирования сложных структур данных

Этапы развития программирования: Структурные языки – Паскаль, Си

```
#include <stdio.h>

struct student
{
    char *name;
    float marks;
}   student1, student2;

int main ( )
{
    struct student student3;

    student1.name = "Tom";
    student2.marks = 99.9;
    printf (" Name is %s \n", student1.name);
    printf (" Marks are %f \n", student2.marks);
}
```

- Хорошая читаемость кода
- Переносимость кода на другую платформу
- Возможность программирования сложных структур данных
- Сложность создания больших и распределённых программных систем

```
public class ClassShapeMain {
  /** Iterate over all the Shapes, getting their areas */
  public double totalAreas() {
    Iterator it = allShapes.iterator();
    double total = 0.0;
    while (it.hasNext()) {
      Shape s = (Shape)it.next();
      total += s.computeArea();
    }
    return total;
  }
}
class Circle extends Shape {
  double radius;
  public double computeArea() {
    return Math.PI * radius * radius;
  }
}
class Rectangle extends Shape {
  double width, height;
  public double computeArea() {
    return width * height;
  }
}
abstract class Shape {
  protected int x, y;
  public abstract double computeArea();
}
```

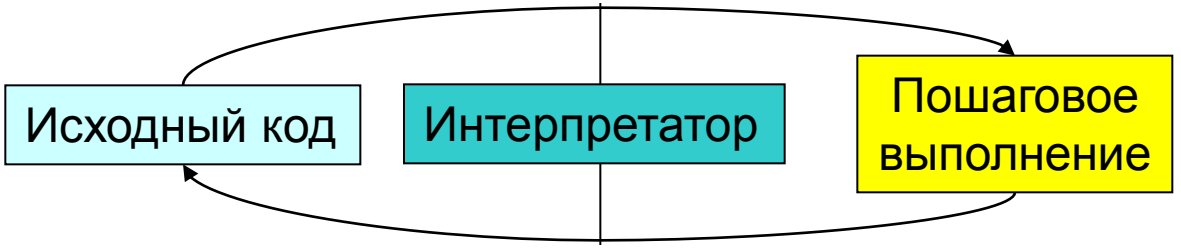
Этапы развития программирования: Объектно-ориентированные языки – C++, Java

- Хорошая читаемость кода
- Переносимость кода на другую платформу
- Возможность программирования сложных структур данных
- Разделение функциональности и «зон ответственности» - разделяй и властвуй
- Повторное использование кода
- Упрощение создания больших и распределённых программных систем

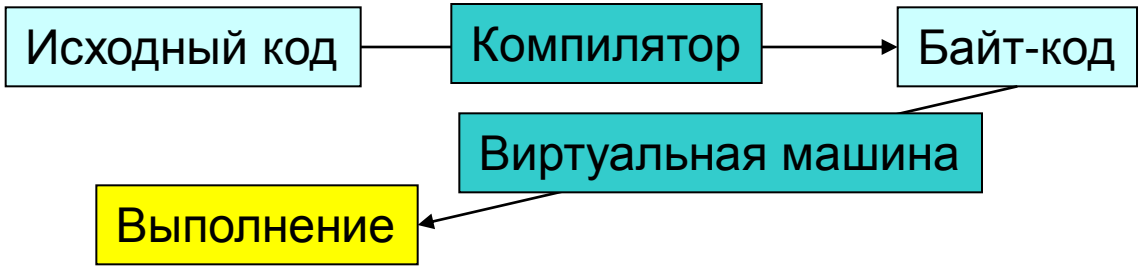
Компилируемые и интерпретируемые языки

Компилируемые языки

Интерпретируемые языки



Интерпретация компилирующего типа



Язык Java

- Java – язык и платформа (вирт.машина)
- Выдающаяся кроссплатформенность
- Безопасность
- Реализация концепций ООП
- Совместимость со «старыми» языками (C/C++ и Фортран)
- Встроенные в язык средства для работы со строками, сетью, многопоточностью (унификация)
- Автоматическое управление памятью (сборка мусора)

Задачи курса

- Изучение базового синтаксиса Java
- Изучение основных типов данных Java
- Изучение основных принципов ООП
- Работа со строками
- Работа с файлами
- Работа с биоинформатическими библиотеками
- Знакомство с форматами представления биологических данных

Простейшая программа на Java

```
public class HelloWorldApplication {


    /**
     * @param args - входные параметры (аргументы командной строки
    */


    public static void main(String[] args) { // точка входа
        System.out.println("Hello, world!");// вывод на печать
    }
}
```

Установка Java

- Скачать JDK (1.7) с сайта www.java.com
  - http://www.java.com/ru/download/
- Установить JDK на компьютер
- Добавить каталог Java/bin в переменную пути PATH (для работы команд java, javac)

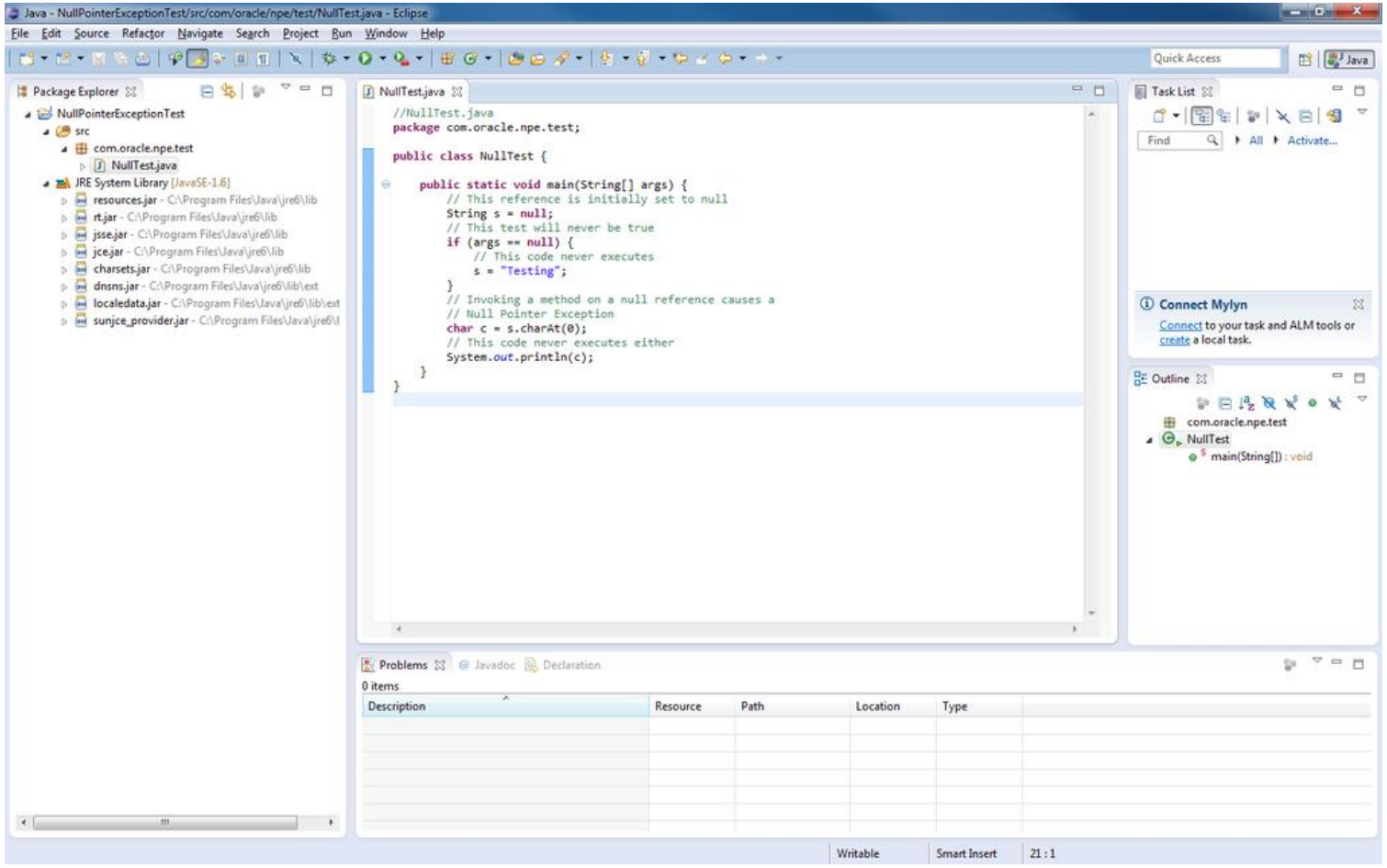
Компиляция и запуск программы

- Создать файл с текстом программы
  - имя файла должно совпадать с именем класса
  - расширение файла должно быть .java (например, HelloWorldApplication.java)
- Скомпилировать программу с помощью программы javac (например, javac HelloWorldApplication.java
- Запустить получившийся .class файл командой java (например, java HelloWorldApplication ВНИМАНИЕ: расширение .class НЕ указывается)

Использование интегрированных сред разработки

- Eclipse (http://eclipse.org/)
  - Eclipse IDE for Java Developers
- NetBeans (www.netbeans.org)
- IntelliJ IDEA (www.jetbrains.com/idea)
- JDeveloper (http://www.oracle.com/technology/products/jdev/index.html)
- Многие другие…

Использование интегрированных сред разработки (Eclipse)



Лекция 2
Переменные, типы данных и управляющие конструкции в языке Java

Переменные

- В науке – атрибут физической или абстрактной системы, который может изменять своё значение.
  - Примеры: рост ребёнка, температура в воздуха, или параметр функции.
- В программировании – именованная область памяти, имя или адрес которой можно использовать для осуществления доступа к **данным**, находящимся в переменной. *Т.е. переменные – это «места для хранения данных».*

Типы данных в Java

Название	Хранимые начения	Диапазон	Пример
`byte`	Числа целые	$-128..127$	100
`char`	Числа и/или символы	$0..2^{16}-1$, или 0..65535	39 123
`short`	Числа целые	$-2^{15}..2^{15}-1$, или −32768..32767	‘C’
`int`	Числа целые	$-2^{31}..2^{31}-1$, или −2147483648..2147483647	-1000000
`long`	Числа целые	$-2^{63}..2^{63}-1$, или примерно $-9.2\cdot10^{18}..9.2\cdot10^{18}$	$3\cdot10^{17}$
`float`	Числа с плавающей точкой	$-(2-2^{-23})\cdot2^{127}..(2-2-23)\cdot2127$, или примерно $-3.4\cdot10^{38}..3.4\cdot10^{38}$, а также -∞, ∞, NaN	0.172
`double`	Числа с плавающей точкой	$-(2-2^{-52})\cdot2^{1023}..(2-2^{-52})\cdot2^{1023}$, или примерно $-1.8\cdot10^{308}..1.8\cdot10^{308}$, а также -∞, ∞, NaN	$2.5\cdot10^{300}$
`boolean`	Логические значения	`true, false`	true

Особенности целочисленной арифметики

byte: -128…127

```
byte a = 127, b=1;
byte c = a + b;
//c == -128
```

char: 0 1 … 127 -128 -127 … -1

```
int a = 1 / 2;          //a == 0
int b = 9 / 10;         //b == 0
```

Целые константы

```
125          //целое десятичное

0126         //целое восьмеричное

0x2FE        //целое шестнадцатеричное

12U          //целое беззнаковое

12L          //целое типа long int
```

Вещественные константы

1.86 //по умолчанию все вещественные
//константы типа double

.25 == 0.25

12. != 12 //12. – вещественное, а 12 – целое!

3.8e-6 == 3.8E-6 //== $3.8*10^{-6}$

68.13F //константа типа float

Символьные и строковые константы

```
'a'             //символьная константа (char)
                //тождественна коду символа

"name"          //строковая константа

"name\n"        //строковая константа со
                //специальным символом \n
"long"
"Name"          // == "longName"
```

Специальные символы

\n	//новая строка
\r	//возврат каретки
\t	//табуляция
\\	//back-slash
\’	//одинарная кавычка
\”	//двойная кавычка

Имена и идентификаторы

Алфавит имен: A-Z, a-z, _, 0-9

Первый символ имени: A-Z, a-z, _

Правильные имена: loopCounter, i, _98a, x15

Неправильные имена: loop$Cnt, 9x

Условные обозначения

expr – любое допустимое выражение, составленное из переменных, констант и операторов

type – имя типа

lvalue – (left value) изначально – выражение, которое может стоять в левой части оператора присваивания.
Более точно – объект, занимающий память.

Арифметические операторы

```
expr + expr          //сложение
expr - expr          //вычитание
expr * expr          //умножение
expr / expr          //деление
expr % expr          //деление по модулю (целые)
- expr               //унарный минус
+ expr               //унарный плюс
```

Операторы присваивания

```
lvalue = expr          //присваивание

lvalue += expr         //lvalue = lvalue + expr

-=, *=, /=, %=, |=, &=, ^=, >>=, <<=

++lvalue               //lvalue += 1 префиксный
                       //инкремент
lvalue++               //постфиксный инкремент

--lvalue               //префиксный декремент

lvalue--               //постфиксный декремент
```

Особенности операторов присваивания

```
int a = 0;
int b = 0;
a = b++;
//a==0
//b==1
```

```
int a = 0;
int b = 0;
a = ++b;
//a==1
//b==1
```

```
int a = 0;
int b = 0;

//плохой стиль
a = b++ + b++;
//a==0
//b==2
```

```
int a = 0;
int b = 0;
a = b = 1;
//a==1
//b==1
```

```
int a = 0;
int b = 0;
(a = b) = 1;
//a==1
//b==0
```

Логические операторы

```
expr == expr        //сравнение на равенство
expr != expr        //сравнение на неравенство
expr > expr         //больше
expr < expr         //меньше
expr >= expr        //больше или равно
expr <= expr        //меньше или равно
```

Логические операторы

```
expr && expr        //логическое AND

expr || expr        //логическое OR

! expr              //логическое отрицание
```

Пример

```
int a;// Объявление переменной
int b = 10; // Объявление и инициализация
c = 5; // Ошибка: не указан тип переменной
a = 1; // Значение a равно 1


int d = a + b; // ...
a++; //...
b--; //...


d += b; //...
```

Инструкции

Инструкция это:

1. Выражение, построенное из имен,
   констант и операторов в соответствии с синтаксисом.
   ```
   a = b + c*10;
   30;
   ```

2. Объявление переменных, типов, функций и т.д..
   ```
   int a = 1;
   double b;
   ```

3. Набор инструкций, объединенных в составную.

4. Одна из специальных управляющих инструкций.

Составная инструкция, область жизни переменных

```
public static void main (String[] args)
{
        int a = 1;
        int c = 4;
        //a == 1, c == 4
        {               //открытие составной инструкции
                int a = 2;
                //a == 2 - другой a!
                //c == 4 - тот же c!
                int b = 3;
                //b == 3
        }               //закрытие составной инструкции
        //a == 1, c == 4
        //b не определен!
        return 0;
}
```

Условный выбор

```
if (expr) инструкция1 else инструкция2
```

Если expr == true, то выполняется инструкция1, иначе – инструкция2

```
if (a > 0){
      a--;
      System.out.println (“success”);
}
else{
      a = 0;
      System.out.println(“failure\n”);
}
```

Цикл for

```
for (инструкция1; expr; инструкция2) инструкция
```

1. Выполняется инструкция1
2. Если expr == false – выход из цикла
3. Выполняется инструкция
4. Выполняется инструкция2
5. Переход к пункту 2

```
int a, i;
for (a = 1, i = 2; i <= 10; ++i)
      a *=i;
//i == 11!
```

```
//бесконечный цикл
for (;;);
```

Управление циклами

break – безусловный выход из цикла

continue – пропускает текущую итерацию цикла

```
int a = 1;
int i = 1;
for(;;)
{
      if (++i > 10) break;
      a += i;
}
//i == 11!
```

```
int a = 1;
int i;
for (i=1; i<=20; ++i)
{
      if (i > 10) continue;
      a += i;
}
//i == 21!
```

Ветвление

1. Вычисляется значение expr.
2. Если значение expr совпадает с одним из значений в метках case, переход к метке.
3. В противном случае переход к метке default.
4. Инструкции выполняются до конца ветвления, либо до break

```
switch (expr)
{
case знач1:
       набор_инструкций1 //break;
case знач2:
       набор_инструкций2 //break;
…
default:
       набор_инструкцийD
};
```

Ветвление: пример

```
int mark;
...
switch (mark)
{
case 5:
      System.out.prinln (“Да вы отличник”);
      break;
case 4:
      System.out.prinln (“Да вы хорошист”);
      break;
case 3:
      System.out.prinln (“Да вы троечник”);
      break;
default:
      System.out.prinln (“Похоже вы двоечник :(”);
}
```

Лекция 3
Основы объектно-ориентированного программирования.
Классы и объекты.

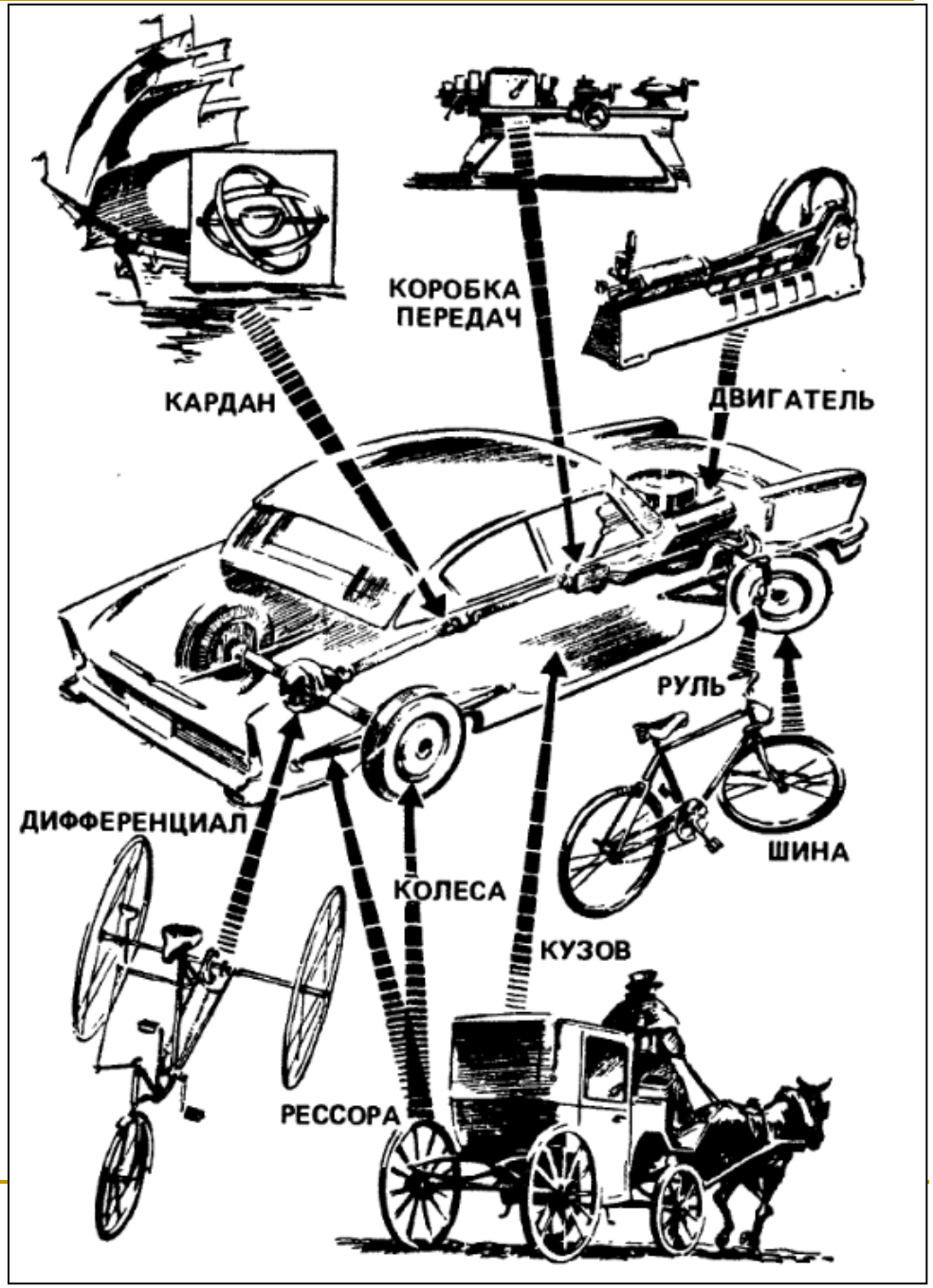
Две парадигмы программирования

- Модель, ориентированная на процесс (process oriented model)
  - Программу определяет последовательность операторов её кода. Код воздействует на данные (code acting on data).
- Модель, ориентированная на данные (object oriented model)
  - Управляемый данными доступ к коду (data controlling access to code)/

Идеи ООП. Абстракция

- Рассмотрение сложного объекта в качестве единого целого, обладающего собственным уникальным состоянием и поведением
- Отсутствие необходимости вникать в сложные детали и составные части

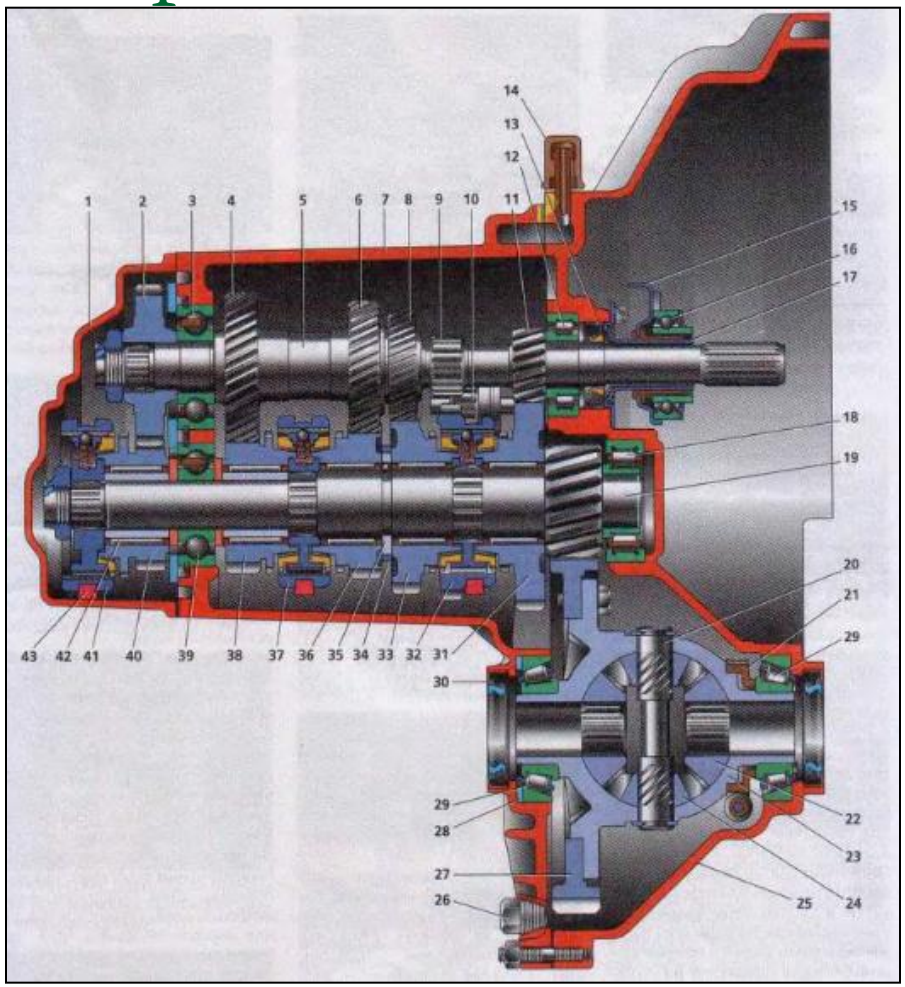
Абстракция. Пример. Автомобиль.



Три принципа ООП

- Инкапсуляция – объединение кода и данных в объект, скрывающее реализацию объекта и непосредственный доступ данным (ОСНОВА ИНКАПСУЛЯЦИИ – КЛАСС)
- Наследование – описание нового класса объектов на основе уже существующего (родительского); при этом свойства и функциональность родительского класса наследуются классом-потомком (и могут добавиться новые)
- Полиморфизм (греч. «много форм») – свойство, позволяющее применять один интерфейс для общего класса действий (одни и те же операции к объектам разных классов, связанных единой спецификацией)

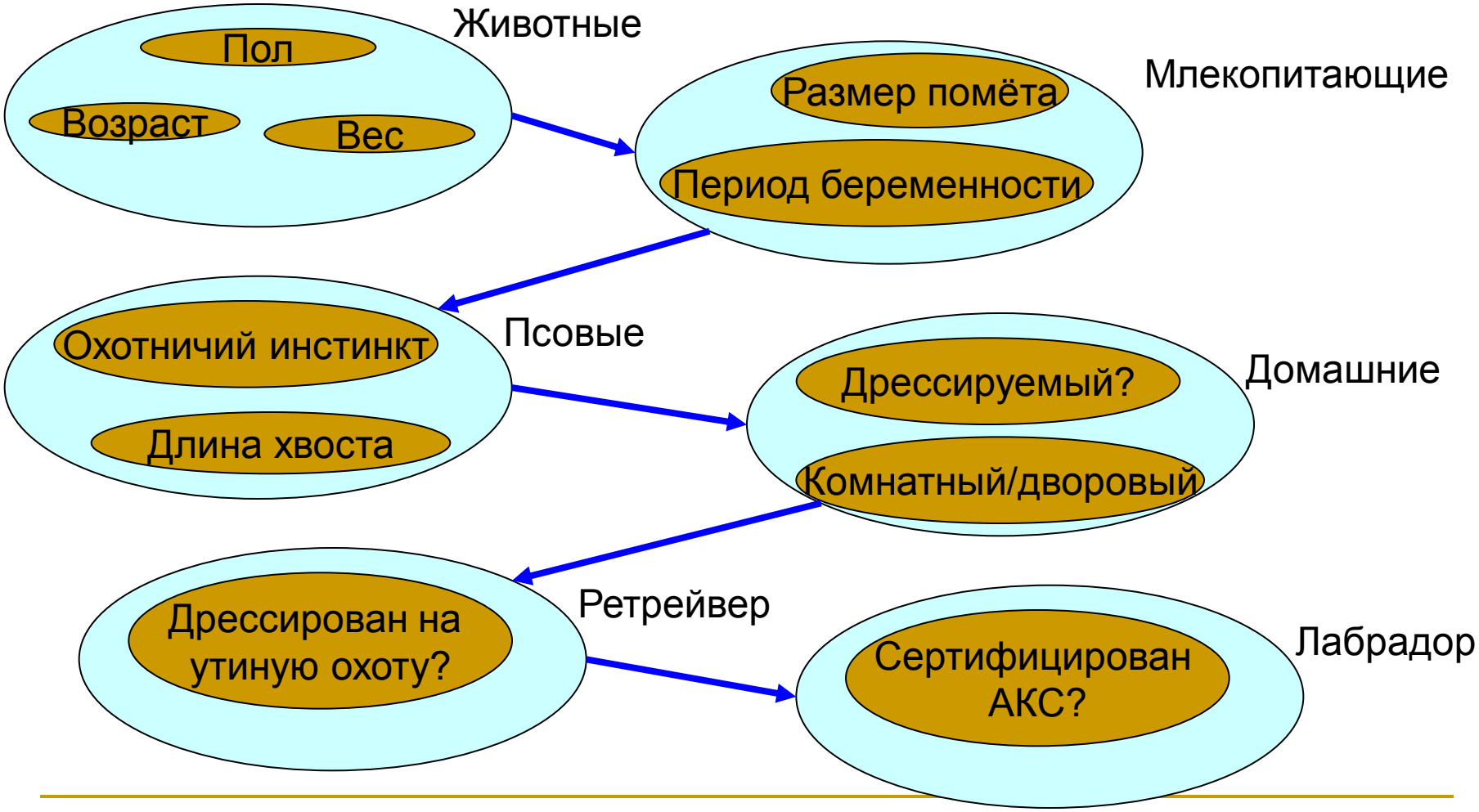
Инкапсуляция. Пример. Коробка передач автомобиля



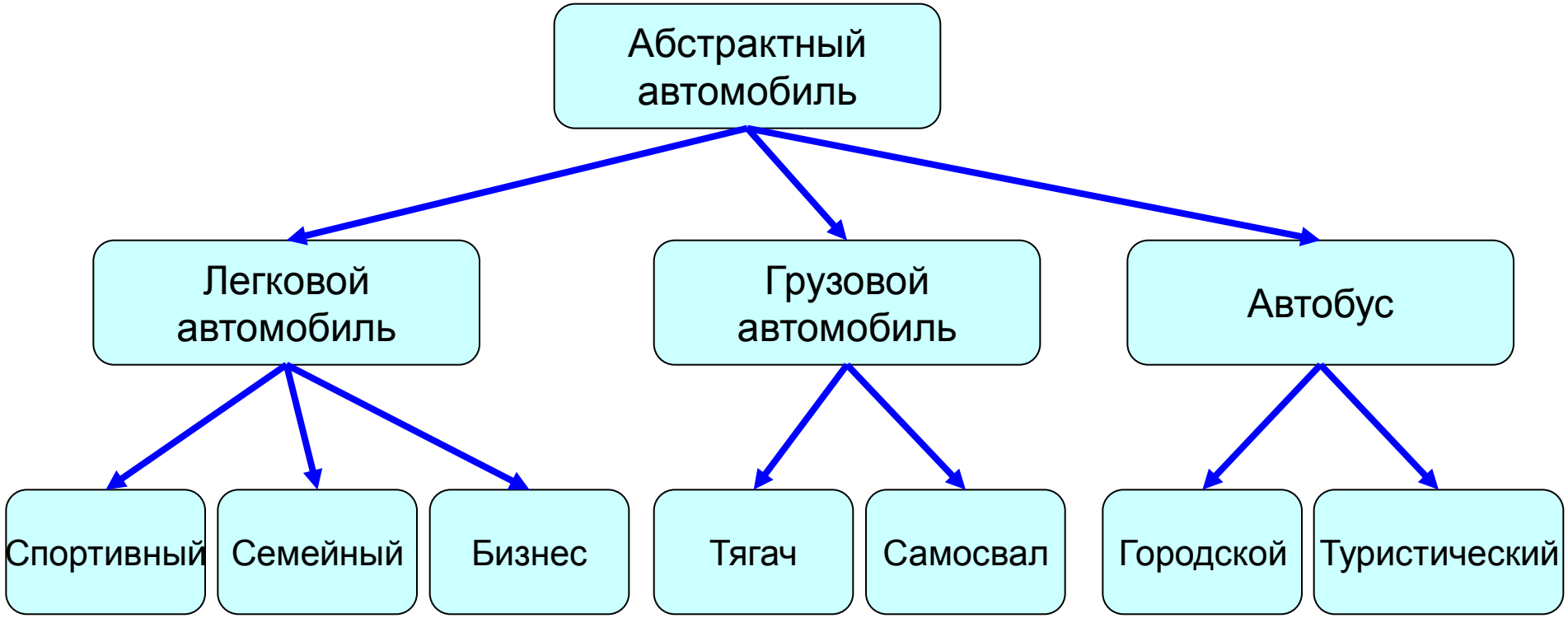
Пользователь ограждён от сложного устройства коробки. Он использует только рычаг переключения.

Карданный вал ограждён от сложного устройства коробки. Он получает только крутящий момент.

Наследование. Пример. Классификация животных



Полиморфизм



Классы и объекты в Java

- Класс – прототип, описывающий устройство объектов – его экземпляров
- Класс определяет тип данных (кроме специальных случаев)
- Класс задаёт функциональность – поведение всех его представителей

- **Объект** – конкретный «экземпляр» класса

Объект

- Состояние (state)
- Поведение (behavior)
- Уникальность (identity)

Поведение

- Способность реагировать на внешние события
- Способность воздействовать на другие объекты

Состояние

- Влияет на поведение объекта
- Может меняться в зависимости от внешних воздействий
- Объект имеет набор (пространство) разрешенных состояний

Уникальность

- Каждый объект можно отличить от других при любых условиях
- Объекты дискретны (измеряются в «штуках»)

Сообщения

Сообщения – способ взаимодействия объектов.

Поведение объекта определяет набор сообщений, на которые он может реагировать, а также которые он может посылать другим объектам.

Комплексные числа. Пример класса

```
public class ComplexNumber {
    private double real;  // Вещественная часть
    private double image; // Мнимая часть
    public ComplexNumber(){    // Конструктор по умолчанию
        this.real = 0.0;
        this.image = 0.0;
    }
    public ComplexNumber(double re, double im){ // Конструктор
        this.real = re;
        this.real = im;
    }

    public double getModule(){// Модуль комплексного числа
        return Math.sqrt(real*real+image*image);
    }

    public double getArg(){// Аргумент комплексного числа
        return Math.acos(real/this.getModule());
    }
}
```

Комплексные числа. Пример объектов

```
ComplexNumber a = new ComplexNumber(); // (0.0 + 0.0*i)
ComplexNumber b = new ComplexNumber(1.0,2.0); // (1.0 + 2.0*i)

double d = a.getModule(); // Модуль числа a

double f = b.getArg(); // Аргумент числа b

a = new ComplexNumber(0.5,2.4); // Теперь a = (0.5 + 2.4*i)
```

Атрибуты и методы классов

- Атрибуты – данные (поля) класса; отображают состояние объекта

```
private double real;  // Вещественная часть
private double image; // Мнимая часть
```

- Методы – функции, являющиеся полями класса; отображают поведение объектов класса (сообщения, которые могут воспринимать); методы применяются к объектам (экземплярам класса)

```
public double getModule(){
    return Math.sqrt(real*real+image*image);
}
```

Модификаторы доступа

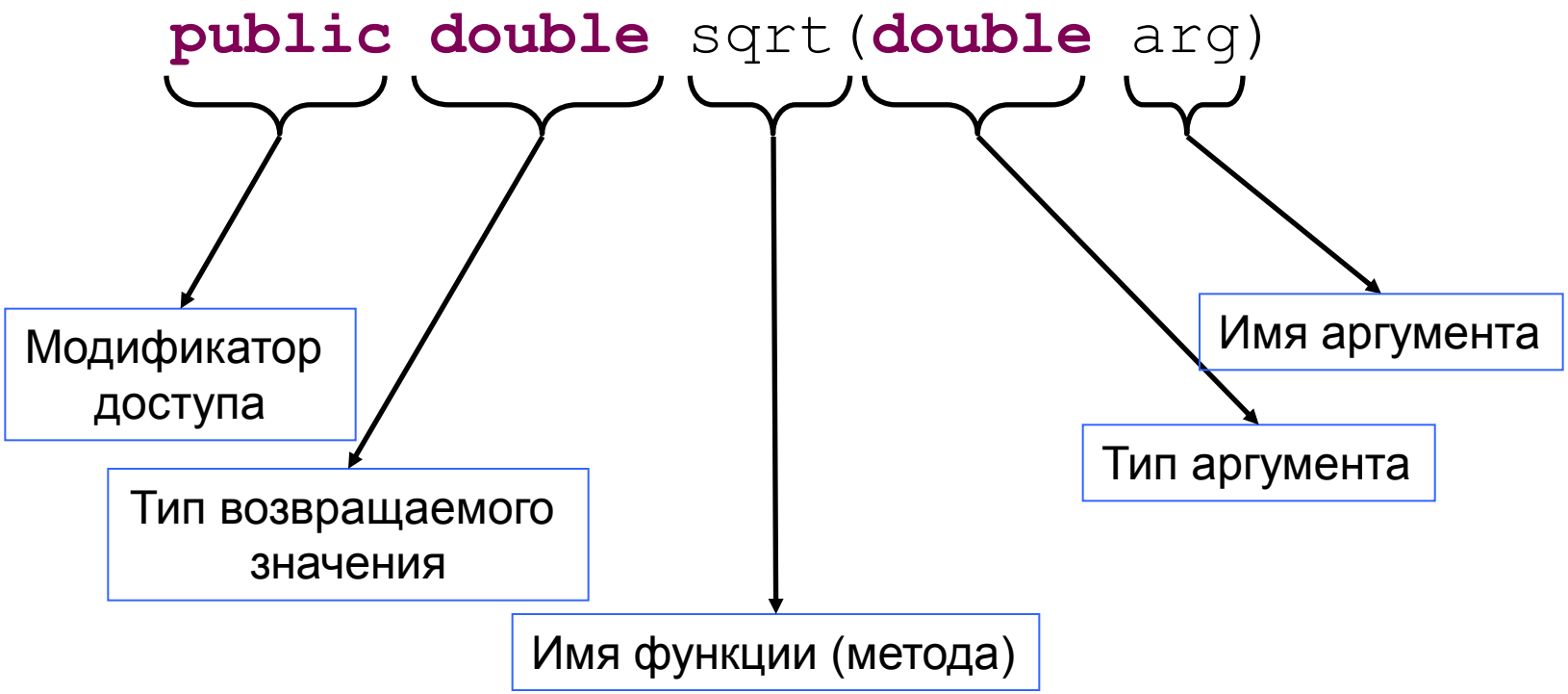
public: поле доступно всем.

private: поле доступно методам класса.

protected: поле доступно методам класса и подклассов.

(без модификатора) по умолчанию: поле доступно в пределах пакета (package)

Функции. Сигнатура



void – «пустой» тип возвращаемого значения (ничего не возвращает)

Методы. Перегрузка

- Перегрузка методов (methods overloading) – объявление нескольких методов с одинаковым именем, но с разной сигнатурой (в частности, с разным числом и типами аргументов)

```
public Complex[] getRoot(); // возвращает квадратный корень
public Complex[] getRoot(int n); // возвращает корень n-й степени (целой)
public Complex[] getRoot(double deg); // возвращает корень дробной степени
```

Конструкторы

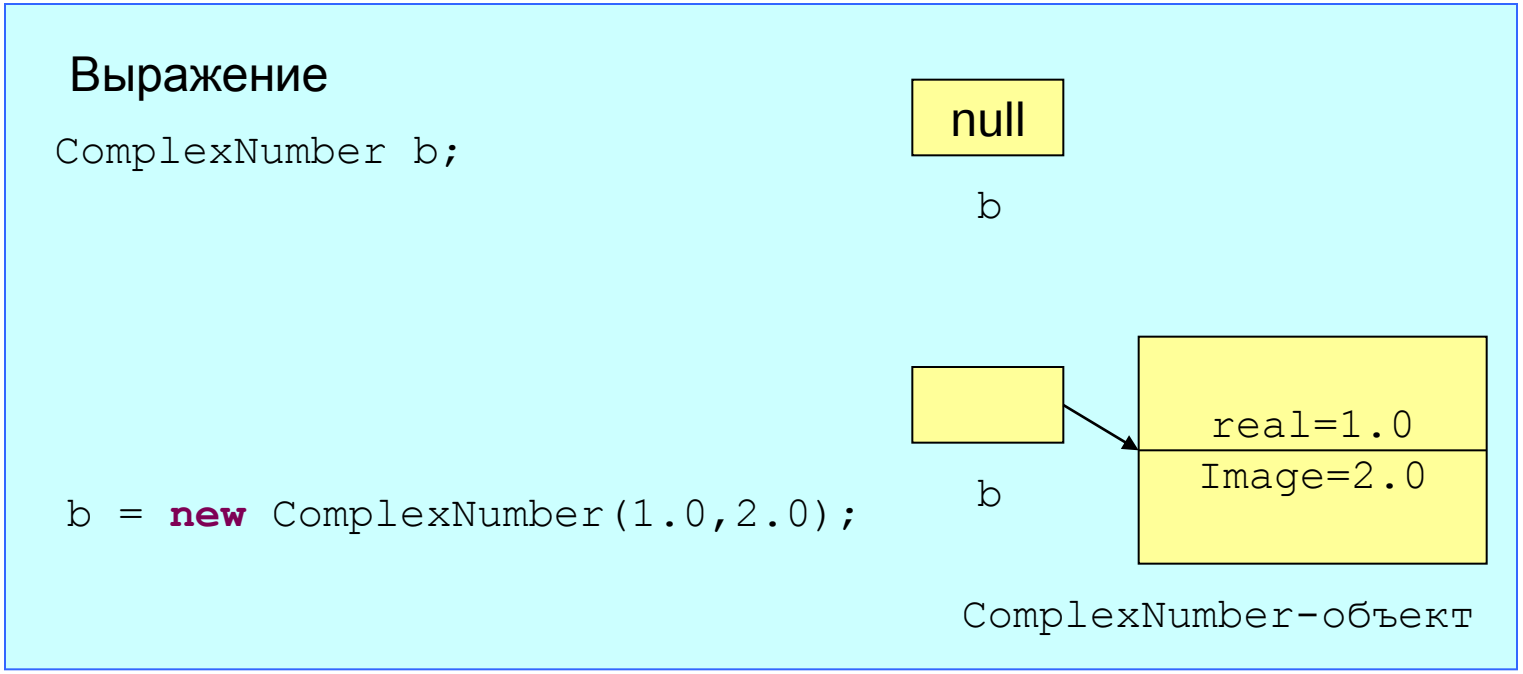
Конструктор - специальный метод для конструирования объекта класса; конструкторов может быть несколько

```
public ComplexNumber(){    // Конструктор по умолчанию
        this.real = 0.0;
        this.image = 0.0;
    }
    public ComplexNumber(double re, double im){ // Конструктор
        this.real = re;
        this.real = im;
    }
```

Создание объектов

new – операция создания объекта (ключевое слово Java)

null – указатель на нулевой адрес («пустое место»)



Статические методы

- Статический метод – метод, который можно вызвать не создавая объекта класса
  - Пример: Math.sqrt(…) – при вызове не создаётся объект класса Math
  - Особый статический метод `void main` – служит для запуска программы (точка входа)

Классы как программные модули

- В языке Java класс представляет единицу трансляции и программный модуль («нет жизни вне класса»)
- Классы могут составлять пакеты (package) - библиотечные модули, объединённые общей функциональностью

```
package ru.nsc.bionet.mathematics;
```

Объявление пакета

```
import ru.nsc.bionet.mathematics;
```

Импорт пакета

Область действия и время жизни переменных

- Переменная хранится
  - Поле класса, статическая переменная (static) – в течение работы программы
  - Поле объекта – в течение жизни объекта
  - Объявленная внутри метода – в течение вызова метода
  - Объявленная внутри инструкции – в течение выполнения инструкции

Лекция 4
Строки, Массивы

Массивы в Java

- Массив – набор однотипных элементов, на которые ссылаются по общему имени и доступ к которым реализуется с помощью индексации

```
// Одномерный массив
int array[]; // Объявление
array = new int[10]; // Инициализация
array[0] = 1; // Индексация (доступ)
```

```
// Двумерный массив
int array2[][]; // Объявление
array2 = new int[10][10]; // Инициализация
array2[0][0] = 1; // Индексация (доступ)
```

Массивы в Java

```
// Одномерный массив: объявление и инициализация
int array[] = {13, 10, 24, -17, 6, 85};
int len = array.length; // возвращает длину массива
System.out.println(array[6]); // ОШИБКА ВРЕМЕНИ ИСПОЛНЕНИЯ
```

```
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 6
at test.HelloWorldApplication.main(HelloWorldApplication.java:37)
```

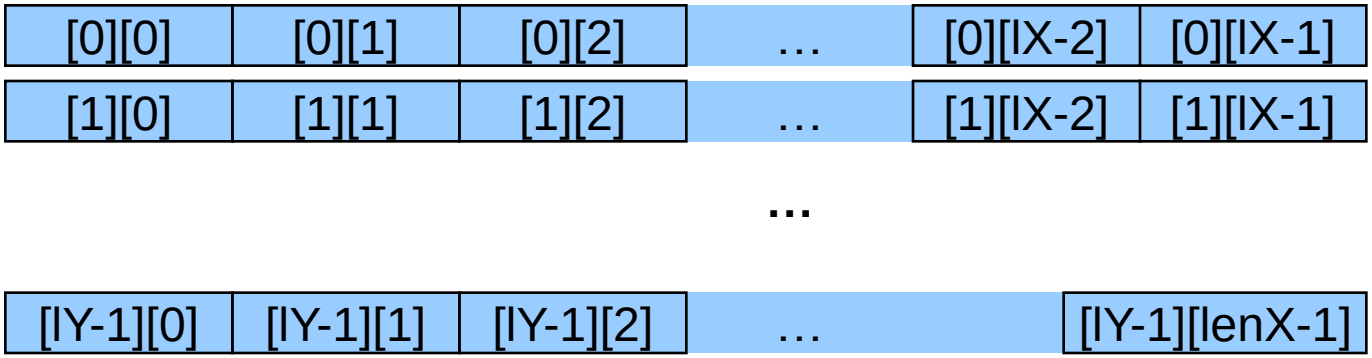
- Замечания:
  - Индексация имеет диапазон от 0 до length-1
  - Обращение к элементу массива – lvalue
  - Необходимо либо явно задавать значения массива, либо создавать массив с помощью `new`

Массивы в Java

Расположение элементов в памяти

Одномерный массив

Двумерный массив



Массивы в Java: массивы объектов

```
// Создание массива
ComplexNumber[] cNumbers = new ComplexNumber[5];


for(int i = 0; i<cNumbers.length; i++){
    // Создание элементов массива
    cNumbers[i] = new ComplexNumber();
}
//...
// вызов метода объекта,
// хранящегося в массиве
double a = cNumbers[0].getArg();
```

Строки в Java

- String – встроенный тип (класс) для работы со строками.
- Строки в Java – объекты класса String

```
String str;
str = new String("stop ");


char chars[] = {'a','u','g'};
String str2 = new String(chars);


String str3 = str+str2; // str3 = "stop aug"


String str4 = str+"codon "+str2; // str4 = "stop codon aug"
```

Строки в Java

```
System.out.println(str4.length());
System.out.println("Stop codon".length());
```

Длина строки

```
int age = 20;
String str = "Ему " + age + "лет";
```

Конкатенация строк
И других типов данных

```
char ch;
ch = str.charAt(2);
```

Извлечение символа из строки
(по индексу)

```
int start = 5;
int end = 10;
char buf[] = new char[end-start];
str4.getChars(start, end, buf, 0);
```

Извлечение нескольких
символов из строки
(по индексам)

Строки в Java

```
char[] chars = str4.toCharArray();
```

Преобразование в массив

```
String str1 = "STOP";
String str2 = "stop";
System.out.println(str1.equals(str2)); // false
System.out.println(str2.equalsIgnoreCase(str1)); // true
```

Сравнение строк – регистрозависимое и регистронезависимое

Строки в Java

```
String str1 = "STOP";
String str2 = "stop";
System.out.println(str1.startsWith("STO")); // true
System.out.println(str2.endsWith("top")); // true
```

Определение начала (префикса) и конца (суффикса) строки

```
String str1 = "start";
String str2 = "STOP";
System.out.println(str1.compareTo(str2)); // 32
System.out.println(str1.compareToIgnoreCase(str2)); // -14
```

Лексикографическое сравнение строк

Строки в Java

```
String str1 = "stararararart";
int fromIndex = 5;
System.out.println(str1.indexOf("ar")); // 2
System.out.println(str1.indexOf("ar", fromIndex)); // 6
System.out.println(str1.lastIndexOf("ar")); // 8
```

Поиск позиции (индекса) подстроки

```
int start = 5;
int end = 8;
String str1 = "stararararart";
System.out.println(str1.substring(start, end)); // "rar"
```

Выделение подстроки (по индексам)

Строки в Java

```
String org = "Это стоп-кодон, стоп";
String search = "стоп";
String subst = "старт";
String result = "";
int i;
do{ // заменить все совпавшие подстроки
    System.out.println(org);
    i = org.indexOf(search); // индекс первого вхождения
    if(i != -1){ // поиск удался
        result = org.substring(0,i);
        result = result + subst;
        result = result + org.substring(i+search.length());
        org = result;
    }
}while(i != -1);
```

Это стоп-кодон, стоп
Это старт-кодон, стоп
Это старт-кодон, старт

Строки в Java

```
String str1 = "Здравствуйте, товарищи!";
String str2 = str1.replace('р', 'г');
System.out.println(str2); // Здгавствуйте, товагищи!
```

Замена всех вхождений одного символа другим

```
String str1 = "Здравствуйте, товарищи!";
String str2 = str1.replace("вствуйте", "ссте");
System.out.println(str2); // Здрассте, товарищи!
```

Замена одной подстроки другой

Строки в Java

```
String str1 = "ttgaacatg";
if(str1.contains("atg")){
    System.out.println("Содержит старт-кодон");
}
else{
    System.out.println("Не содержит старт-кодон");
}
```

Проверяет наличие подстроки в строке

Строки в Java

```
String str1 = "What;do;you;mean?";
String str2 = str1.replaceAll(";", " ");
System.out.println(str2); // “What do you mean?”
```

Замена всех вхождений регулярного выражения строкой

```
String str1 = "What;do;you;mean?";
String str2 = str1.replaceFirst(";", " ");
System.out.println(str2); // What do;you;mean?
```

Замена первого вхождения регулярного выражения строкой

Строки в Java

```
String str1 =
"Instruction1;;;Instruction2;;;Instruction3;;;Instruction4";

String[] arr = str1.split(";;;");

System.out.println(arr.length);

for(int i = 0; i< arr.length; i++)

    System.out.println(arr[i]);
```

```
4
Instruction1
Instruction2
Instruction3
Instruction4
```

Разделяет строку на массив подстрок по ограничителю (регулярное выражение)

Строки в Java

```
String str1 =
"Instruction1;;;Instruction2;;;Instruction3;;;Instruction4";

String[] arr = str1.split(";;;",2);

System.out.println(arr.length);

for(int i = 0; i< arr.length; i++)

System.out.println(arr[i]);
```

```
2
Instruction1
Instruction2;;;Instruction3;;;Instruction4
```

Разделяет строку на массив подстрок по ограничителю (регулярное выражение), длина массива не больше числа, определяемого вторым аргументом

Строки в Java

```
String str1 = "gcgcgcgcgcgcgcgc";
String str2 = "gcgcgcgcgcgcgcga";
if(str1.matches("(gc)*")){
    System.out.println("str1: GC-повтор");
}else{
    System.out.println("str1: не GC-повтор");
}
if(str2.matches("(gc)*")){
    System.out.println("str2: GC-повтор");
}else{
    System.out.println("str2: не GC-повтор");
}
```

Проверяет, удовлетворяет ли строка регулярному выражению

```
str1: GC-повтор
str2: не GC-повтор
```

Строки в Java

```
String str1 = "ЭтА СТроКа нАПИсАнА КАк ПопалО";
String strU = str1.toUpperCase();
String strL = str1.toLowerCase();
System.out.println(strU);
System.out.println(strL);
```

ЭТА СТРОКА НАПИСАНА КАК ПОПАЛО
эта строка написана как попало

Переводит все символы строки
в верхний (нижний) регистр

Строки в Java

```
String str1 = "\t\t Строка содержит символы табуляции и
пробелов по краям   ";
String str2 = str1.trim();
System.out.println(str1);
System.out.println(str2);
```

```
                Строка содержит символы табуляции и пробелов по краям
Строка содержит символы табуляции и пробелов по краям
```

Обрезает символы пробелов и табуляции (whitespaces) по краям строки

Строки в Java: метод toString()

- Специальный метод, определённый в классе Object, используемый при конкатенации и печати строк, возвращающий «удобочитаемое» строковое представление объекта
  - Если метод не определён в классе, то при печати или конкатенации будет использоваться метод класса-родителя (вплоть до Object – печатающего адрес объекта и класс)

Строки в Java: метод toString()

Метод toString НЕ переопределён классе ComplexNumber

```
ComplexNumber a = new ComplexNumber(3.0,4.0);
ComplexNumber b = new ComplexNumber(1.0,2.0);
ComplexNumber c = a.add(b);
String str = a+" + "+b+" = "+c;
System.out.println(str);
```

ComplexNumber@9304b1 + ComplexNumber@190d11 = ComplexNumber@a90653

Строки в Java: метод toString()

```
public class ComplexNumber {
...
    public String toString(){
        return "("+this.real+"+i*"+this.image+")";
    }
}
```

```
ComplexNumber a = new ComplexNumber(3.0,4.0);
ComplexNumber b = new ComplexNumber(1.0,2.0);
ComplexNumber c = a.add(b);
String str = a+" + "+b+" = "+c;
System.out.println(str);
```

(3.0+i*4.0) + (1.0+i*2.0) = (4.0+i*6.0)

Лекция 5
Исключения, Работа с файлами.

Исключения в Java

- Исключение (exception) – аварийное состояние, которое возникает в программе во время выполнения (runtime); Пример: деление на нулю, выход за пределы массива
- Исключение в языке Java – объект, описывающий ошибочную (исключительную) ситуацию, произошедшую в некоторой части кода
- Когда возникает исключение, создаётся объект и «вбрасывается» в метод, вызвавший ошибку
  - Метод может обработать исключение
  - Метод может передать исключение в вызвавший его метод
- В некоторой точке исключение «захватывается» и обрабатывается

Исключения: генерация исключения

- Оператор throws в сигнатуре метода указывает на то, что метод может сгенерировать исключение. Если внутри метода генерируется исключение, этот оператор обязан присутствовать в сигнатуре
- Оператор throw непосредственно генерирует исключение (внутри метода)

Пример

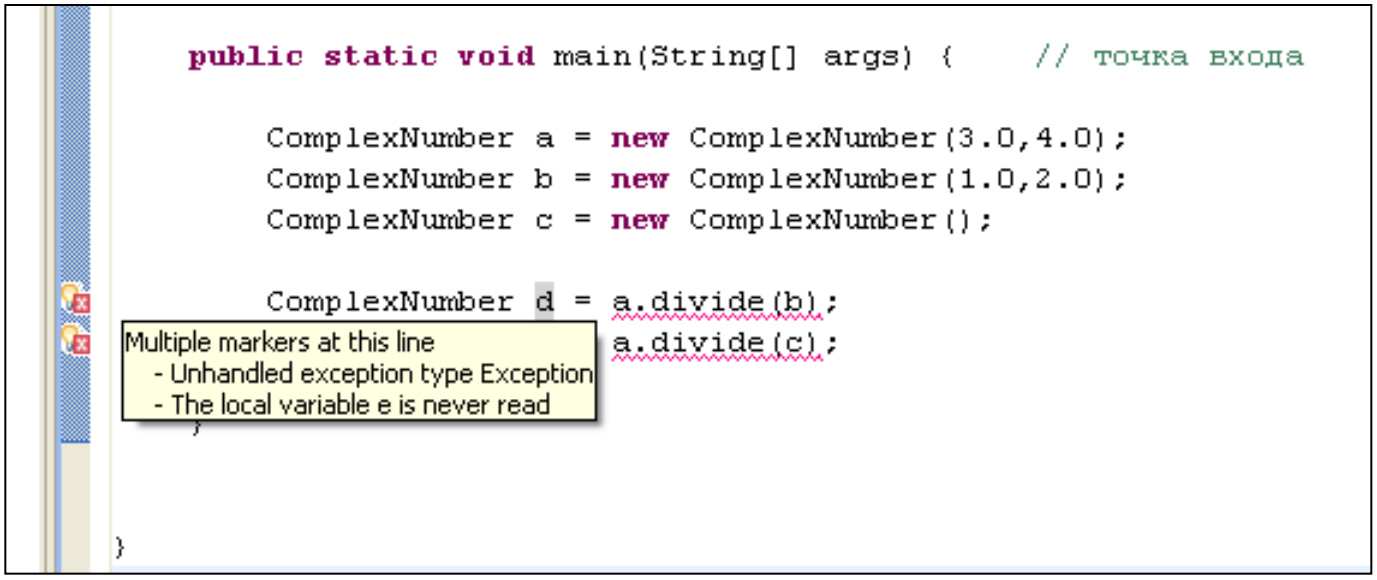
ComplexNumber.java

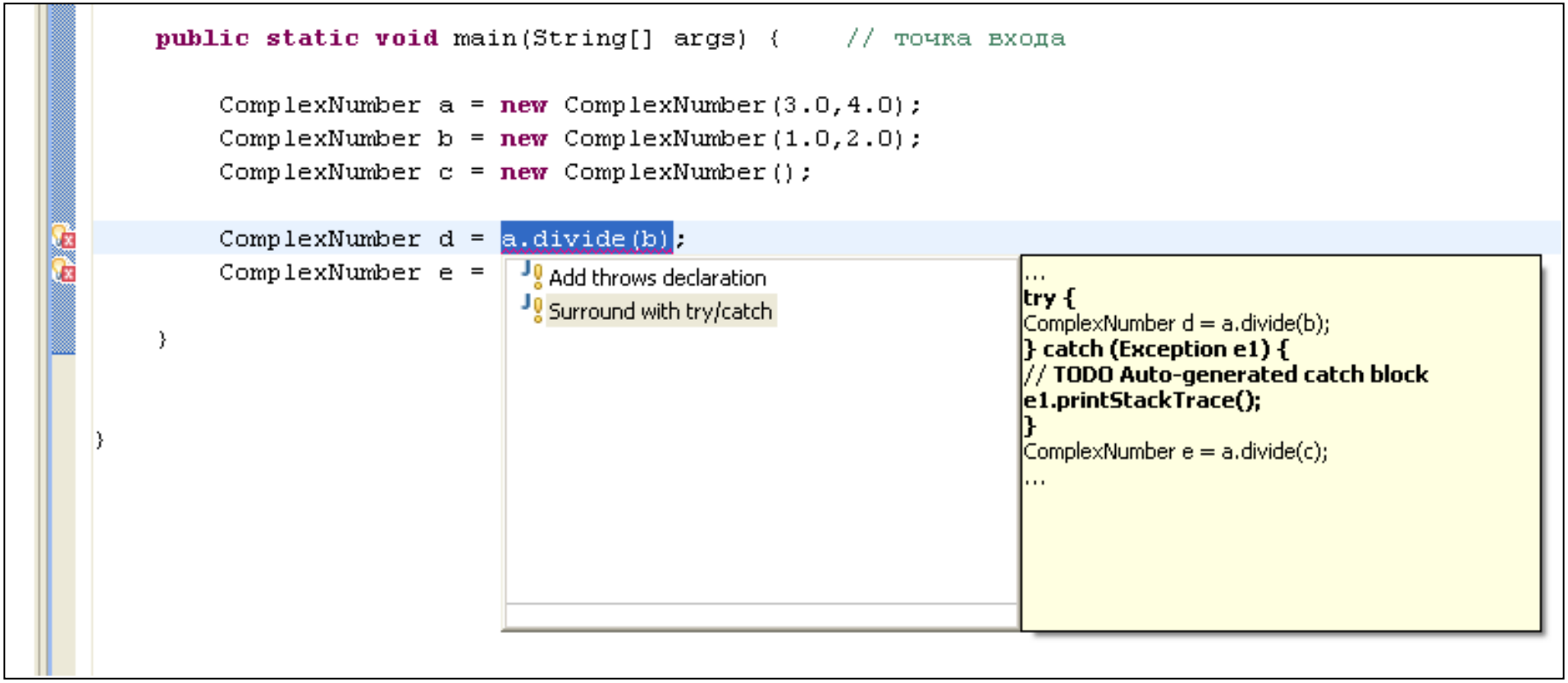
```
public ComplexNumber divide(ComplexNumber b) throws Exception{
        if((b.real==0) && (b.image == 0))
                throw new Exception();

        double denom = b.real*b.real+b.image*b.image;
        double re = (this.real*b.real+this.image*b.image)/denom;
        double im = (this.image*b.real-this.real*b.image)/denom;
        return new ComplexNumber(re,im);
}
```

Исключения: перехват исключения

- Оператор try определяет блок кода, который может вызвать ошибку исполнения и исключение
- Оператор catch определяет тип перехватываемого исключения и описывает обработчик исключения
- Оператор finally определяет блок кода, выполняющийся после блока try/catch, независимо от того, было исключение, или нет (необязательный блок)

```
public ComplexNumber divide(ComplexNumber b) throws Exception{
    if((b.real==0) && (b.image == 0))
        throw new Exception();
    double denom = b.real*b.real+b.image*b.image;
    double re = (this.real*b.real+this.image*b.image)/denom;
    double im = (this.image*b.real-this.real*b.image)/denom;
    return new ComplexNumber(re,im);
}
```




public static void main(String[] args) {   // точка входа
ComplexNumber a = new ComplexNumber(3.0,4.0);
ComplexNumber b = new ComplexNumber(1.0,2.0);
ComplexNumber c = new ComplexNumber();
ComplexNumber d = a.divide(b);
ComplexNumber e =
Add throws declaration
Surround with try/catch
}
}
...
try {
ComplexNumber d = a.divide(b);
} catch (Exception e1) {
// TODO Auto-generated catch block
e1.printStackTrace();
}
ComplexNumber e = a.divide(c);
...

Пример

Program.java

```
ComplexNumber a = new ComplexNumber(3.0,4.0);
ComplexNumber b = new ComplexNumber(1.0,2.0);
ComplexNumber c = new ComplexNumber();

try {
    ComplexNumber d = a.divide(b);
    System.out.println("d = "+d);
    ComplexNumber e = a.divide(c);
    System.out.println("e = "+e);
} catch (Exception e1) {
    // Здесь принимаются меры для борьбы с ошибкой
    //(повторный ввод,подстановка значения по умолчанию и т.д.)
    e1.printStackTrace(); // Печать стека вызовов
}
```

Пример

```
d = (2.2-i*0.4)

java.lang.Exception

at test.ComplexNumber.divide(ComplexNumber.java:27)

at test.Program.main(Program.java:18)
```

Исключения : общая форма обработки исключения

```
try{
// Контроллируемый блок кода
}
catch(ExceptionType1 ex){
// обработчик исключения для ExceptionType1
}
catch(ExceptionType2 ex){
// обработчик исключения для ExceptionType2
}
//...
finally{
// блок кода для обработки перед возвратом
}
```

Ввод/вывод Java

- Поток (stream) – программная абстракция, используемая для ввода/вывода данных
- Java-программы выполняют ввод/вывод через потоки
- Существуют потоки ввода (клавиатура, файлы, интернет-соединения (сокеты), и т.д.) и потоки вывода (экран, файл, принтер, звуковая карта и т.д.)

Стандартные потоки

`System.`*`in`*	Стандартный поток ввода (клавиатура)
`System.`*`out`*	Стандартный поток вывода (консоль)
`System.`*`err`*	Стандартный поток ошибок (консоль)

Перенаправление потоков

Program.java

```
public static void main(String[] args) {
    System.out.println("Hello");
}
```

```
> Java Program
Hello
```

Выводит “Hello” на экран

```
> Java Program > out.txt
Hello
```

Записывает “Hello” в файл out.txt

Типы потоков в Java

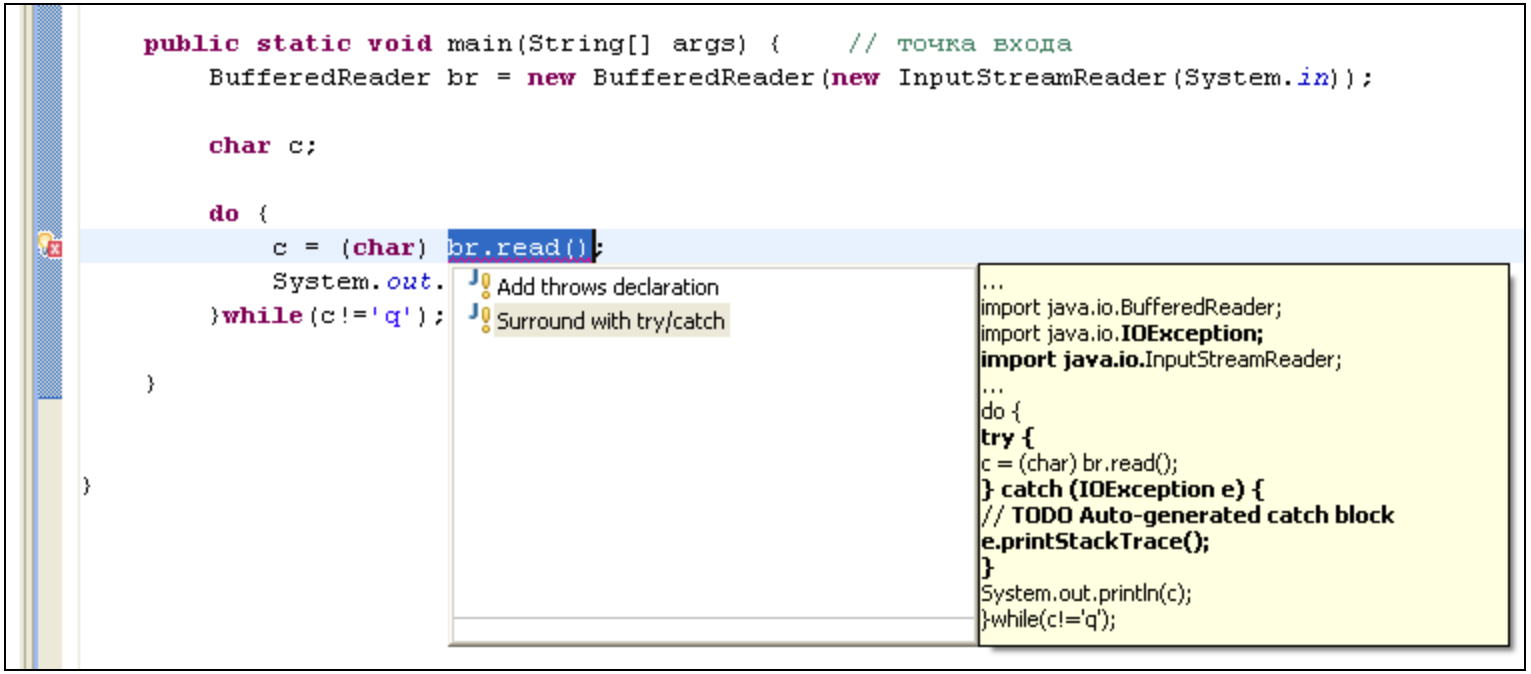
Байтовые	Символьные
`InputStream`	`Reader`
`OutputStream`	`Writer`

Ввод с клавиатуры: чтение символов

```
public static void main(String[] args) {
    BufferedReader br = new BufferedReader(new
       InputStreamReader(System.in));
    char c;
    do {
        c = (char) br.read();
        System.out.println(c);
    }while(c!='q');
}
```

```
public static void main(String[] args) {    // точка входа
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

    char c;

    do {
        c = (char) br.read();
        System.out.println(c);
    }while(c!='q');

}
}
```

Ввод с клавиатуры: чтение символов



Ввод с клавиатуры: чтение строк

```
public static void main(String[] args) {
BufferedReader br = new BufferedReader(new
    InputStreamReader(System.in));
    String str="";
    do {
        try {
            str = br.readLine();
       } catch (IOException e) {
            e.printStackTrace();
       }
      System.out.println(str);
   }while(!str.equals("stop"));
}
```

Работа с файлами: чтение

Program.java

```
public static void main(String[] args) {
BufferedReader reader;
try {
reader = new BufferedReader(new FileReader("seq.txt"));
String line = "";
while ((line = reader.readLine()) != null) {
    System.out.println("Line: "+line+
    " : has "+line.length()+" length");
}
}
catch (IOException e) {
    e.printStackTrace();
}
}
```

seq.txt

```
aattgc
gggcca
actgctag
```

```
Line: aattgc : has 6 length
Line: gggcca : has 6 length
Line: actgctag : has 8 length
```

Работа с файлами: запись

Program.java

```
public static void main(String[] args) {
    PrintStream out;
    try {
        out = new PrintStream("out.txt");
        for(int i = 1; i <= 100; i++)
            out.println(i*i);
    }
    catch (IOException e) {
        e.printStackTrace();
    }
}
```

out.txt

```
1
4
9
16
...
9604
9801
10000
```

Лекция 6
Синтаксический анализ. Контейнеры.

Аргументы командной строки

- Программа может запускаться с определёнными параметрами – аргументами командной строки (command-line arguments)
- Аргументы командной строки позволяют пользователю влиять на исполнение приложения для одного конкретного вызова
- Аргументы могут быть числами, буквами и строками
- Аргументов может быть несколько

```
copy file1 file2
mkdir dir1

java HelloWorldApplication
```

Аргументы командной строки

- В java аргументы командной строки передаются через параметр метода main (массив строк – String[] args

Program.java

```
public static void main(String[] args) {
    for(int i = 0; i < args.length; i++){
        System.out.println(args[i]);
    }
}
```

```
> java Program
```

```
> java Program 1 file2 file3.txt 5 4
1
file2
file3.txt
5
4
```

Аргументы командной строки. Сложение двух целых чисел

```
public class Sum {
public static void main(String[] args) {
    System.out.println(args[0] + args[1]);
}}
```

```
> java Sum 1 2
12
```

```
public class Sum {
public static void main(String[] args) {
    int m = args[0];
    int n = args[1];
    System.out.println((m+n));
}}
```

Ошибка

Аргументы командной строки. Сложение двух целых чисел

```
public class HelloWorldApplication {
    public static void main(String[] args) {
        int m = Integer.parseInt(args[0]);
        int n = Integer.parseInt(args[1]);
        System.out.println((m+n));
    }
}
```

```
> java Sum 1 2
3
```

Разбор числовых аргументов командной строки

- Преобразование строки в число осуществляется с помощью методов parseXXX (где XXX – конкретный числовой класс)
- При невозможности преобразования генерируется исключение NumberFormatException:

```
Integer.parseInt(String s);

Long.parseLong(String s);

Double.parseDouble(String s);

Float.parseFloat(String s);
```

Основы синтаксического анализа

- Синтаксический анализ (syntax analysis, парсинг, parsing) – процесс анализа входной последовательности символов с целью разбора грамматической структуры
- Синтаксический анализатор (парсер, parser) – это программа или часть программы, выполняющая синтаксический анализ
- При парсинге исходный текст преобразуется в некую структуру данных

Биоинформатика как анализ генетических текстов

- поиск гомологии и выравнивание генетических текстов, множественное выравнивание
- статистический анализ генетических текстов, исследование структуры повторов и модели порождения символьных последовательностей, сегментация геномов;
- предсказание кодирующих участков генов и открытых рамок считывания;
- предсказание функциональных сигналов (функциональных сайтов и регуляторных районов);
- анализ вторичной структуры РНК и сигналов трансляции;

Биоинформатика как анализ генетических текстов

>gi|108885074|ref|NC_000908.2| Mycoplasma genitalium G37, complete genome

TAAGTTATTATTTAGTTAATACTTTTAACAATATTATTAAGGTATTTAAAAAATACTATTATAGTATTTA

ACATAGTTAAATACCTTCCTTAATACTGTTAAATTATATTCAATCAATACATATATAATATTATTAAAAT

ACTTGATAAGTATTATTTAGATATTAGACAAATACTAATTTTATATTGCTTTAATACTTAATAAATACTA

CTTATGTATTAAGTAAATATTACTGTAATACTAATAACAATATTATTACAATATGCTAGAATAATATTGC

TAGTATCAATAATTACTAATATAGTATTAGGAAAATACCATAATAATATTTCTACATAATACTAAGTTAA

TACTATGTGTAGAATAATAAATAATCAGATTAAAAAAATTTTATTTATCTGAAACATATTTAATCAATTG

AACTGATTATTTTCAGCAGTAATAATTACATATGTACATAGTACATATGTAAAATATCATTAATTTCTGT

TATATATAATAGTATCTATTTTAGAGAGTATTAATTATTACTATAATTAAGCATTTATGCTTAATTATAA

GCTTTTTATGAACAAAATTATAGACATTTTAGTTCTTATAATAAATAATAGATATTAAAGAAAATAAAAA

AATAGAAATAAATATCATAACCCTTGATAACCCAGAAATTAATACTTAATCAAAAATGAAAATATTAATT

AATAAAAGTGAATTGAATAAAATTTTGAAAAAAATGAATAACGTTATTATTTCCAATAACAAAATAAAAC

...

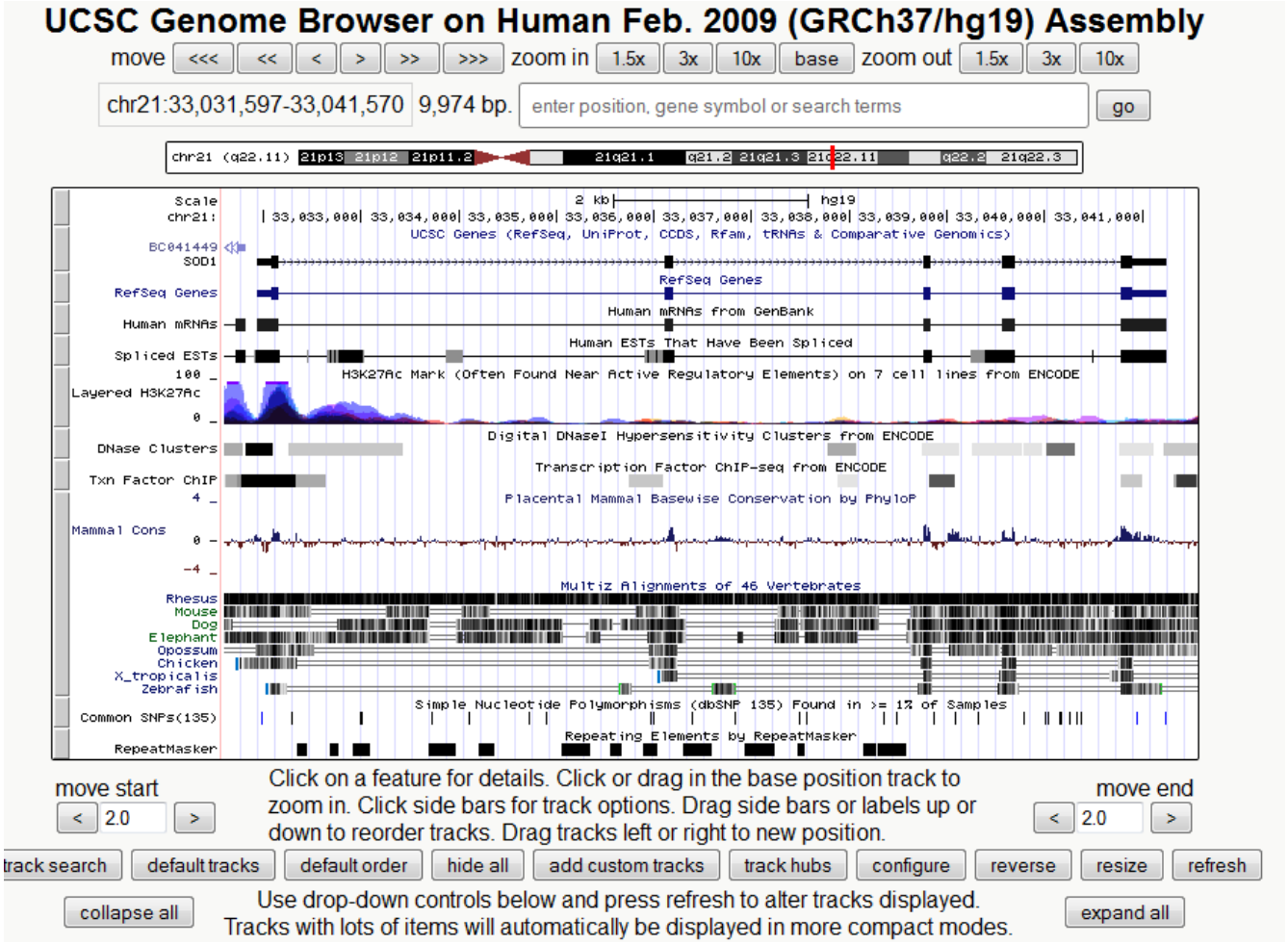
Биоинформатика как анализ генетических текстов. Выравнивание

```
A00825_hsp17_L.Merr.          TTACAATCTCCCTAGTTTC---TAATCTCAGCTAAGAAAA-ACCAAAAGA
A00826_hsp17.5-M_L.Merr.      TTTTGATCTCCCAAGTTTC---AAATCTCGCGAATAAATATATCAAAAGA
A00662_mult.r.g.1b_Rat        GCCGCTGCTCCCATCTTC----GAGGCTCAGCTCAACTCAGAGCTACTT-
A00172_mult.r.g.1b_Mus        GCCGCTGCTTCCATCTTCT---GAGGTTCCGCTCAACTCAGAGCTACT--
A00597_M35021_70hsp_Mus       TCCAGAG---ACAAGCGAA---GACAAGAGAAGCAGAGC-GAGCGGCGC-
A00597_M76613_70hsp_Mus       TCCAGAG---ACAAGCGAA---GACAAGAGAAGCAGAGCAGAGCGGCGC-
A00551_heme_ox.1_Homo         GCACGAA----CGAGCCC-----AGCACCGGCCGGATGGAGCGTCCGCAA
A00569_heme_ox.1_Rat          GCCGGAG----CAGAGCC-----ATCTCGAGCGGAGCCCGGAGCCTGAAG
A00552_hsp70_Xenopus          TACTTAC----TGGGCAA-----AGACGCAGCTGCGCATATTCTAGCGAA
A00553_hsp70_Xenopus          TACTTAC----TGGGCAA-----AGACGCAGCTGCGCATATTCTAGCGAA
A00973_hsp17.3B_L.Merr.       TTTCGAC----CCTTTCTC---CCTCGATGTGTGGGACCCCTTCAAGGA-
A00171_multid.r.g._Homo       GCCGCTGTTCGTTTCCTTTAG-GTCTTTCCACTAAAGTCGGAGTATCTT-
A00527_sm.hsp18_L.Merr.       GTTCGAT----CCTTTCTC---ACTGGACGTGTGGGATCCCTTCAAGGA-
A00570_hs-90B_Gallus          GGTCGGGGCTGGTCGCGTGGG-CCGTTATCGCTGTAGCCTTGGTGCAAAC
A01169_hsp90a._Gallus         GACCGACAGCCCTTCCCC-----CGCTGCCAAGGTGAGCGGCGGTAGAGC
A00761_hsp70B_Homo            CACTGCTGA-GCGCCCCT-----CGACGCGGGCGGCAGCAGCCTCCGTGG
```

Биоинформатика как анализ генетических текстов. Genbank

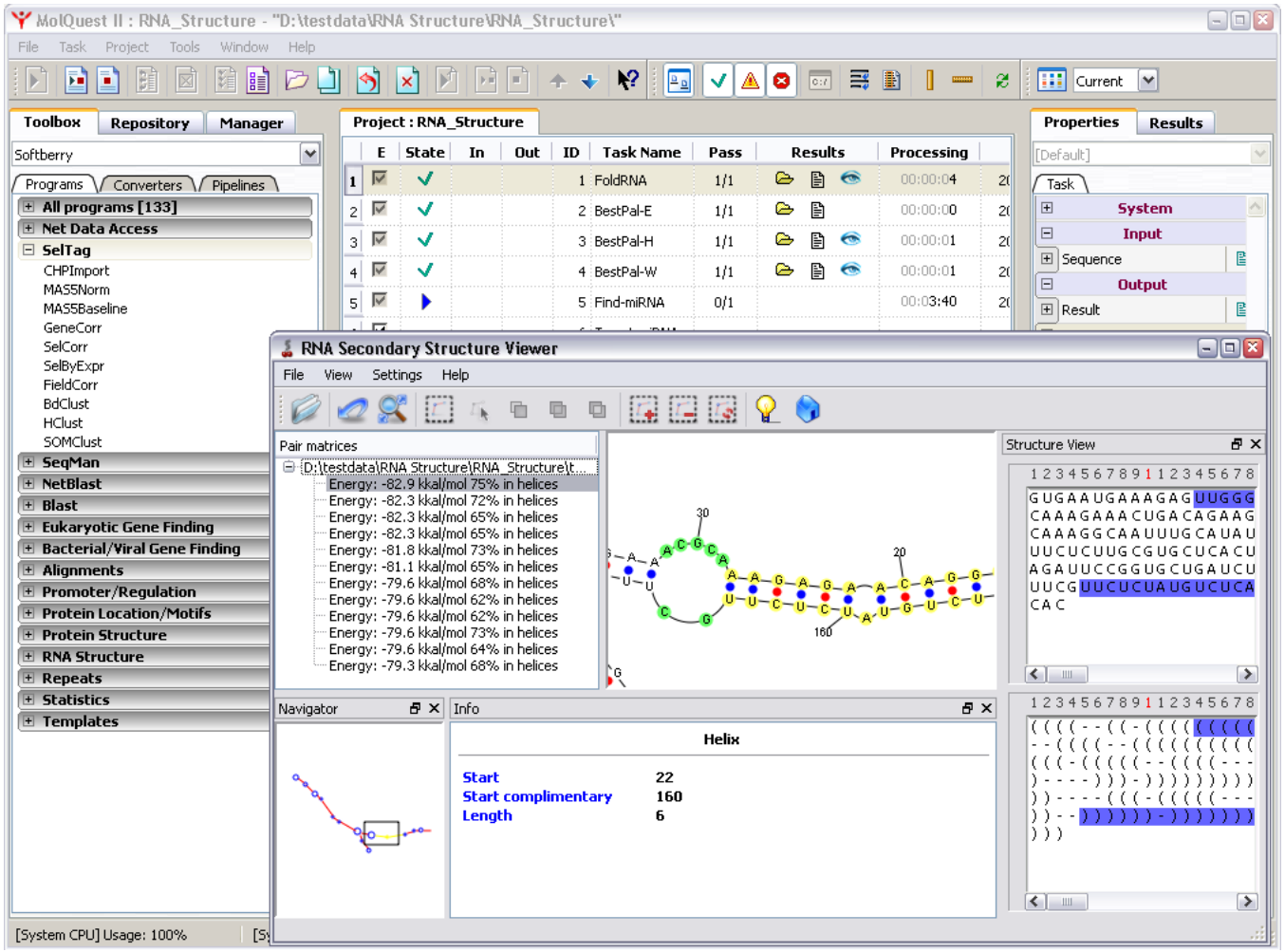
```
LOCUS       NC_009926          374161 bp    DNA     circular BCT 09-DEC-2007
FEATURES             Location/Qualifiers
     source          1..374161
                     /organism="Acaryochloris marina MBIC11017"
                     /mol_type="genomic DNA"
                     /strain="MBIC11017"
                     /db_xref="taxon:329726"
                     /plasmid="pREB1"
                     /note="type strain of Acaryochloris marina"
     gene            join(373616..374161,1..174)
                     /locus_tag="AM1_A0001"
                     /db_xref="GeneID:5685236"
     CDS             join(373616..374161,1..174)
                     /locus_tag="AM1_A0001"
                     /codon_start=1
                     /transl_table=11
                     /product="glutathione S-transferase, putative"
                     /protein_id="YP_001520660.1"
                     /db_xref="GI:158339653"
                     /db_xref="GeneID:5685236"
                     /translation="MKIVSFKICPFVQRVTALLEAKGIDYDIEYIDLSHKPQWFLDLS
                     PNAQVPILITDDDDVLFESDAIVEFLDEVVGTPLSSDNAVKKAQDRAWSYLATKHYLV
                     QCSAQRSPDAKTLEERSKKLSKAFGKIKVQLGESRYINGDDLSMVDIAWLPLLHRAAI
                     IEQYSGYDFLEEFPKVKQWQQHLLSTGIAEKSVPEDFEERFTAFYLAESTCLGQLAKS
                     KNGEACCGTAECTVDDLGCCA"
     gene            241..381
```

Геномные браузеры – анализ генетических текстов



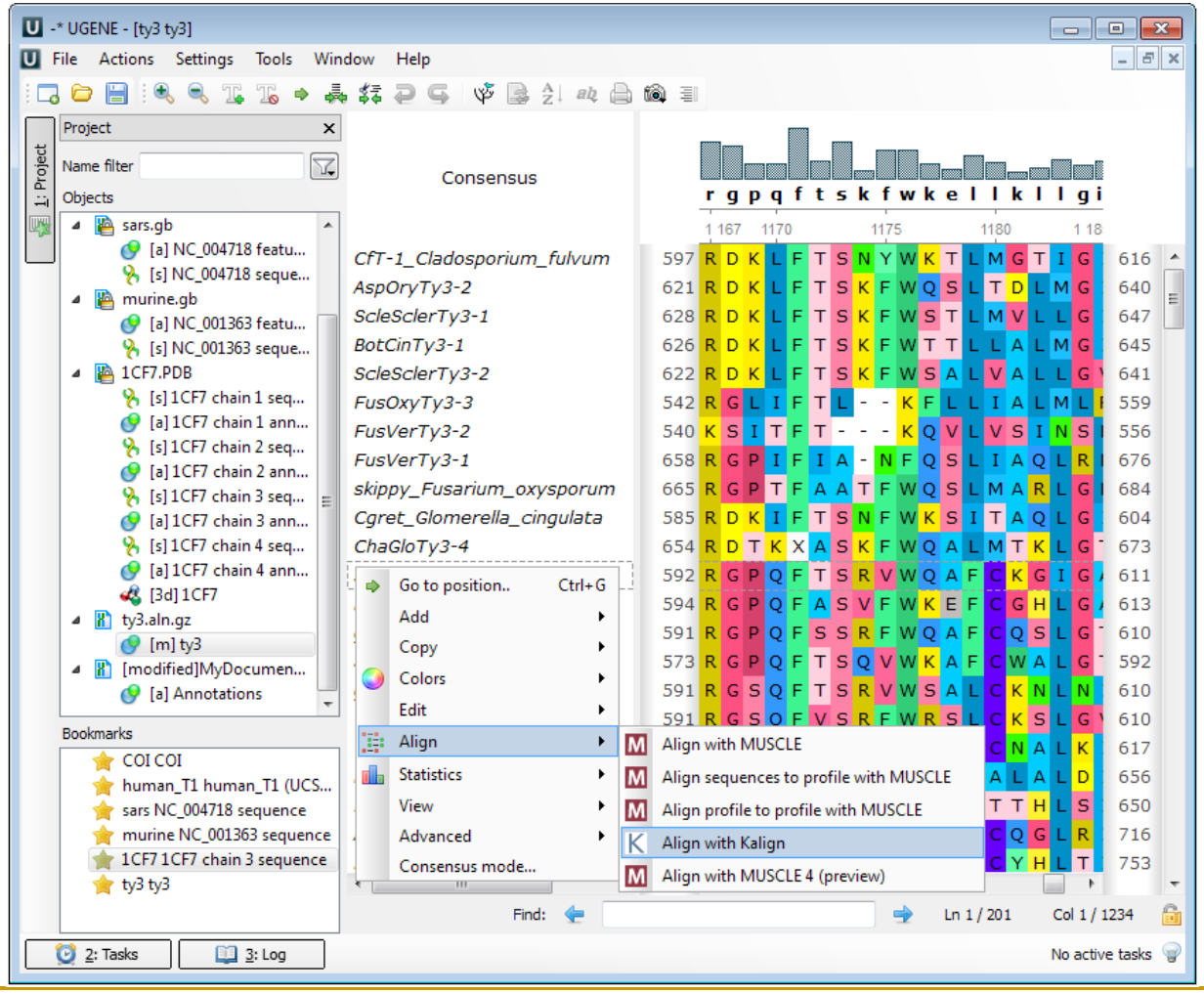
UCSC Genome Browser(http://genome.ucsc.edu/cgi-bin/hgTracks?hgsid=298750707)

Геномные браузеры – анализ генетических текстов



Softberry Molquest (http://www.molquest.com/)

Геномные браузеры – анализ генетических текстов



Unipro Ugene (http://ugene.unipro.ru/rus/)

Разбиение строки на элементы

- Tokenizing – процесс разбиения строки символов на последовательность значащих элементов, слов («tokens»), раздёленных определёнными (разделяющими) символами

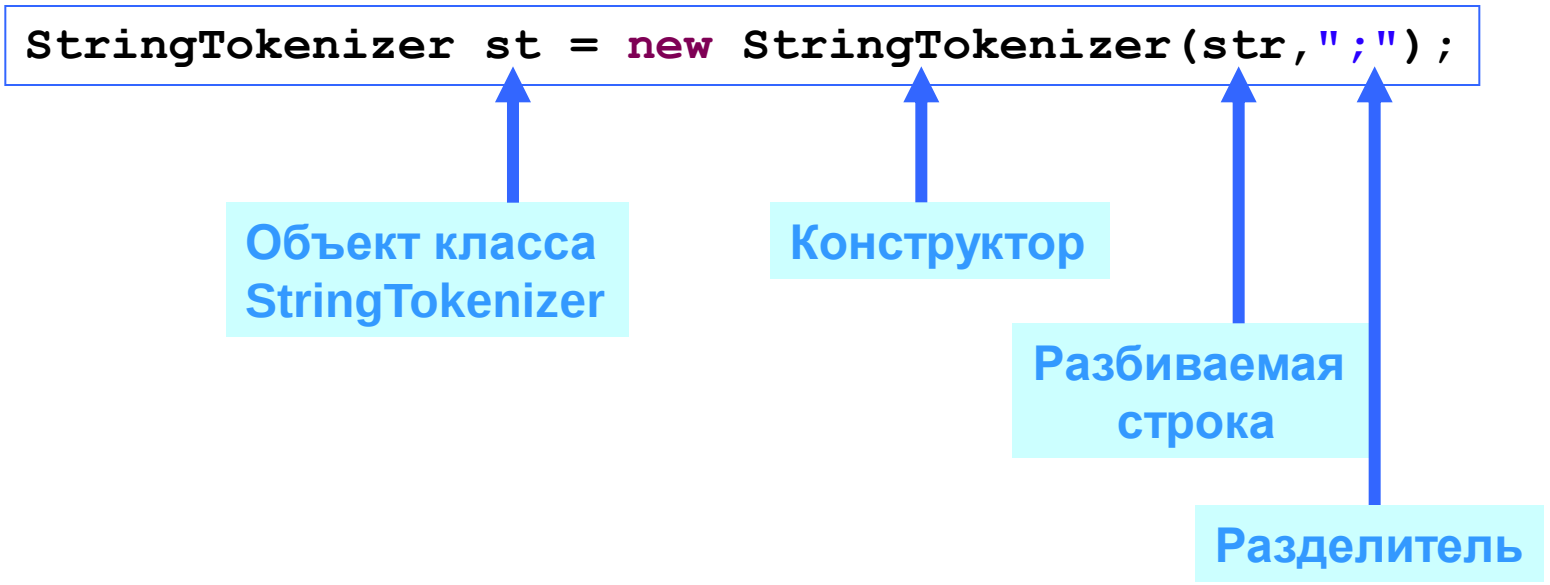
double A = 10; int B = 20; long C = 30; float D = 50;

“;“
double A = 10
int B = 20
long C = 30
float D = 50

“=“
double A
10; int B
20; long C
30; float D
50;

Разбиение строки на элементы

- Класс StringTokenizer – реализует разбиение строки на элементы (tokens) по разделяющим символам (delimiters)



Класс String Tokenizer, методы

int countTokens() – возвращает количество элементов (токенов)

```
String str = "A = 10; B = 20; C = 30; D = 50;";
StringTokenizer st1 = new StringTokenizer(str,";");
System.out.println("tokens "+st1.countTokens());
```

st1.countTokens() = 4

String nextToken() – возвращает следующий элемент (токен), переходит на следующую позицию

```
String str = "A = 10; B = 20; C = 30; D = 50;";
StringTokenizer st1 = new StringTokenizer(str,";");
System.out.println("st1.nextToken(): " + st1.nextToken());
```

st1.nextToken(): A = 10

String nextToken(String delim) – возвращает следующий элемент (токен), отделённый символом из группы delim, переходит на следующую позицию

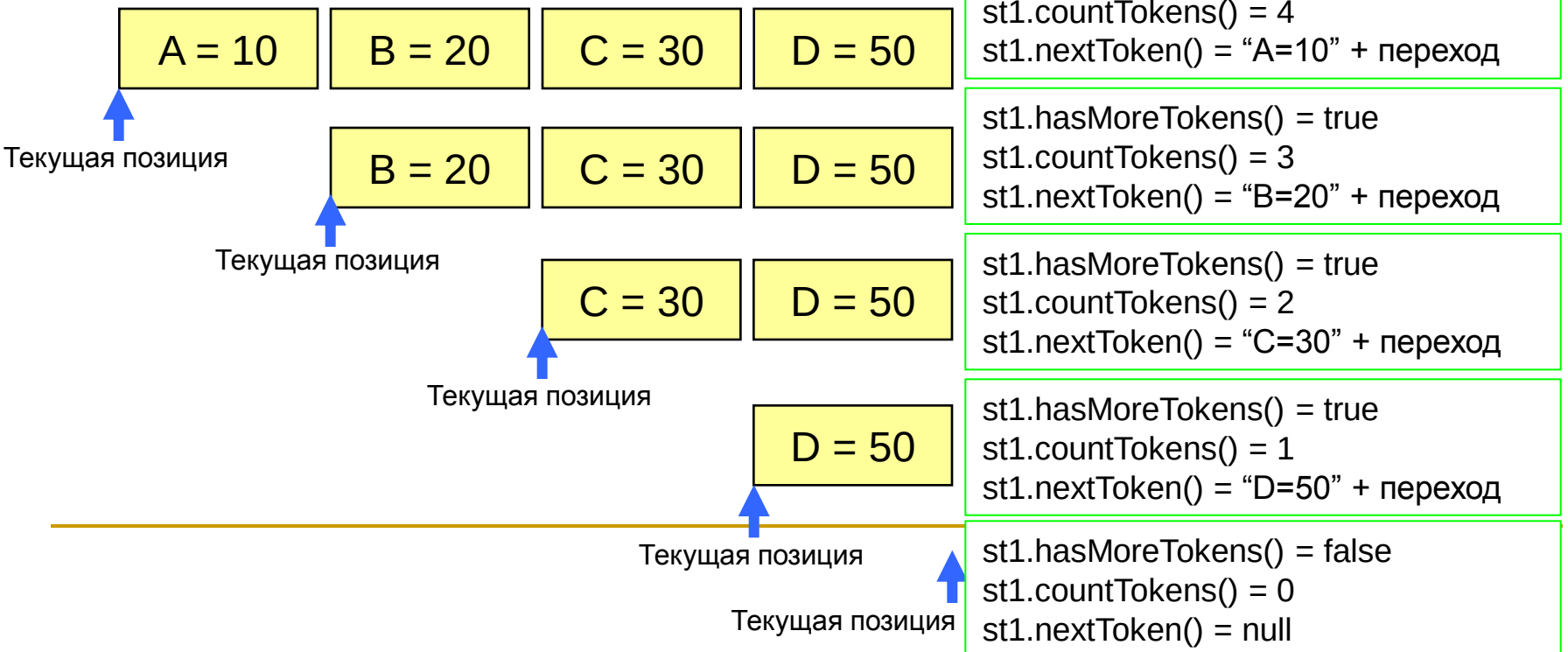
```
String str = "A = 10; B = 20; C = 30; D = 50;";
StringTokenizer st1 = new StringTokenizer(str,";");
System.out.println("st1.nextToken(\"=\"): " +
st1.nextToken("="));
```

st1.nextToken("="): A

Класс StringTokenizer, методы

boolean hasMoreTokens() – возвращает true, если в строке ещё есть непрочитанные токены, иначе возвращает false

```
String str = "A = 10; B = 20; C = 30; D = 50;";

StringTokenizer st1 = new StringTokenizer(str,";");

while (st1.hasMoreTokens()){

    System.out.println(st1.nextToken());

}
```



Класс String Tokenizer

В методе String nextToken(String delim) разделителем может выступать любой символ из строки delim (delim рассматривается как группа разделителей):

```
String str = "A = 10; B = 20: C = 30. D = 50;";

StringTokenizer st1 = new StringTokenizer(str,";:.");

while (st1.hasMoreTokens()){

    System.out.println(st1.nextToken());

}
```

double A = 10
int B = 20
long C = 30
float D = 50

Контейнеры в Java

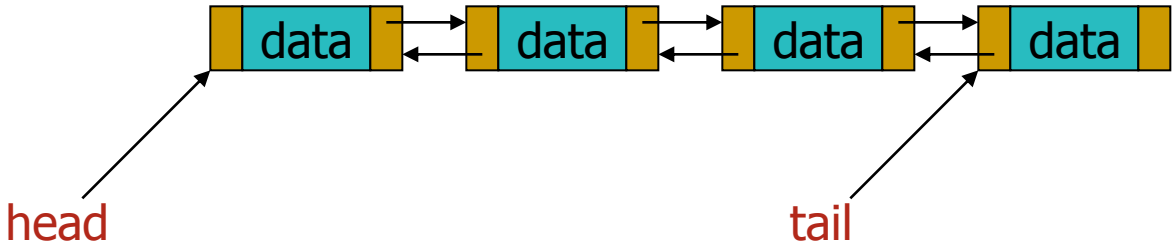
- Контейнер – это структура данных, позволяющая хранить объекты определённых типов (данные) и совершать следующие операции:
  - Вставка элемента
  - Удаление элемента
  - Обращение к элементу
- В Java есть два основных вида контейнеров: с жёсткой типизацией (хранятся объекты одного типа) и без типизации (могут храниться объекты разных типов)
- Контейнеры с жёсткой типизацией построены на основе механизма шаблонов
- Виды контейнеров:
  - Списки
  - Ассоциативные массивы
  - Деревья
  - а также стеки, очереди и т.д. …

Интерфейсы и абстрактные классы

- Интерфейс – сущность, подобная классу, описывающая, что должен делать класс, который реализует ее, но не определяющая это поведение.
- Абстрактный класс – такой класс, который не может иметь непосредственных экземпляров.
- Абстрактный метод – виртуальный метод, не имеющий реализации.
- Класс, имеющий хотя бы один абстрактный является абстрактным.

Список (List)

- Список (list) – контейнер, в котором каждый элемент содержит в себе данные, а также ссылки на предыдущий и последующий элементы



Список (List)

- List<Type> – абстрактный класс списка в Java
- Является шаблонным типом (хранит элементы типа Type)
- специфицирует операции:

void add(Type obj) void add(int index, Type obj)	Добавление элемента в список (в конец списка, или в позицию index)
void remove(Type obj) void remove(int index)	Удаление элемента из списка (задаётся либо элемент, либо индекс
Type get(int index)	Доступ к элементу списка по индексу (возвращается ссылка на объект)
Type set(int index, Type obj)	Изменение значения элемента списка по индексу (возвращается ссылка на старое значение)

Список (List)

int size()	Возвращает размер списка
int indexOf(Type obj) int lastIndexOf(Type obj)	Возвращает индекс объекта (первый/последний)
boolean isEmpty()	Проверяет, пуст ли список
boolean contains(Type obj)	Проверяет, содержит ли список элемент
void clear()	Очищает список
List<Type> subList(int from, int to)	Возвращает часть, ограниченную индексами

```
List<ComplexNumber> l = new ArrayList<ComplexNumber>();
ComplexNumber b = new ComplexNumber();
ComplexNumber c = new ComplexNumber(10,10);
l.add(b);
l.add(new ComplexNumber(-1,-2));
l.add(1,new ComplexNumber(10,10));

System.out.println("Размер: "+l.size());
System.out.println("Содержит "+b+" ? "+(l.contains(b)?"да": "нет"));
System.out.println("Содержит "+c+" ? "+(l.contains(c)?"да": "нет"));

l.set(2, new ComplexNumber(3,4));

for(int i = 0; i < l.size(); i++){
    System.out.println(l.get(i));
}
```

```
Размер: 3
Содержит (0.0+i*0.0) ? да
Содержит (10.0+i*10.0) ? нет
(0.0+i*0.0)
(10.0+i*10.0)
(3.0+i*4.0)
```

Контейнеры

- Ассоциативный массив (словарь, хэш) – контейнер, позволяющий хранить пары «ключ-значение»

Телефонная книга:
Вася 333222
Петя 222333
Оля 232323
Васина мама 333222

Класс Map

- Map<Key, Value> - абстрактный класс ассоциативного массива в Java
- Является шаблонным типом двух аргументов (Key – ключ, Value – значение)
- Специфицирует операции:

V put(Key k, Value v)	Добавление пары k-v
V get(Key k)	Возвращает значение, ассоциированное с ключом
V remove(Key k)	Удаляет элемент, чей ключ равен k
void clear()	Удаляет все пары ключ-значение
boolean isEmpty()	Проверяет, пустая ли карта
int size()	Возвращает число пар ключ-значение в карте
boolean containsKey(Key k)	Проверяет, есть ли в карте ключ k
boolean containsValue(Value v)	Проверяет, есть ли в карте значение v

Класс Map

```
Map<String, Integer> m = new HashMap<String, Integer>();


m.put("Иван",555444);

m.put("Марья", 111222);

m.put("Николай Александрович", 3333412);

m.put("318", 3333119);



System.out.println("Телефон Колчанова Н.А.:
"+m.get("Николай Александрович"));
```

Телефон Колчанова Н.А.: 3333412

Лекция 7
Запуск внешних программ. Организация программных пакетов.

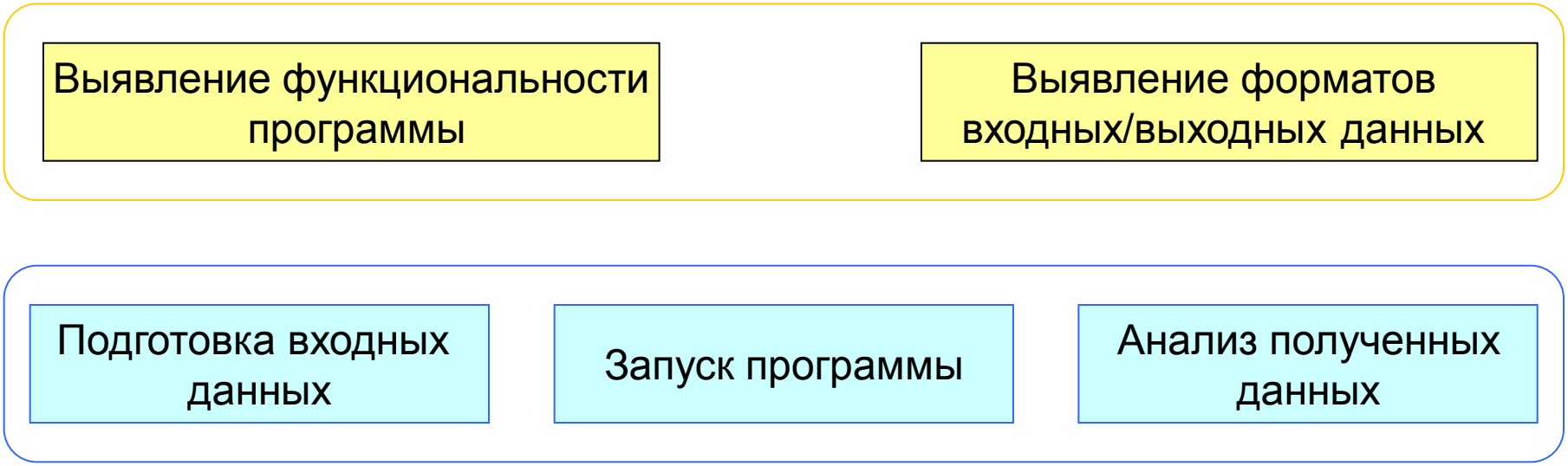
Многопоточность

- Многопоточность – свойство операционной системы или приложения, состоящее в том, что процесс, порождённый в операционной системе, может состоять из нескольких потоков (threads), выполняющихся «параллельно», то есть без предписанного порядка во времени
- Такие потоки называют также потоками команд (в противоположность потокам данных) или потоками выполнения (англ. thread of execution); иногда называют «нитями» («тредами»)

Использование сторонних программ в собственных программах

- Мотивация: во время работы возникает необходимость использования сторонних разработок (программ, алгоритмов и т.д.). Это можно сделать двумя основными способами:
  - Использование классов и функций из библиотек с открытым API
  - Непосредственное использование (запуск) программ и последующее использование данных, сгенерированных этими программами

Использование сторонних программ (схема)



Порождение процесса внутри программы.

- Класс ProcessBuilder – обеспечивает запуск и управление внешними процессами (программами)
- Для запуска процесса необходимо
  - создать объект класса ProcessBuilder, указав имя программы и необходимые аргументы
  - вызвать метод start()
- Конструкторы класса:

```
ProcessBuilder(List<String> args)

ProcessBuilder(String… args)
```

Порождение процесса внутри программы. Простейший пример

```
public static void main(String[] args) {


ProcessBuilder pb = new ProcessBuilder("notepad");
try {
    pb.start();
} catch (IOException e) {
    e.printStackTrace();
}
}
```

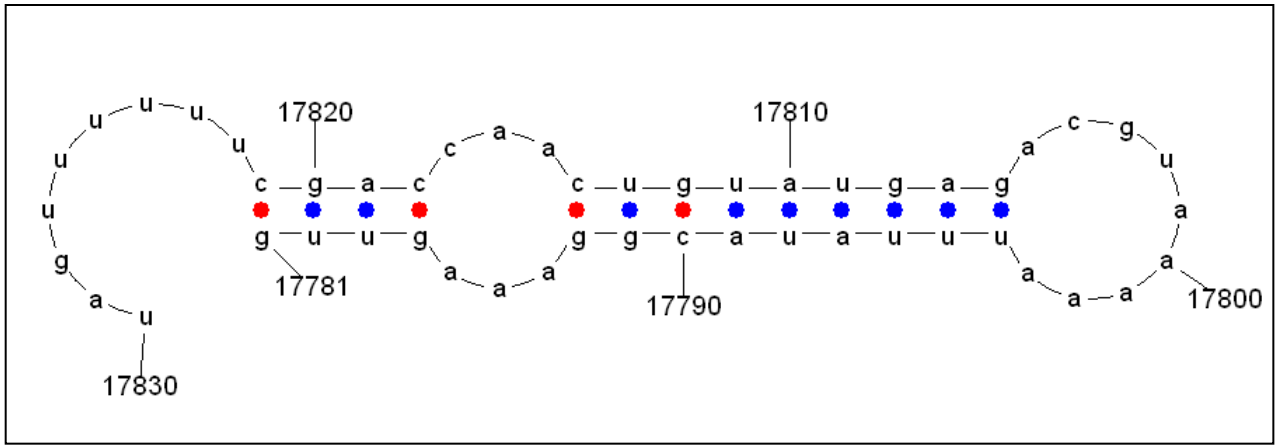
Порождение процесса внутри программы. Настройки

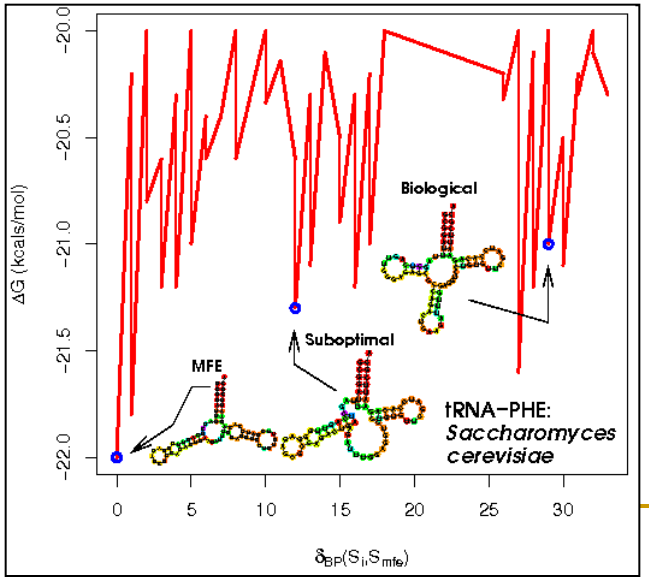
ProcessBuilder directory(File dir)	Устанавливает рабочий каталог вызываемой программы
File directory()	Возвращает текущий рабочий каталог вызываемой программы
ProcessBuilder redirectErrorStream(boolean m)	Если m == true, перенаправляет поток ошибок в стандартный поток
boolean redirectErrorStream()	Возвращает true, если поток ошибок перенаправлен в стандартный поток вывода

Подход к решению задач биоинформатики с привлечением сторонних программ

- Анализ задачи
  - Чёткая формулировка задания. Выделение отдельных (атомарных) стадий
  - Поиск существующих решений (программ, методов, алгоритмов и т.д.)
- Анализ найденных программ (методов): изучение возможности применения
  - Анализ входных и выходных данных
  - Оценка временных и аппаратных затрат
- Встраивание сторонней программы (метода) в собственную программу
  - Написание драйверов (генераторов-«переводчиков» данных) из(в) собственного формата в(из) формат сторонней программы
  - Внедрение полученной функциональности в собственную программу

Пример – построение вторичной структуры РНК с получением её энергии





http://en.wikipedia.org/wiki/Image:Yeast.png

Пример. Поиск существующих решений

http://en.wikipedia.org/wiki/List_of_RNA_structure_prediction_software

Single sequence structure prediction [edit]

Name	Description	Knots	Links	References
CONTRAfold	Secondary structure prediction method based on conditional log-linear models (CLLMs), a flexible class of probabilistic models which generalize upon SCFGs by using discriminative training and feature-rich scoring.	no	sourcecode webserver	[1]
KineFold	Folding kinetics of RNA sequences including pseudoknots.	yes	linuxbinary, webserver	[2][3]
MC-Fold MC-Sym Pipeline	Thermodynamics and Nucleotide cyclic motifs for RNA structure prediction algorithm. 2D and 3D structures.	yes	sourcecode, webserver	[4]
Mfold	MFE RNA structure prediction algorithm.	no	sourcecode, webserver	[5]
Pknots	A dynamic programming algorithm for optimal RNA pseudoknot prediction using the nearest neighbour energy model.	yes	sourcecode	[6]
PknotsRG	A dynamic programming algorithm for the prediction of a restricted class of RNA pseudoknots.	yes	sourcecode, webserver	[7]
RNAfold	MFE RNA structure prediction algorithm. Includes an implementation of the partition function for computing basepair probabilities and circular RNA folding.	no	sourcecode, webserver	[8] [5][9] [10][11][12]
Sfold	Statistical sampling of all possible structures. The sampling is weighted by partition function probabilities.	no	webserver	[13][14][15][16]
UNAFold	The UNAFold software package is an integrated collection of programs that simulate folding, hybridization, and melting pathways for one or two single-stranded nucleic acid sequences.	no	sourcecode	[17]

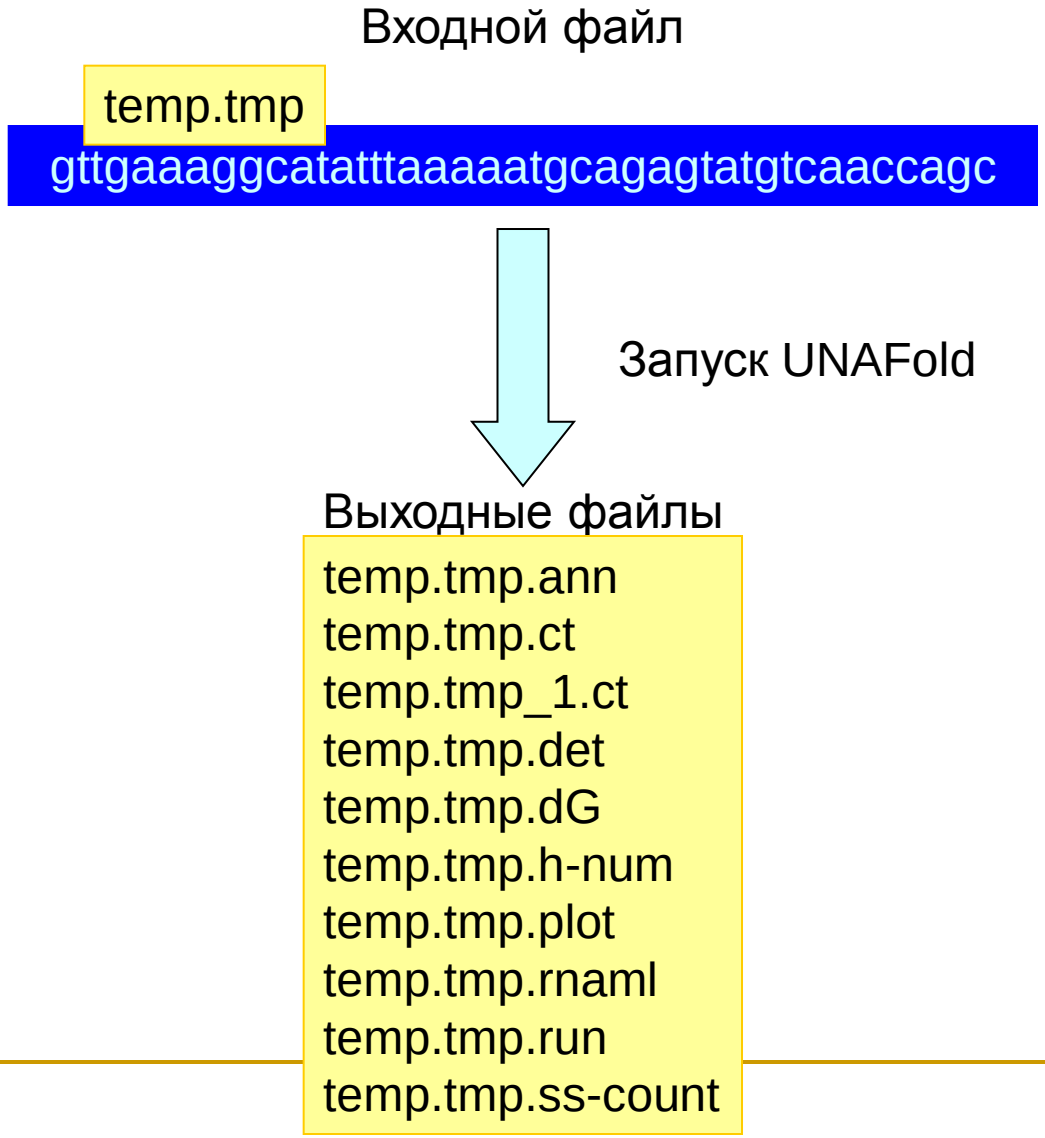
*Knots: Pseudoknot prediction, <yes|no>.

Пример. Скачивание и установка программы UNAFold

Format	Operating System	Architecture	Type	File
Binary RPM	Linux	x86 (32-bit)	Binary	unafold-3.6-1.i386.rpm (506 KB)
Binary RPM	Linux	x86 (64-bit)	Binary	unafold-3.6-1.x86_64.rpm (610 KB)
Source RPM	Linux	(Any)	Source	unafold-3.6-1.src.rpm (345 KB)
EXE	Windows	x86	Binary	UNAFold-3.6.exe (2.5 MB)
tar/bzip2	(Any)	(Any)	Source	unafold-3.6.tar.bz2 (271 KB)
tar/gzip	(Any)	(Any)	Source	unafold-3.6.tar.gz (345 KB)

http://www.bioinfo.rpi.edu/applications/hybrid/download.php

Запуск программы UNAFold



Выходные файлы UNAFold

temp.tmp.run

```
hybrid-ss-min 3.4 ran on temp.tmp at
Thu Oct 23 17:32:53 2008

suffix = DAT
maxloop = 30
prefilter 2/2
mfold mode: P=5, W=-1, MAX=100
```

temp.tmp.plot

level	length	i	j	energy
1	4	1	41	-91
1	9	8	34	-91

temp.tmp.h-num

level	length	istart	jstart	h-num
1	4	1	41	2
1	9	8	34	2

temp.tmp.dG

#T	-RT ln Z	Z
37	-9.1	2.58391e+06

Выходные файлы UNAFold

temp.tmp.ann

1	1
2	1
3	1
4	1
5	0
6	0
7	0
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1
16	1
17	0
18	0
19	0
20	0
21	0
22	0
23	0
24	0
25	0
26	1
27	1
28	1
29	1
30	1
31	1
32	1
33	1
34	1
35	0
36	0
37	0
38	1
39	1
40	1
41	1

temp.tmp.ann

1	0	g
2	0	t
3	0	t
4	0	g
5	1	a
6	1	a
7	1	a
8	0	g
9	0	g
10	0	c
11	0	a
12	0	t
13	0	a
14	0	t
15	0	t
16	0	t
17	1	a
18	1	a
19	1	a
20	1	a
21	1	a
22	1	t
23	1	g
24	1	c
25	1	a
26	0	g
27	0	a
28	0	g
29	0	t
30	0	a
31	0	t
32	0	g
33	0	t
34	0	c
35	1	a
36	1	a
37	1	c
38	0	c
39	0	a
40	0	g
41	0	c

Выходные файлы UNAFold

temp.tmp.ct

41	dG = -9.1	temp.tmp					
1	g	0	2	41	1	0	2
2	t	1	3	40	2	1	3
3	t	2	4	39	3	2	4
4	g	3	5	38	4	3	5
5	a	4	6	0	5	4	0
6	a	5	7	0	6	0	0
7	a	6	8	0	7	0	8
8	g	7	9	34	8	7	9
9	g	8	10	33	9	8	10
10	c	9	11	32	10	9	11
11	a	10	12	31	11	10	12
12	t	11	13	30	12	11	13
13	a	12	14	29	13	12	14
14	t	13	15	28	14	13	15
15	t	14	16	27	15	14	16
16	t	15	17	26	16	15	0
17	a	16	18	0	17	0	0
18	a	17	19	0	18	0	0
19	a	18	20	0	19	0	0
20	a	19	21	0	20	0	0
21	a	20	22	0	21	0	0
22	t	21	23	0	22	0	0
23	g	22	24	0	23	0	0
24	c	23	25	0	24	0	0
25	a	24	26	0	25	0	0
26	g	25	27	16	26	0	27
27	a	26	28	15	27	26	28
28	g	27	29	14	28	27	29
29	t	28	30	13	29	28	30
30	a	29	31	12	30	29	31
31	t	30	32	11	31	30	32
32	g	31	33	10	32	31	33
33	t	32	34	9	33	32	34
34	c	33	35	8	34	33	35
35	a	34	36	0	35	34	0
36	a	35	37	0	36	0	0
37	c	36	38	0	37	0	38
38	c	37	39	4	38	37	39
39	a	38	40	3	39	38	40
40	g	39	41	2	40	39	41
41	c	40	0	1	41	40	0

Выходные файлы UNAFold

temp.tmp.det

```
Structure 1: dG = -9.10

External loop:   ddG =  +0.00 0 ss bases & 1 closing helices
Stack:           ddG =  -2.50 External closing pair is g(     1)-c(    41)
Stack:           ddG =  -0.60 External closing pair is t(     2)-g(    40)
Stack:           ddG =  -2.10 External closing pair is t(     3)-a(    39)
Helix:           ddG =  -5.20 4 base pairs.
Interior loop:   ddG =  +2.00 External closing pair is g(     4)-c(    38)
Stack:           ddG =  -1.50 External closing pair is g(     8)-c(    34)
Stack:           ddG =  -2.50 External closing pair is g(     9)-t(    33)
Stack:           ddG =  -2.10 External closing pair is c(    10)-g(    32)
Stack:           ddG =  -1.10 External closing pair is a(    11)-t(    31)
Stack:           ddG =  -1.30 External closing pair is t(    12)-a(    30)
Stack:           ddG =  -1.40 External closing pair is a(    13)-t(    29)
Stack:           ddG =  -0.60 External closing pair is t(    14)-g(    28)
Stack:           ddG =  -1.30 External closing pair is t(    15)-a(    27)
Helix:           ddG = -11.80 9 base pairs.
Hairpin loop:    ddG =  +5.90 Closing pair is t(    16)-g(    26)
```

Выходные файлы UNAFold

temp.tmp.rnaml

```
<?xml version="1.0"?>
<!DOCTYPE rnaml PUBLIC "-//University of Montreal//RNAML 1.1//EN" "http://www-lbit.iro.umontreal.ca/

<rnaml version="1.1" comment="Output from UNAFold">

  <analysis id="UNAFold">
    <program>
      <prog-name>UNAFold</prog-name>
      <prog-version>3.4</prog-version>
    </program>
  </analysis>


  <!-- 1st sequence and its associated foldings ******************** -->
  <molecule id="temp.tmp" type="dna">
    <sequence circular="false" strand="1">
      <seq-data>
        gttgaaaggc atatttaaaa atgcagagta tgtcaaccag c
      </seq-data>
    </sequence>

    <structure analysis-ids="UNAFold">
      <!-- Start of folding 1 for sequence 1 ************************* -->
      <model id="_1.1">
        <model-info>
          <free-energy>-9.10</free-energy>
        </model-info>

        <str-annotation>
          <helix>
            <base-id-5p>
              <base-id>
                <position>1</position>
              </base-id>
            </base-id-5p>
            <base-id-3p>
              <base-id>
                <position>41</position>
              </base-id>
            </base-id-3p>
            <length>4</length>
          </helix>
          <helix>
            <base-id-5p>
              <base-id>
                <position>8</position>
              </base-id>
            </base-id-5p>
```

Структура файла .ct программы UNAFold

Строка 1

Строки 2..42

Число нуклеотидов

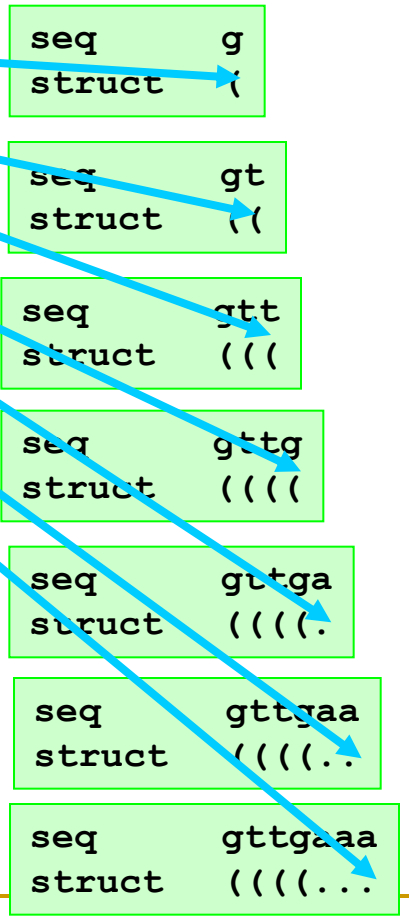
Энергия структуры

Файл исходной посл.

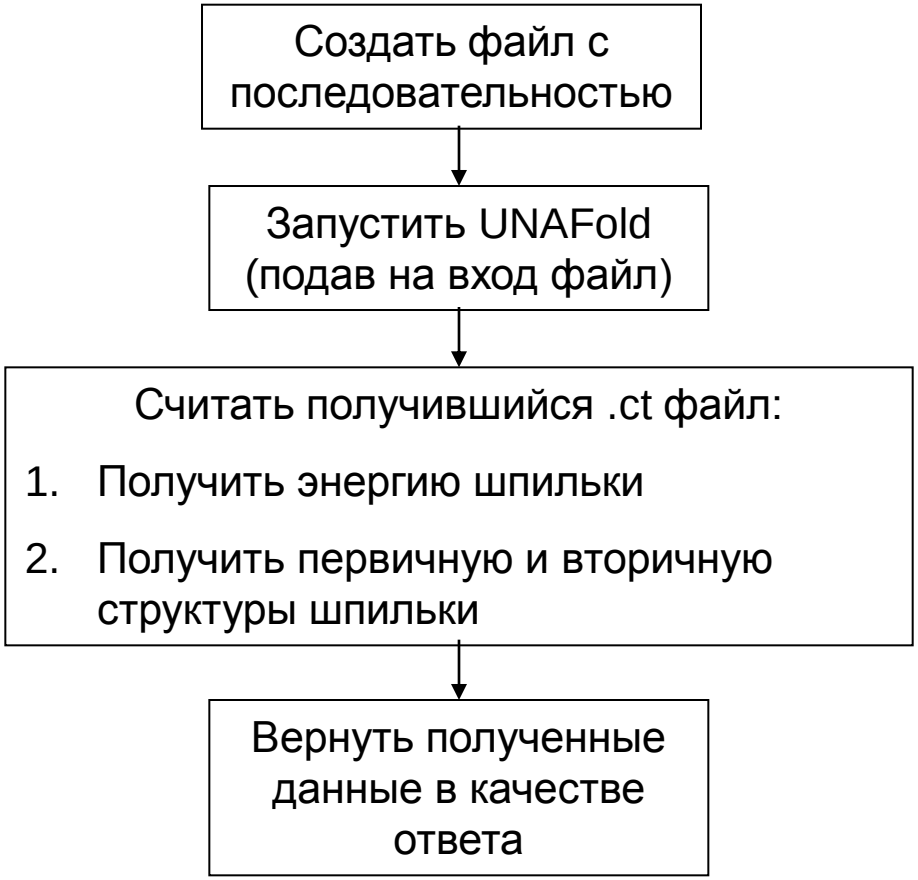
41	**dG = -9.1 temp.tmp**						
1	**g**	**0**	**2**	**41**	**1**	**0**	**2**
2	**t**	**1**	**3**	**40**	**2**	**1**	**3**
3	**t**	**2**	**4**	**39**	**3**	**2**	**4**
4	**g**	**3**	**5**	**38**	**4**	**3**	**5**
5	**a**	**4**	**6**	**0**	**5**	**4**	**0**
6	**a**	**5**	**7**	**0**	**6**	**0**	**0**
7	**a**	**6**	**8**	**0**	**7**	**0**	**8**
8	**g**	**7**	**9**	**34**	**8**	**7**	**9**
…							
34	**c**	**33**	**35**	**8**	**34**	**33**	**35**
35	**a**	**34**	**36**	**0**	**35**	**34**	**0**
36	**a**	**35**	**37**	**0**	**36**	**0**	**0**
37	**c**	**36**	**38**	**0**	**37**	**0**	**38**
38	**c**	**37**	**39**	**4**	**38**	**37**	**39**
39	**a**	**38**	**40**	**3**	**39**	**38**	**40**
40	**g**	**39**	**41**	**2**	**40**	**39**	**41**
41	**c**	**40**	**0**	**1**	**41**	**40**	**0**

Построение вторичной структуры в скобочном виде по .ct файлу

41	dG = -9.1 temp.tmp		
1	g	…	41
2	t	…	40
3	t	…	39
4	g	…	38
5	a	…	0
6	a	…	0
7	a	…	0
8	g	…	34
…			
34	c	…	8
35	a	…	0
36	a	…	0
37	c	…	0
38	c	…	4
39	a	…	3
40	g	…	2
41	c	…	1



Блок-схема алгоритма использования программы UNAFold в своих программах



Примерная организация программы для использования UNAFold

Основная программа

String s1….

// Требуется посчитать
//структуру и энергию s1

RNACalculator::calculate(s1)
//возвращает список RNAInfo

Класс RNACalculator (РНК калькулятор)

Метод calculate(String s)
Подготовить файл
Запустить UNAFold
Прочитать вывод
Возвратить список RNAInfo

Сервис-класс RNAInfo

Первичная структура
Вторичная структура
Энергия Гиббса

Организация программы

- Наиболее часто используемым механизмом включения сторонних разработок в собственный проект является механизм *библиотек*, или, в терминах Java – *пакетов*.

- Механизм пакетов позволяет организовывать Java классы в *пространства имён* (аналогично модулям в других языках программирования).

Основные пакеты стандартной версии Java SE

Имя пакета	Краткое описание функциональности
java.lang	Базовая функциональность языка и основные типы данных (классы)
java.util	Классы коллекций и вспомогательных структур данных
java.io	Операции ввода-вывода, в том числе файлового
java.math	Математические операции высокой точности
java.nio	Расширенная (новая – New I/O) функциональность ввода-вывода
java.net	Операции и сетью, сокеты, протоколы и др.
java.security	Безопасность: генерация ключей, шифрование/дешифрование
java.sql	Работа с базами данных
java.awt	Иерархия пакетов для построения графического интерфейса (базовые компоненты)
javax.swing	Иерархия пакетов для построения платформо-независимого графического интерфейса (расширенные компоненты)

Использование пакетов

- Для использования класса из пакетов, необходимо подключить этот класс с помощью команды `import`:

```
import java.math.BigInteger; // Импорт
класса поддержки больших чисел
```

- Для подключения всех классов пакета используется символ звёздочки (*):

```
import java.math.*; // Импорт всех классов
пакета java.math
```

Использование пакетов

- Для создания своего собственного пакета необходимо использовать ключевое слово `package имя.пакета` в начале файла, в котором содержится класс:

```
package testpackage;
public class TestClass1{
// Тело класса 1…
}
package testpackage;
public class TestClass2{
// Тело класса 2…
}
```

Использование пакетов

- Для именования пакетов используется соглашение, практически исключающее дублирование имён при выкладывании пакета в общий доступ и последующем его использовании совместно с пакетами других производителей.
- Идея заключается в использовании в наименовании пакета уникального доменного имени, характерного для разработчика(-ков) данного пакета.
- Например, если некий пакет **`BioPackage`** разрабатывает в Лаборатории теоретической генетики Института цитологии и генетики СО РАН, которая имеет интернет-адрес: http://bionet.nsc.ru/theorylab, то подходящим полным названием для такого пакета будет:
- **`package ru.nsc.bionet.theorylab.BioPackage;`**
- Таким образом, практически наверняка исключается конфликт имён с пакетами **`BioPackage`** других производителей

Лекция 8
Форматы представления биологических данных

Разделы геномики

- Структурная геномика – изучение организации и содержания информации в геноме
- Функциональная геномика – изучение фенотипической реализации информации в геноме
- Сравнительная (эволюционная) геномика – сравнительное исследование геномов разных организмов

Геномика: компьютерный анализ генетических текстов (алфавит A, T, G, C)

1) предсказание кодирующих участков генов и открытых рамок считывания;
2) предсказание функциональных сигналов (функциональных сайтов и регуляторных районов);
3) статистический анализ генетических текстов, исследование структуры повторов и модели порождения символьных последовательностей, сегментация геномов;
4) поиск гомологии и выравнивание генетических текстов, множественное выравнивание
5) анализ вторичной структуры РНК и сигналов трансляции;

Форматы представления биологических данных: FASTA

- FASTA формат используется для представления нуклеотидных или аминокислотных последовательностей в обычном текстовом виде (plain text)
- Файл содержит одну или несколько последовательностей
- Последовательность представляется в строках, длина каждой их которых не должна превышать 120 символов (обычно не превышают 80 символов)
- Первым символом в FASTA-файле должен быть либо символ «больше» (>), либо символ «точка с запятой» (;) с последующим далее кратким комментарием
- После чего следуют непосредственно строки, содержащие последовательность. Каждой новой последовательности предшествует строка с комментарием, также начинающаяся с символов >

Форматы представления биологических данных: FASTA

Пример файла в FASTA-формате: последовательность белка цитохром-б индийского слона (*Elephas maximus*).

```
>gi|5524211|gb|AAD44166.1| cytochrome b [Elephas maximus maximus]
LCLYTHIGRNIYYGSYLYSETWNTGIMLLLITMATAFMGYVLPWGQMSFWGATVITNLFSAIPYIGTNLV
EWIWGGFSVDKATLNRFFAFHFILPFTMVALAGVHLTFLHETGSNNPLGLTSDSDKIPFHPYYTIKDFLG
LLILILLLLLLALLSPDMLGDPDNHMPADPLNTPLHIKPEWYFLFAYAILRSVPNKLGGVLALFLSIVIL
GLMPFLHTSKHRSMMLRPLSQALFWTLTMDLLTLTWIGSQPVEYPYTIIGQMASILYFSIILAFLPIAGX
IENY
```

Форматы представления биологических данных: FASTA

База данных	Идентификатор
GenBank	gi|gi-номер|gb|образец|локус
EMBL, Data Library	gi|gi-номер|emb|образец|локус
DDBJ, DNA Database of Japan	gi|gi-номер|dbj|образец|локус
NBRF PIR	pir||запись
Protein Research Foundation	prf||имя
SWISS-PROT	sp|образец|имя
Brookhaven Protein Data Bank (1)	pdb|запись|цепь
Brookhaven Protein Data Bank (2)	запись:цепь|PDBID|CHAIN|SEQUENCE
Patents	pat|страна|номер
GenInfo Backbone Id	bbs|номер
Общий идентификатор базы данных	gnl|база даных|идентификатор
NCBI Reference Sequence	ref|образец|локус
Local Sequence identifier	lcl|идентификатор

Формат уникальных идентификаторов и соответствующих баз данных биологических последовательностей

Форматы представления биологических данных: FASTA

Принятые расширения для FASTA-файлов

Расширение	Содержание	Примечание
fasta, fa, seq, fsa	общий FASTA	Любой FASTA-файл
fna	нуклеиновые кислоты	Для кодирующих районов конкретных геномов следует использовать расширение ffn, данное расширение полезно для остальных типов нуклеиновых кислот
ffn	нуклеотидные кодирующие районы FASTA	Содержит кодирующие участки геномов
faa	аминокислоты	Содержит аминокислотную последовательность. Для файлов с несколькими последовательностями также используется расширение mpfa
frn	некодирующие РНК	Содержит некодирующие участки генома (например, тРНК, рРНК)

Разметка геномов

- Цель – выявление всех структурно-функциональных единиц в геноме: генов (белков, тРНК, рРНК), оперонов и регуляторных элементов
  - Выявление и анализ промоторных областей
  - Выявление и анализ терминаторных областей
  - Выявление функций генов

Разметка геномов. Genbank

- GenBank – база данных, содержащая последовательности ДНК, разработанная NCBI (Национальным центром биотехнологической информации США)
- В настоящее время формат Genbank является стандартом де факто для хранения информации о геномах
- http://www.ncbi.nlm.nih.gov/genbank/

Разметка геномов. Формат Genbank

LOCUS NC_009782 2880168 bp DNA circular BCT 03-MAY-2010
DEFINITION Staphylococcus aureus subsp. aureus Mu3, complete genome.
ACCESSION NC_009782
VERSION NC_009782.1 GI:156978331
DBLINK Project:18509
KEYWORDS .
SOURCE Staphylococcus aureus subsp. aureus Mu3
ORGANISM Staphylococcus aureus subsp. aureus Mu3
Bacteria; Firmicutes; Bacillales; Staphylococcus.
REFERENCE 1 (bases 1 to 2880168)
AUTHORS Neoh,H.M., Cui,L., Yuzawa,H., Takeuchi,F., Matsuo,M. and Hiramatsu,K.
TITLE Mutated response regulator graR is responsible for phenotypic conversion of Staphylococcus aureus from heterogeneous vancomycin-intermediate resistance to vancomycin-intermediate resistance
JOURNAL Antimicrob. Agents Chemother. 52 (1), 45-53 (2008)
PUBMED 17954695

Разметка геномов. Формат Genbank

```
source          1..2880168
                /organism="Staphylococcus aureus subsp. aureus Mu3"
                /mol_type="genomic DNA"
                /strain="Mu3"
                /sub_species="aureus"
                /db_xref="taxon:418127"
  gene          517..1878
                /gene="dnaA"
                /locus_tag="SAHV_0001"
                /db_xref="GeneID:5559096"
    CDS         517..1878
                /gene="dnaA"
                /locus_tag="SAHV_0001"
                /note="binds to the dnaA-box as an ATP-bound complex at
                the origin of replication during the initiation of
                chromosomal replication; can also affect transcription of
                multiple genes including itself."
                /codon_start=1
                /transl_table=11
                /product="chromosomal replication initiation protein"
                /protein_id="YP_001440591.1"
                /db_xref="GI:156978332"
                /db_xref="GeneID:5559096"
```

Координаты гена в хромосоме

Название гена

CDS – coding DNA
sequence – кодирующаяя ДНК

Разметка геномов. Формат Genbank

```
gene            14011..14087
                /locus_tag="SACE_8001"
                /note="tRNA-Ile(GAT)"
                /db_xref="GeneID:4939872"
     tRNA          14011..14087
                /locus_tag="SACE_8001"
                /product="tRNA-Asp"
                /codon_recognized="GAU"
                /db_xref="GeneID:4939872"
```

Не все гены являются белок-кодирующими. Нетранслируемые гены (tRNA, rRNA) не имеют полей CDS и /translation

В конце файла Genbank приводится полная нуклеотидная последовательность

```
ORIGIN
        1 gaattccggc ctgctgccgg gccgcccgac ccgccgggcc acacggcaga gccgcctgaa
       61 gcccagcgct gaggctgcac ttttccgagg gcttgacatc agggtctatg tttaagtctt
      121 agctcttgct tacaaagacc acggcaattc cttctctgaa gccctcgcag ccccacagcg
      181 ccctcgcagc cccagcctgc
```

Разметка геномов. Формат EMBL

```
ID   SC10H5 standard; DNA; PRO; 4870 BP.
XX
AC   AL031232;
XX
DE   Streptomyces coelicolor cosmid 10H5.
XX
KW   integral membrane protein.
XX
OS   Streptomyces coelicolor
OC   Eubacteria; Firmicutes; Actinomycetes; Streptomycetes;
OC   Streptomycetaceae; Streptomyces.
XX
RN   [1]
RP   1-4870
RA   Oliver K., Harris D.;
RT   ;
RL   Unpublished.
XX
RN   [2]
RP   1-4870
RA   Parkhill J., Barrell B.G., Rajandream M.A.;
RT   ;
RL   Submitted (10-AUG-1998) to the EMBL/GenBank/DDBJ databases.
RL   Streptomyces coelicolor sequencing project,
RL   Sanger Centre, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SA
RL   E-mail: barrell@sanger.ac.uk
RL   Cosmids supplied by Prof. David A. Hopwood, [3]
RL   John Innes Centre, Norwich Research Park, Colney,
RL   Norwich, Norfolk NR4 7UH, UK.
```

Детальное описание : http://www.ebi.ac.uk/embl/Documentation/User_manual/usrman.html

Разметка геномов. Формат EMBL

```
CC   Notes:
CC
CC   Streptomyces coelicolor sequencing at The Sanger Centre is funded
CC   by the BBSRC.
XX
FH   Key             Location/Qualifiers
FH
FT   source          1..4870
FT                   /organism="Streptomyces coelicolor"
FT                   /strain="A3(2)"
FT                   /clone="cosmid 10H5"
FT   CDS             complement(<1..327)
FT                   /note="SC10H5.01c, unknown, partial CDS, len >109 aa;
FT                   possible integral membrane protein"
FT                   /gene="SC10H5.01c"
FT                   /product="hypothetical protein SC10H5.01c"
FT   CDS             complement(350..805)
FT                   /note="SC10H5.02c, probable integral membrane protein, len:
FT                   151 aa; similar to S. coelicolor hypothetical protein
FT                   TR:O54194 (EMBL:AL021411) SC7H1.35 (155 aa), fasta scores;
FT                   opt: 431 z-score: 749.8 E(): 0, 53.5% identity in 114 aa
FT                   overlap."
FT                   /product="putative integral membrane protein«
FT   misc_feature    4769..4870
FT                   /note="overlap with cosmid 3A7 from 1 to 102"
XX
SQ   Sequence 4870 BP; 769 A; 1717 C; 1693 G; 691 T; 0 other;
     gatcagtaga cccagcgaca gcagggcggg gcccagcagg ccggccgtgg cgtagagcgc        60
     gaggacggcg accggcgtgg ccaccgacag gatggctgcg gcgacgcgga cgacaccgga       120
     gtgtgccagg gcccaccaca cgccgatggc cgcgagcgcg agtcccgcgc tgccgaacag       180
     ggcccacagc acactgcgca gaccggcggc cacgagtggc gccaggacgg tgcccagcag       240
```

Детальное описание : http://www.ebi.ac.uk/embl/Documentation/User_manual/usrman.html

Разметка геномов. Формат EMBL

- Идентификаторы строк:

Идентификатор	Значение
ID	Идентификатор (вкл. таксономическую классификацию организма)ч
AC	Идентификатор последовательности
PR	Идентификатор проекта
DT	Дата
DE	Описание
KW	Ключевые слова
OS	Вид организма
OC	Классификация организма
OG	Органелла

Разметка геномов. Формат EMBL
Таксономическая классификация организмов

Код классификатора	Таксономическая классификация организма
PHG (Bacteriophage)	Бактериофаги
ENV (Environmental Sample)	Природные сообщества (метагеномы)
FUN (Fungal)	Грибы
HUM (Human)	Человек
INV (Invertebrate)	Беспозвоночные
MAM (Mammal)	Млекопитающие (искл. человека)
VRT (Vertebrate)	Позвоночные (искл. млекопитающих)
MUS (Mus musculus)	Мышь
PLN (Plant)	Растения
PRO (Prokaryote)	Прокариоты
ROD (Other Rodent)	Грызуны (искл. мышь)
SYN (Synthetic)	Синтезированные последовательности
TGN (Transgenic)	Трансгены
UNC (Unclassified)	Неклассифицированные последовательности
Viral VRL	Вирусы

Разметка геномов. Формат EMBL

- Идентификаторы строк:

Идентификатор	Значение
RN (reference number)	Справочный (референсный) номер
RC (reference comment)	Справочный (референсный) комментарий
RP (reference positions)	Справочные (референсные) позиции
RX (reference cross-reference)	Перекрёстная ссылка
RG (reference group)	Контрольная (референсная) группа
RA (reference author(s))	Авторы
RT (reference title)	Ссылка на публикацию
RL (reference location)	Выходные данные публикации

Разметка геномов. Формат EMBL

- Идентификаторы строк:

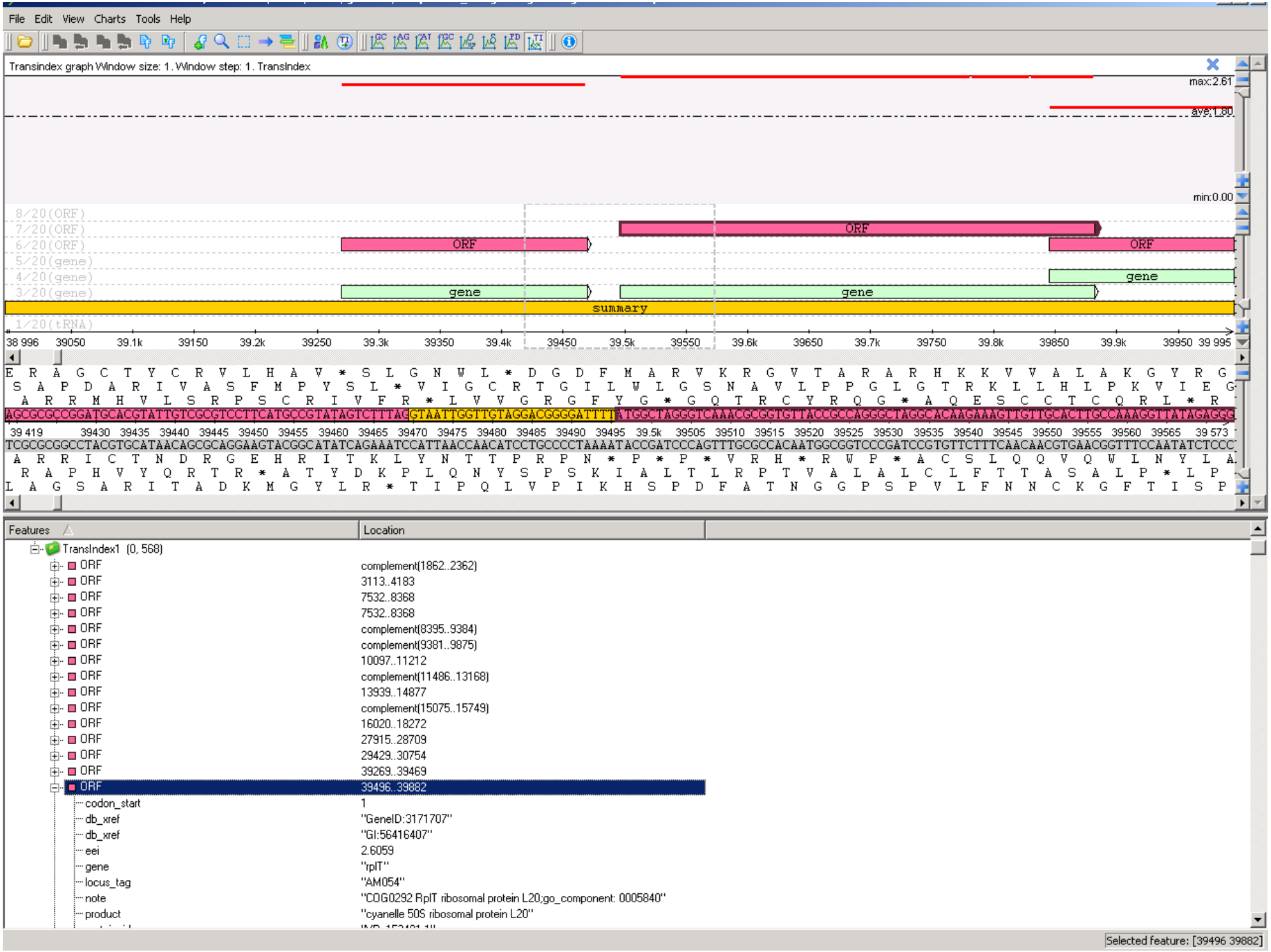
Идентификатор	Значение
DR (database cross-reference)	Ссылка в базе данных
AH (assembly header)	Начало блока описания сборки последовательности
AS (assembly information)	Информация о сборке последовательности
CO (contig/construct)	Контиг/векторная конструкция (ДНК)
FH (feature table header)	Начало описания свойств
FT (feature table data)	Описание свойства
SQ (sequence header)	Начало последовательности
CC	Комментарии
//	Конец записи

Разметка геномов. Другие форматы

- BSML (Bioinformatic Sequence Markup Language) - в формате XML (http://xml.coverpages.org/bsml.html)
- EntrezGene – в формате ASN1 (http://www.bioperl.org/wiki/EntrezGene_sequence_format, http://jura.wi.mit.edu/entrez_gene/)
- GFF (ранние версии - Gene Finding Format, нынешние - Generic Feature Format) – простой текстовый (plain-text) формат
  - GFF2 (http://www.sanger.ac.uk/resources/software/gff/spec.html)
  - GTF или GFF2.5 (http://mblab.wustl.edu/GTF2.html)
  - GFF3 (http://www.sequenceontology.org/gff3.shtml) наиболее свежая версия
- BED (http://genome.ucsc.edu/FAQ/FAQformat.html#format1)
- Многие другие форматы…
  - http://genome.ucsc.edu/FAQ/FAQformat.html
  - http://www.bioperl.org/wiki/Category:Formats

Работа с размеченными геномами

- Геномные браузеры (Genome Browsers) – программы для визуализации и работы с геномами.
- Зачастую содержат средства геномного анализа и первичной разметки геномов (идеология pipe-line)
- Примеры геномных браузеров:
  - Unipro UGENE (http://genome.unipro.ru/)
  - Softberry Molquest (http://www.molquest.com/)


File Edit View Charts Tools Help
Transindex graph Window size: 1. Window step: 1. TransIndex
max:2.61
ave:1.80
min:0.00
8/20(ORF)
7/20(ORF)
6/20(ORF)
5/20(gene)
4/20(gene)
3/20(gene)
1/20(tRNA)
ORF
ORF
ORF
gene
gene
gene
summary
38 996 39050 39.1k 39150 39.2k 39250 39.3k 39350 39.4k 39450 39.5k 39550 39.6k 39650 39.7k 39750 39.8k 39850 39.9k 39950 39 995
39 419 39430 39435 39440 39445 39450 39455 39460 39465 39470 39475 39480 39485 39490 39495 39.5k 39505 39510 39515 39520 39525 39530 39535 39540 39545 39550 39555 39560 39565 39 573
Features
Location
TransIndex1 (0, 568)
ORF complement(1862..2362)
ORF 3113..4183
ORF 7532..8368
ORF 7532..8368
ORF complement(8395..9384)
ORF complement(9381..9875)
ORF 10097..11212
ORF complement(11486..13168)
ORF 13939..14877
ORF complement(15075..15749)
ORF 16020..18272
ORF 27915..28709
ORF 29429..30754
ORF 39269..39469
ORF 39496..39882
codon_start 1
db_xref "GeneID:3171707"
db_xref "GI:56416407"
eei 2.6059
gene "rplT"
locus_tag "AM054"
note "COG0292 RplT ribosomal protein L20;go_component: 0005840"
product "cyanelle 50S ribosomal protein L20"
Selected feature: [39496 39882]

Лекция 9
Программные библиотеки для работы с геномными данными

Программные библиотеки (API) для работы в области биоинформатики

- API (application programming interface) – интерфейс программирования приложений, представляющий набор готовых
  - Классов
  - Процедур
  - Функций
  - Структур
  - Констант

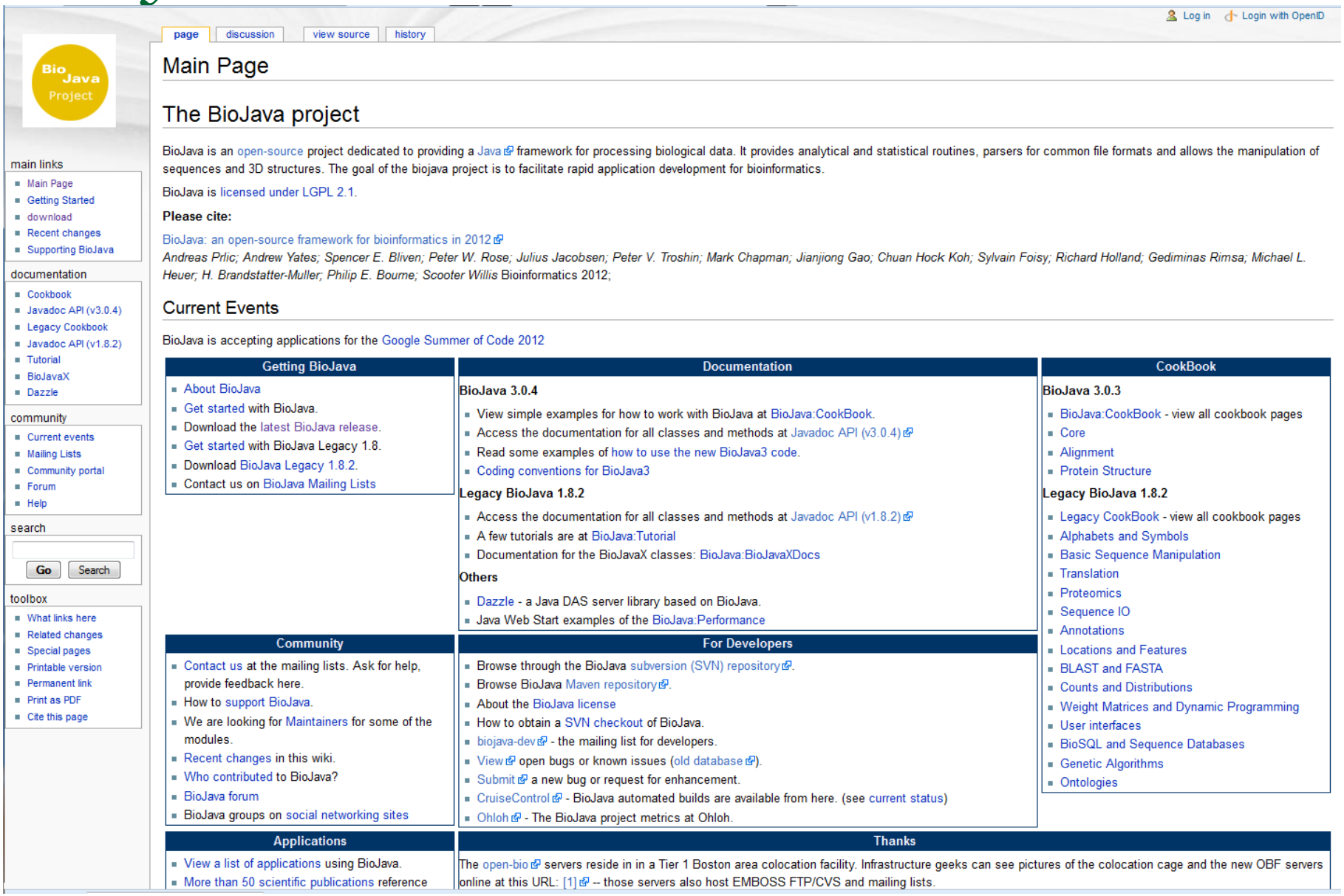
Программные библиотеки (API) для работы в области биоинформатики

- BioJava (http://biojava.org)
- BioPerl (http://www.bioperl.org)
- BioPython (http://biopython.org/)
- BioRuby (http://bioruby.open-bio.org/)
- Bioclipse (http://www.bioclipse.net/)
- Bioconductor (http://www.bioconductor.org/)
- EMBOSS (http://emboss.sourceforge.net/)
- Gaggle (http://gaggle.systemsbiology.net/docs/)
- Больше библиотек:
  - http://en.wikipedia.org/wiki/List_of_Open_Source_Bioinformatics_software

BioJava

- BioJava (www.biojava.org) представляет собой проект с открытым исходным кодом (open-source) и предоставляет платформу для обработки биологических данных, включающую в себя
  - объекты для работы с биологическими последовательностями,
  - разборщики файлов, поддержку распределённой системы аннотации (Distributed Annotation System – DAS),
  - доступ к базам данных BioSQL и Ensembl,
  - средства для визуального и статистического анализа последовательностей
  - и многие другие средства.

BioJava



Log in Login with OpenID

page | discussion | view source | history

Main Page

The BioJava project

BioJava is an open-source project dedicated to providing a Java framework for processing biological data. It provides analytical and statistical routines, parsers for common file formats and allows the manipulation of sequences and 3D structures. The goal of the biojava project is to facilitate rapid application development for bioinformatics.

BioJava is licensed under LGPL 2.1.

Please cite:

BioJava: an open-source framework for bioinformatics in 2012
Andreas Prlic; Andrew Yates; Spencer E. Bliven; Peter W. Rose; Julius Jacobsen; Peter V. Troshin; Mark Chapman; Jianjiong Gao; Chuan Hock Koh; Sylvain Foisy; Richard Holland; Gediminas Rimsa; Michael L. Heuer; H. Brandstatter-Muller; Philip E. Bourne; Scooter Willis Bioinformatics 2012;

Current Events

BioJava is accepting applications for the Google Summer of Code 2012

Getting BioJava

- About BioJava
- Get started with BioJava.
- Download the latest BioJava release.
- Get started with BioJava Legacy 1.8.
- Download BioJava Legacy 1.8.2.
- Contact us on BioJava Mailing Lists

Documentation

BioJava 3.0.4

- View simple examples for how to work with BioJava at BioJava:CookBook.
- Access the documentation for all classes and methods at Javadoc API (v3.0.4)
- Read some examples of how to use the new BioJava3 code.
- Coding conventions for BioJava3

Legacy BioJava 1.8.2

- Access the documentation for all classes and methods at Javadoc API (v1.8.2)
- A few tutorials are at BioJava:Tutorial
- Documentation for the BioJavaX classes: BioJava:BioJavaXDocs

Others

- Dazzle - a Java DAS server library based on BioJava.
- Java Web Start examples of the BioJava:Performance

CookBook

BioJava 3.0.3

- BioJava:CookBook - view all cookbook pages
- Core
- Alignment
- Protein Structure

Legacy BioJava 1.8.2

- Legacy CookBook - view all cookbook pages
- Alphabets and Symbols
- Basic Sequence Manipulation
- Translation
- Proteomics
- Sequence IO
- Annotations
- Locations and Features
- BLAST and FASTA
- Counts and Distributions
- Weight Matrices and Dynamic Programming
- User interfaces
- BioSQL and Sequence Databases
- Genetic Algorithms
- Ontologies

Community

- Contact us at the mailing lists. Ask for help, provide feedback here.
- How to support BioJava.
- We are looking for Maintainers for some of the modules.
- Recent changes in this wiki.
- Who contributed to BioJava?
- BioJava forum
- BioJava groups on social networking sites

For Developers

- Browse through the BioJava subversion (SVN) repository.
- Browse BioJava Maven repository.
- About the BioJava license
- How to obtain a SVN checkout of BioJava.
- biojava-dev - the mailing list for developers.
- View open bugs or known issues (old database).
- Submit a new bug or request for enhancement.
- CruiseControl - BioJava automated builds are available from here. (see current status)
- Ohloh - The BioJava project metrics at Ohloh.

Applications

- View a list of applications using BioJava.
- More than 50 scientific publications reference

Thanks

The open-bio servers reside in in a Tier 1 Boston area colocation facility. Infrastructure geeks can see pictures of the colocation cage and the new OBF servers online at this URL: [1] -- those servers also host EMBOSS FTP/CVS and mailing lists.

main links
- Main Page
- Getting Started
- download
- Recent changes
- Supporting BioJava

documentation
- Cookbook
- Javadoc API (v3.0.4)
- Legacy Cookbook
- Javadoc API (v1.8.2)
- Tutorial
- BioJavaX
- Dazzle

community
- Current events
- Mailing Lists
- Community portal
- Forum
- Help

search
Go Search

toolbox
- What links here
- Related changes
- Special pages
- Printable version
- Permanent link
- Print as PDF
- Cite this page

BioJava. Основные пакеты

Группа пакетов	Функциональность пакета
Core biological processes (Базовые биологические процессы)	Основные классы для работы с последовательностями (выравнивание, сравнение, использование различные алфавитов, алгоритмы вероятностного анализа биологических последовательностей и др.)
Sequence databases and formats (поддержка баз данных и форматов последовательностей)	Драйверы для работы с различными базами данных
User interface components (компоненты графического интерфейса)	Графические классы для работы с объектами данных biojava (визуализация последовательностей и разметок)
Dynamic programming packages (пакеты динамического программирования)	Классы для работы с марковскими моделями и алгоритмами динамического программирования
Chromatogram and trace file support (поддержка хроматограмм и файлов трассировки	Классы для работы с файлами хроматограмм, генерируемых секвенирующим оборудованием; средства для визуализации хроматограмм

BioJava. Основные пакеты

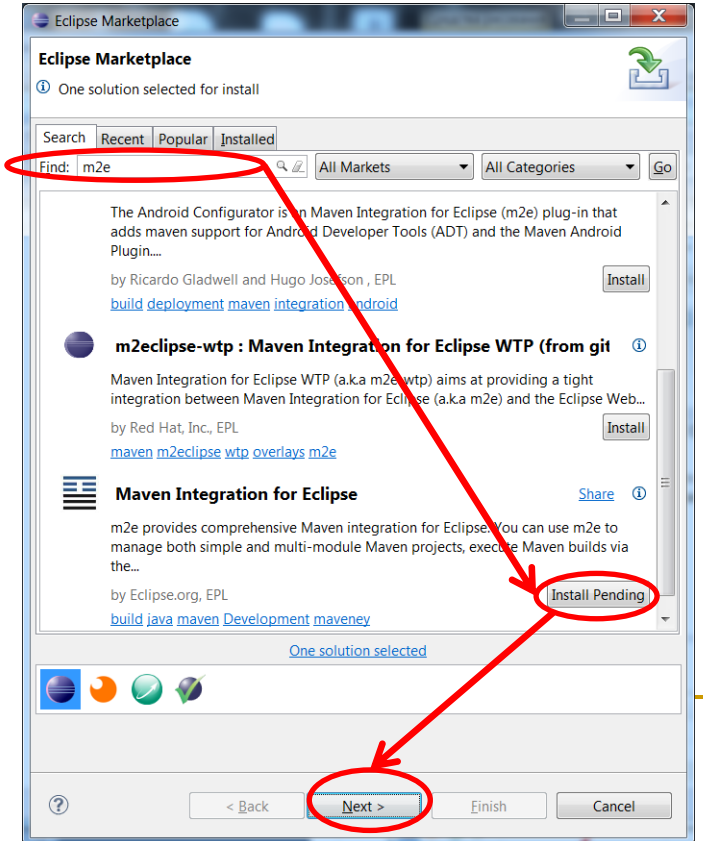
Группа пакетов	Функциональность пакета
Macromolecular structure (макромолекулярные структуры)	Классы для работы с белками и белковыми структурами (PDB), алгоритмы структурного выравнивания
Utilities and developers' packages (утилиты и пакеты для разработки)	Классы для работы с файлами, XML-форматами, онтологиями, регулярными выражениями, математические утилиты и работа с сетью
Molecular biology packages (пакеты молекулярной биологии)	Классы для поддержки экспериментальных методов молекулярной биологии, таких как рестрикционное картирование и ПЦР
Experimental packages (Экспериментальные пакеты)	Экспериментальные классы, расширяющие функциональность за счёт новых технологий
Biojava extension (biojavax) packages (пакеты расширения biojava)	Пакеты расширения для Biojava, а также классы для связи biojava объектов с базами данных biosql и данными NCBI

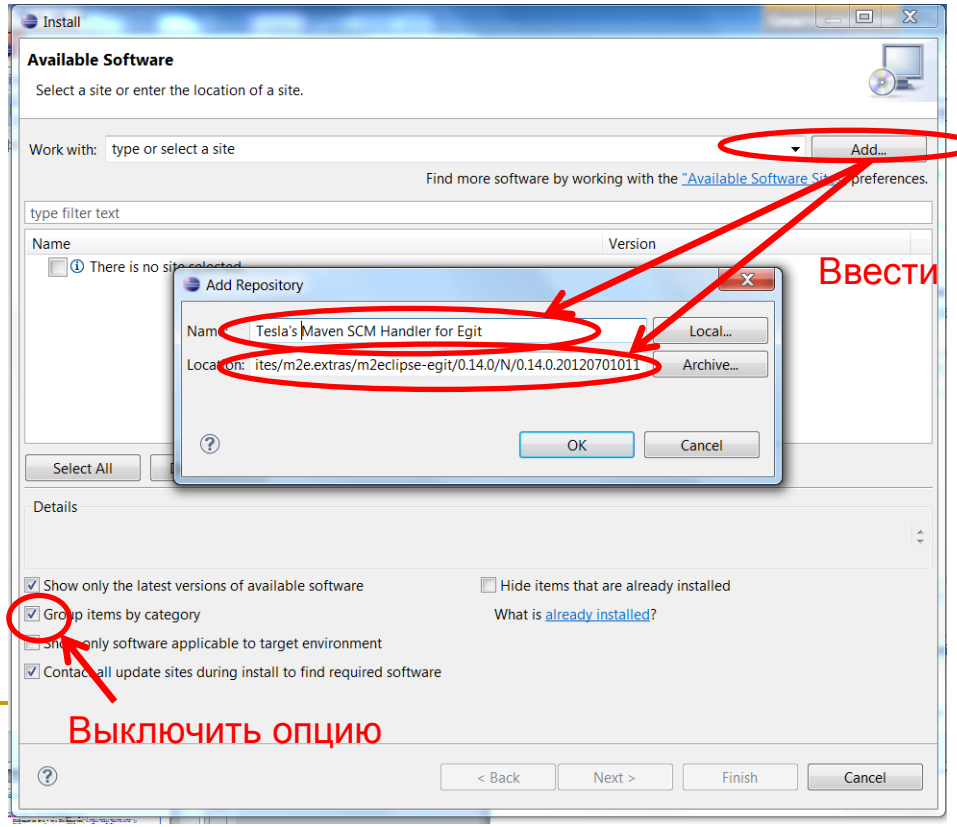
BioJava. Основные пакеты

Группа пакетов	Функциональность пакета
Genetic Algorithm Framework (генетические алгоритмы)	Классы для работы с генетическими алгоритмами
Experiment Phylogeny packages (экспериментальные пакеты для филогении)	Классы для работы с алгоритмами филогении, включая экспорт и импорт формата PHYLIP
Другие пакеты	Классы для работы с различными форматами биологических данных, классы встраивания сторонних графических утилит (например, Jmol) и др.

BioJava. Установка и работа с Eclipse

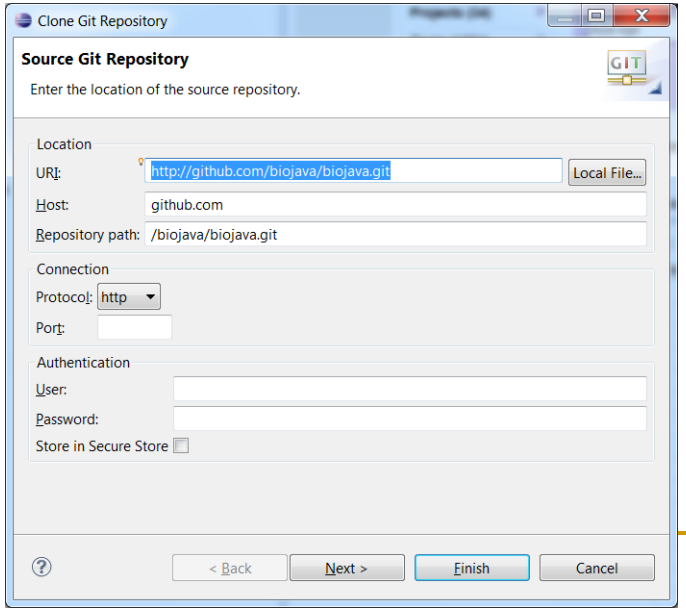
- Используем свежую версию Eclipse (не старше Indigo)
- Устанавливаем
  - плагин Maven Integration for Eclipse, Maven (Help->Eclipse Marketplace)
  - плагин Maven SCM Handler for Egit (Help->Install New Software…) (http://repository.tesla.io:8081/nexus/content/sites/m2e.extras/m2eclipse-egit/0.14.0/N/0.14.0.20120701011/)

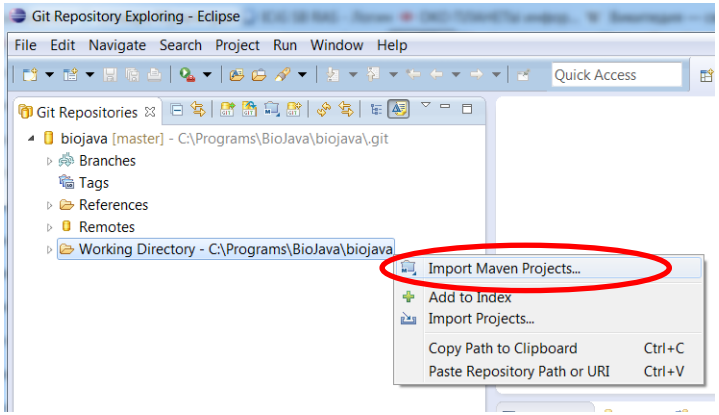




BioJava. Установка и работа с Eclipse

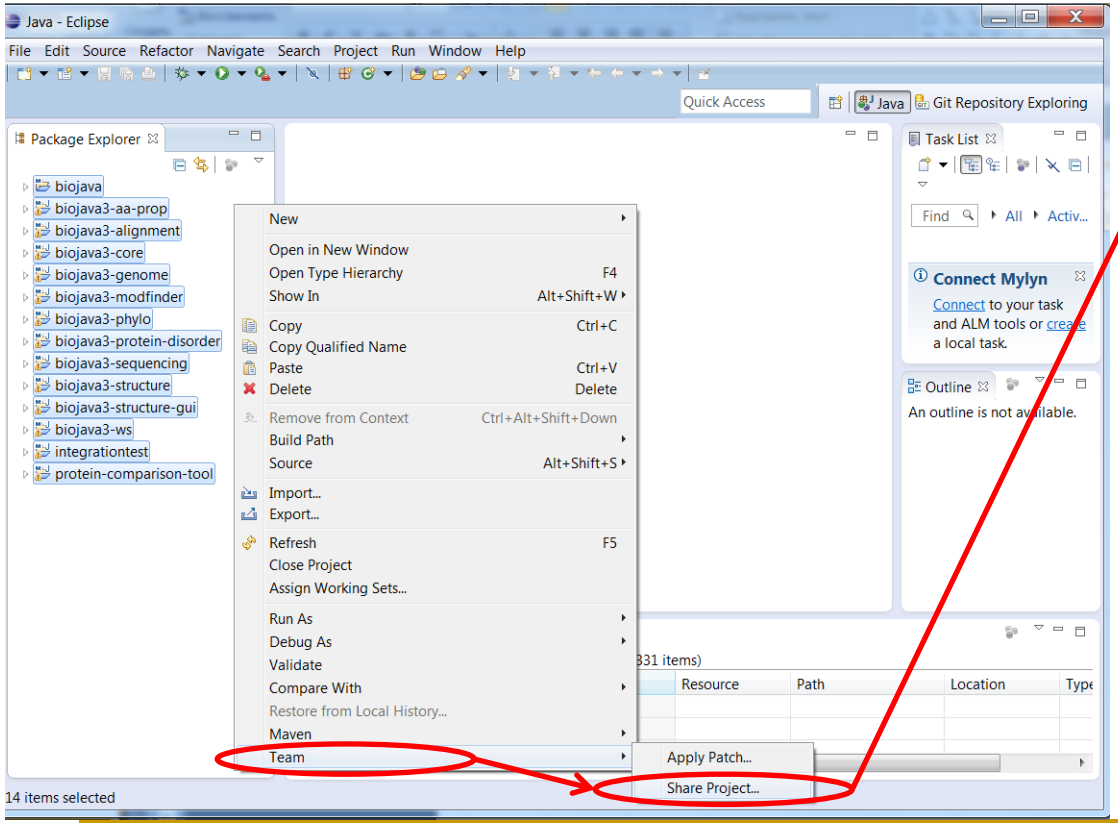
- Переключаемся на перспективу Git Repository Exploring (Window->Open Perspective->Other…)
- Делаем копию проекта BioJava (Clone a Git Repository) http://github.com/biojava/biojava.git
- Импортируем проект

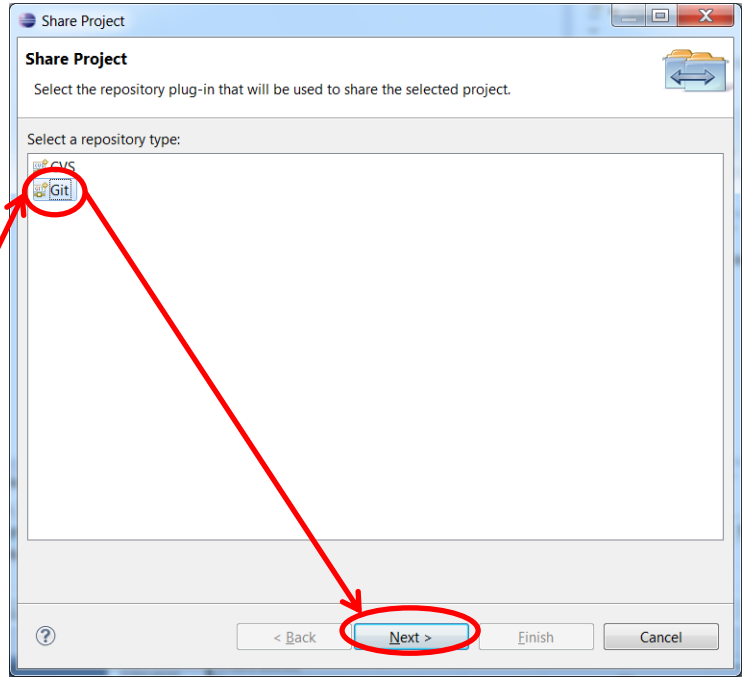




BioJava. Установка и работа с Eclipse

- Переключаемся на перспективу Java
- Выделяем все проекты и подключаем проект к репозиторию





Подробнее: http://biojava.org/wiki/BioJava3_eclipse

BioJava. Работа с последовательностями

```
public static void main(String[] args) {
    DNASequence dna = new DNASequence("AATGAATCGAATGAATGAATG");
    RNASequence rna1 = dna.getRNASequence(Frame.ONE);
    RNASequence rna2 = dna.getRNASequence(Frame.TWO);
    ProteinSequence protein1 = rna1.getProteinSequence();
    ProteinSequence protein2 = rna2.getProteinSequence();

    System.out.println("РНК1\t"+rna1);
    System.out.println("РНК2\t"+rna2);
    System.out.println("Белок1\t"+protein1);
    System.out.println("Белок2\t"+protein2);
}
```

```
РНК1 AAUGAAUCGAAUGAAUGAAUG
РНК2 AUGAAUCGAAUGAAUGAAUG
Белок1 NESNE*M
Белок2 MNRMNE
```

BioJava. Работа с последовательностями

- Варианты транскрипции последовательности

```
public static void main(String[] args) {
    DNASequence dna = new DNASequence("AATGAATCGAATGAATGAATG");
    RNASequence[] rna = new RNASequence[6];
    rna[0] = dna.getRNASequence(Frame.ONE);
    rna[1] = dna.getRNASequence(Frame.TWO);
    rna[2] = dna.getRNASequence(Frame.THREE);
    rna[3] = dna.getRNASequence(Frame.REVERSED_ONE);
    rna[4] = dna.getRNASequence(Frame.REVERSED_TWO);
    rna[5] = dna.getRNASequence(Frame.REVERSED_THREE);

    for(RNASequence r : rna){
        System.out.println(r);
    }
}
```

```
AAUGAAUCGAAUGAAUGAAUG
AUGAAUCGAAUGAAUGAAUG
UGAAUCGAAUGAAUGAAUG
CAUUCAUUCAUUCGAUUCAUU
AUUCAUUCAUUCGAUUCAUU
UUCAUUCAUUCGAUUCAUU
```

Лекция 10
Другие биологические данные, программные библиотеки для работы с ними

Аннотация белков. Формат SwissProt

- **UniProtKB/SWISS-PROT** – экспертная база данных белковых последовательностей, поддерживающая несколько уровней аннотации:
  - Функция белка
  - Доменная структура белка
  - Пост-трансляционные модификации
  - Варианты
  - И т.д. …

- http://web.expasy.org/docs/swiss-prot_guideline.html

Аннотация белков. Формат SwissProt

```
ID TNFA_HUMAN STANDARD; PRT; 233 AA.
AC P01375;
DT 21-JUL-1986 (REL. 01, CREATED)
DT 01-FEB-1995 (REL. 31, LAST ANNOTATION UPDATE)
DE TUMOR NECROSIS FACTOR PRECURSOR (TNF-ALPHA) (CACHECTIN).
GN TNFA.
OS HOMO SAPIENS (HUMAN).
OC EUKARYOTA; METAZOA; CHORDATA; VERTEBRATA; TETRAPODA; MAMMALIA;
OC EUTHERIA; PRIMATES.
RN [1]
RP SEQUENCE FROM N.A.
RX MEDLINE; 87217060.
RA NEDOSPASOV S.A., SHAKHOV A.N., TURETSKAYA R.L., METT V.A.,
RA AZIZOV M.M., GEORGIEV G.P., KOROBKO V.G., DOBRYNIN V.N.,
RA FILIPPOV S.A., BYSTROV N.S., BOLDYREVA E.F., CHUVPILO S.A.,
RA CHUMAKOV A.M., SHINGAROVA L.N., OVCHINNIKOV Y.A.;
RL COLD SPRING HARB. SYMP. QUANT. BIOL. 51:611-624(1986).
CC -!- FUNCTION: CYTOKINE WITH A WIDE VARIETY OF FUNCTIONS: IT CAN
CC CAUSE CYTOLYSIS OF CERTAIN TUMOR CELL LINES, IT IS IMPLICATED
...
DR EMBL; X02910; HSTNFA.
...
KW CYTOKINE; CYTOTOXIN; TRANSMEMBRANE; GLYCOPROTEIN; SIGNAL-ANCHOR;
...
FT PROPEP 1 76
FT CHAIN 77 233 TUMOR NECROSIS FACTOR.
FT TRANSMEM 36 56 SIGNAL-ANCHOR (TYPE-II PROTEIN).
...
FT MUTAGEN 108 108 R->W: BIOLOGICALLY INACTIVE.
...
SQ SEQUENCE 233 AA; 25644 MW; 279986 CN;
MSTESMIRDV ELAEEALPKK TGGPQGSRRC LFLSLFSFLI VAGATTLFCL LHFGVIGPQR
EEFPRDLSLI SPLAQAVRSS SRTPSDKPVA HVVANPQAEG QLQWLNRRAN ALLANGVELR
DNQLVVPSEG LYLIYSQVLF KGQGCPSTHV LLTHTISRIA VSYQTKVNLL SAIKSPCQRE
TPEGAEAKPW YEPIYLGGVF QLEKGDRLSA EINRPDYLDF AESGQVYFGI IAL
//
```

Аннотация белков. Формат SwissProt

Идентификатор	Значение
ID	Идентификатор (вкл. таксономическую классификацию организма)ч
AC	Идентификатор последовательности
PR	Идентификатор проекта
DT	Дата
DE	Описание
KW	Ключевые слова
OS	Вид организма
OC	Классификация организма
OG	Органелла

Аннотация белков. Формат SwissProt

Использование идентификатора ID

```
         1         2         3         4         5         6         7
1234567890123456789012345678901234567890123456789012345678901234567890
ID   CYC_BOVIN      STANDARD;      PRT;   104 AA.
ID   GIA2_GIALA     PRELIMINARY;   PRT;   296 AA.
```

Использование идентификатора DE

```
         1         2         3         4         5         6         7
1234567890123456789012345678901234567890123456789012345678901234567890
DE   NADH DEHYDROGENASE (EC 1.6.99.3).

DE   LYSOPINE DEHYDROGENASE (EC 1.5.1.16) (OCTOPINE SYNTHASE)
DE   (LYSOPINE SYNTHASE) (FRAGMENT).
```

Использование идентификатора OS

```
         1         2         3         4         5         6         7
1234567890123456789012345678901234567890123456789012345678901234567890
OS   ESCHERICHIA COLI.
OS   HOMO SAPIENS (HUMAN).
OS   ROUS SARCOMA VIRUS (STRAIN SCHMIDT-RUPPIN).
OS   NAJA NAJA (INDIAN COBRA), AND NAJA NIVEA (CAPE COBRA).
```

```
import org.biojava.bio.seq.*;
import org.biojava.bio.seq.io.*;
import java.io.*;
import org.biojava.bio.*;
import java.util.*;

public class ReadSwiss {
  public static void main(String[] args) {
    BufferedReader br = null;
    try {

      //create a buffered reader to read the sequence file specified by args[0]
      br = new BufferedReader(new FileReader(args[0]));

    }
    catch (FileNotFoundException ex) {
      //can't find the file specified by args[0]
      ex.printStackTrace();
      System.exit(-1);
    }
    //read the SwissProt File
    SequenceIterator sequences = SeqIOTools.readSwissprot(br);
    //iterate through the sequences
    while(sequences.hasNext()){
      try {
        Sequence seq = sequences.nextSequence();
        //do stuff with the sequence
      }
      catch (BioException ex) {
        //not in SwissProt format
        ex.printStackTrace();
      }catch (NoSuchElementException ex) {
        //request for more sequence when there isn't any
        ex.printStackTrace();
      }
    }
  }
```

http://www.biojava.org/docs/bj_in_anger/ReadingGES.htm

Формат PDB

- PDB (Protein Data Bank) – формат описания трёхмерной структуры белковых молекул.
- Описывает и аннотирует структуру белков и нуклеиновых кислот:
  - координаты атомов
  - выравнивания вторичной структуры
  - связи между атомами
- Используется для описания других биологически значимых молекул (вода, ионы, лиганды и т.д.)
- www.pdb.org

Структура записи PDB: заголовок

```
HEADER    TRANSFERASE(PHOSPHOTRANSFERASE)         08-JAN-93   1ATP
TITLE     2.2 ANGSTROM REFINED CRYSTAL STRUCTURE OF THE CATALYTIC
TITLE    2 SUBUNIT OF CAMP-DEPENDENT PROTEIN KINASE COMPLEXED WITH
TITLE    3 MNATP AND A PEPTIDE INHIBITOR
SOURCE    RECOMBINANT MOUSE (MUS $MUSCULUS) "ALPHA" ISOENZYME
SOURCE   2 EXPRESSED IN (ESCHERICHIA $COLI)
COMPND    MOL_ID: 1;
COMPND   2 MOLECULE: CAMP-DEPENDENT PROTEIN KINASE;
COMPND   3 CHAIN: E;
COMPND   4 EC: 2.7.1.37;

AUTHOR    J.ZHENG,E.A.TRAFNY,D.R.KNIGHTON,N.-H.XUONG,S.S.TAYLOR,
AUTHOR   2 L.F.TEN *EYCK,J.M.SOWADSKI
REVDAT   1   15-APR-93 1ATP    0
JRNL        AUTH   J.ZHENG,E.A.TRAFNY,D.R.KNIGHTON,N.-H.XUONG,
JRNL        AUTH 2 S.S.TAYLOR,L.F.TEN *EYCK,J.M.SOWADSKI
JRNL        REFN   ASTM                                          353
REMARK   1
REMARK   1 REFERENCE 1
REMARK   1  AUTH   D.R.KNIGHTON,S.M.BELL,J.ZHENG,L.F.TEN *EYCK,
REMARK   1  AUTH 2 N.-H.XUONG,S.S.TAYLOR,J.M.SOWADSKI

REMARK   8 OF THE CATALYTIC SUBUNIT, WHICH ARE PRESUMED DISORDERED.
REMARK   9
REMARK   9 RESIDUES E 317-E 321 AND I 23-I 24 HAVE WEAK
REMARK   9 BACKBONE ELECTRON DENSITY.
SEQRES   1 E  350  GLY ASN ALA ALA ALA ALA LYS LYS GLY SER GLU GLN GLU
SEQRES   2 E  350  SER VAL LYS GLU PHE LEU ALA LYS ALA LYS GLU ASP PHE

SEQRES  27 E  350  ILE ASN GLU LYS CYS GLY LYS GLU PHE THR GLU PHE
SEQRES   1 I   20  THR THR TYR ALA ASP PHE ILE ALA SER GLY ARG THR GLY
SEQRES   2 I   20  ARG ARG ASN ALA ILE HIS ASP
FTNOTE   1
FTNOTE   1 RESIDUES THR E 197 AND SER E 338 ARE PHOSPHORYLATED.
HET    PHO  E 197       4     PHOSPHATE GROUP BONDED TO THR E 197
HET    PHO  E 338       4     PHOSPHATE GROUP BONDED TO SER E 338
HET    ATP      1      31     ADENOSINE 5-*PRIME-*TRIPHOSPHATE
HET     MN      2       1     MANGANESE 2+ ION
HET     MN      3       1     MANGANESE 2+ ION
FORMUL   3  PHO    2(O3 P1 --)
FORMUL   4  ATP    C10 H16 N5 O13 P3
FORMUL   5   MN    2(MN1 ++)
FORMUL   6  HOH   *103(H2 O1)
HELIX    1   A LYS E   16  GLU E   31  1
HELIX    2  AB LEU E   40  GLN E   42  5 NOT NOTED IN REF 3
HELIX    3   B LYS E   76  LYS E   81  1
HELIX    4   C ILE E   85  ALA E   97  1


SHEET    2   B 2 LEU E 172  ILE E 174 -1
SHEET    1   C 2 ILE E 180  VAL E 182  0
SHEET    2   C 2 LYS E 189  ARG E 190 -1
CRYST1   73.580   76.280   80.580  90.00  90.00  90.00 P 21 21 21    4
```

Заголовок файла с названием молекулы и ссылкой на публикацию

Дополнительная информация (замечания)

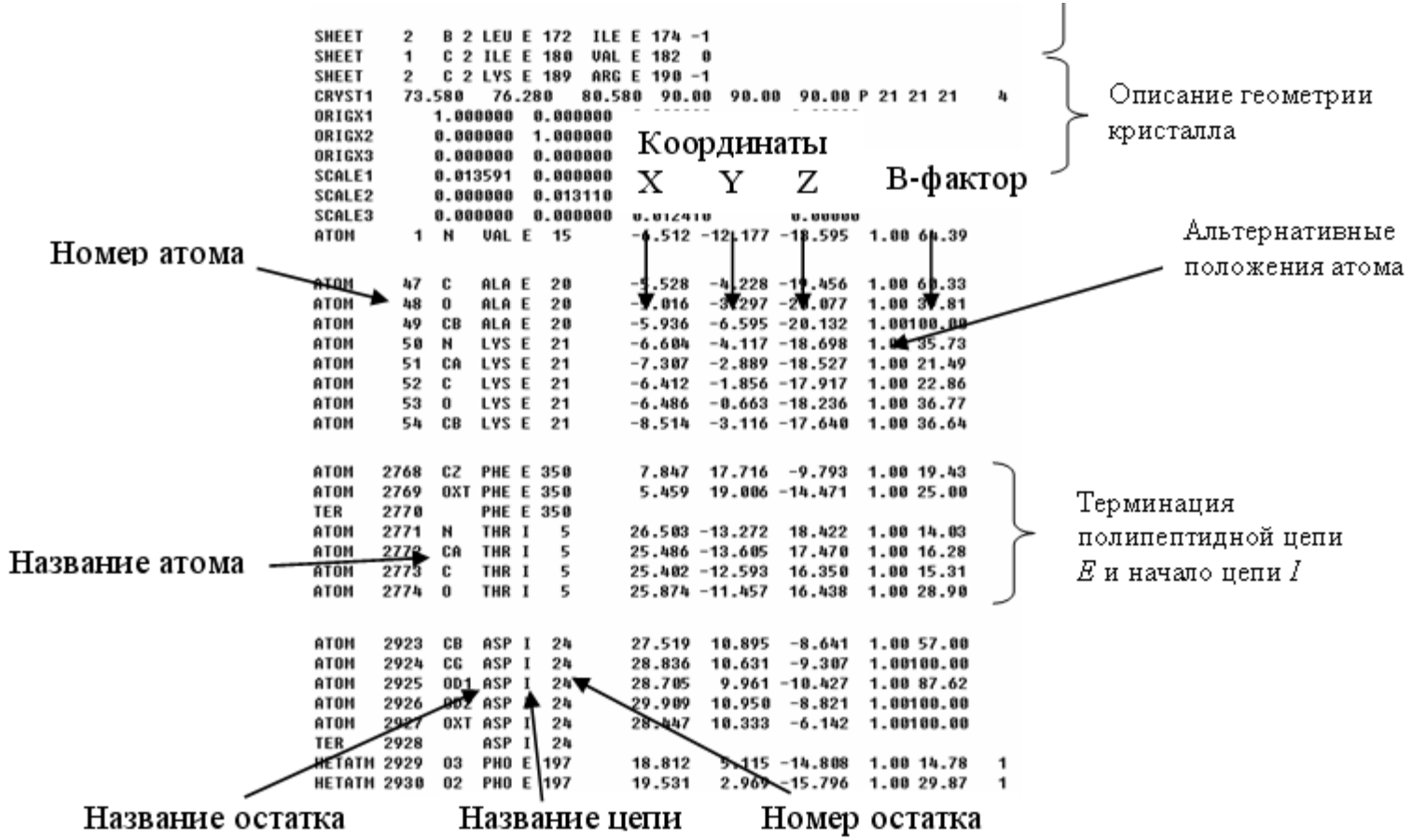
Первичная структура макромолекул (для каждой цепи)

Список гетероатомов (лигандов) и их химические формулы

Описание вторичной структуры

Описание геометрии

Структура записи PDB: координаты атомов



Пространственная структура белка: база PDB

Файл PDB описывает трехмерные координаты атомов белка

Номер атома

X Y Z B-фактор

```
ATOM      7  N   ALA A   2      26.095  -6.510   1.967  1.00  5.91
ATOM      8  CA  ALA A   2      25.775  -5.732   3.200  1.00  5.70
ATOM      9  C   ALA A   2      26.493  -4.380   3.152  1.00  5.15
ATOM     10  O   ALA A   2      27.457  -4.164   3.859  1.00  5.42
ATOM     11  CB  ALA A   2      26.220  -6.494   4.447  1.00  6.12
ATOM     12 1H   ALA A   2      26.814  -6.002   1.414  1.00  6.16
ATOM     13 2H   ALA A   2      26.459  -7.446   2.232  1.00  5.89
ATOM     14 3H   ALA A   2      25.234  -6.622   1.394  1.00  6.18
ATOM     15  HA  ALA A   2      24.710  -5.565   3.246
ATOM     16 1HB  ALA A   2      27.107  -7.069   4.226
ATOM     17 2HB  ALA A   2      26.437  -5.795   5.242
ATOM     18 3HB  ALA A   2      25.434  -7.162   4.766
ATOM     19  N   LYS A   3      26.003  -3.500   2.313
ATOM     20  CA  LYS A   3      26.636  -2.149   2.196
ATOM     21  C   LYS A   3      25.564  -1.056   2.283
ATOM     22  O   LYS A   3      25.532  -0.290   3.226
ATOM     23  CB  LYS A   3      27.367  -2.041   0.849
ATOM     24  CG  LYS A   3      27.528  -3.441   0.229
ATOM     25  CD  LYS A   3      26.173  -3.946  -0.295
ATOM     26  CE  LYS A   3      26.288  -4.220  -1.797
```

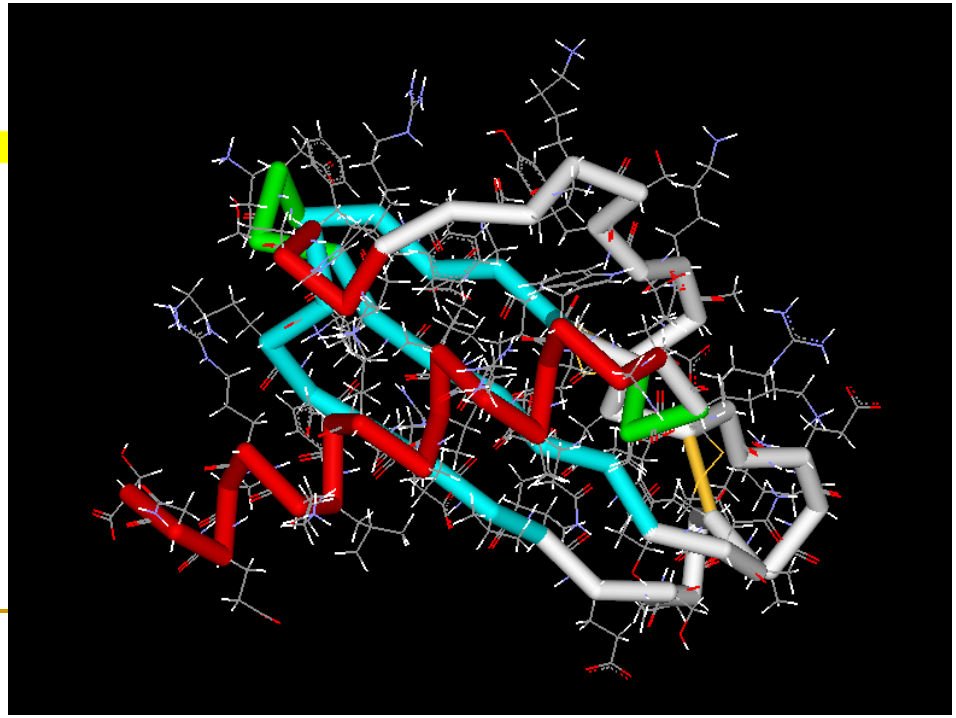
Альтернат. положения

Название цепи

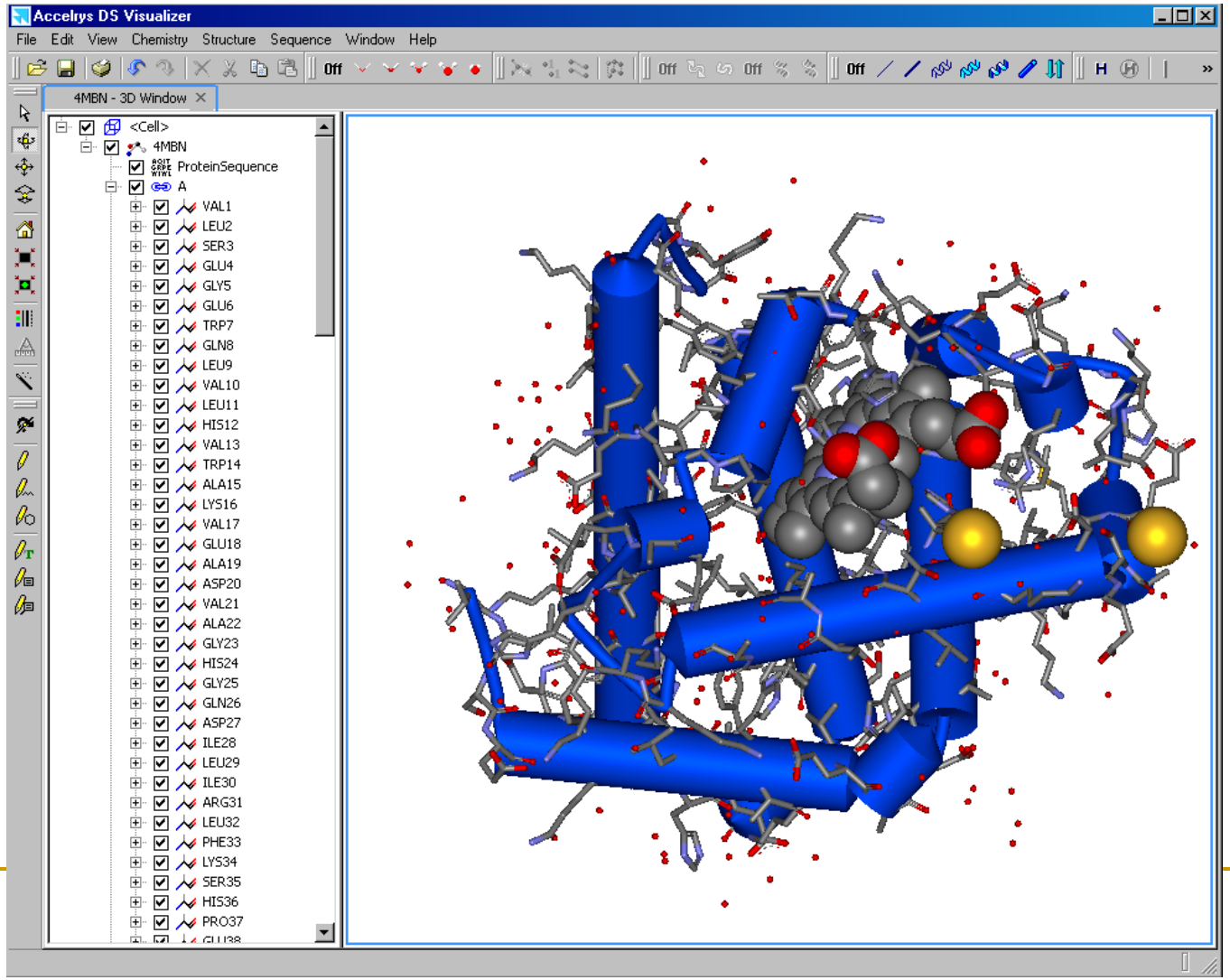
Название атома

Название остатка

Номер остатка



Программы визуализации структур (DSVisualizer)



Чтение PDB с помощью BioJava

```
public void basicLoad(String pdbId){
      try {
         PDBFileReader reader = new PDBFileReader();
         reader.setPath("/tmp");// the path to the local PDB installation

         // are all files in one directory, or are the files split,
         // as on the PDB ftp servers?
         reader.setPdbDirectorySplit(true);
         // should a missing PDB id be fetched automatically from the FTP servers?
         reader.setAutoFetch(true);
         // configure the parameters of file parsing
         FileParsingParameters params = new FileParsingParameters();
       // should the ATOM and SEQRES residues be aligned when creating the internal data model?
         params.setAlignSeqRes(true);
         // should secondary structure get parsed from the file
         params.setParseSecStruc(false);
         params.setLoadChemCompInfo(true);
         reader.setFileParsingParameters(params);

         Structure structure = reader.getStructureById(pdbId);
         System.out.println(structure);
         for (Chain c: structure.getChains()){
            System.out.println("Chain " + c.getName() + " details:");
            System.out.println("Atom ligands: " + c.getAtomLigands());
            System.out.println(c.getSeqResSequence());
         }
      } catch (Exception e){
         e.printStackTrace();
      }
 }
```

http://biojava.org/wiki/BioJava:CookBook:PDB:read3.0

Чтение PDB с помощью BioJava

```
structure  4HHB Authors: G.FERMI,M.F.PERUTZ IdCode: 4HHB Classification: OXYGEN TRANSPORT DepDate: Wed Mar 07 00:00:00 PST 1984
Technique: X-RAY DIFFRACTION  Resolution: 1.74 ModDate: Tue Feb 24 00:00:00 PST 2009 Title: THE CRYSTAL STRUCTURE OF HUMAN
DEOXYHAEMOGLOBIN AT 1.74 ANGSTROMS RESOLUTION
 chains:
chain 0: >A< HEMOGLOBIN (DEOXY) (ALPHA CHAIN)
 length SEQRES: 141 length ATOM: 198 aminos: 141 hetatms: 57 nucleotides: 0
chain 1: >B< HEMOGLOBIN (DEOXY) (BETA CHAIN)
 length SEQRES: 146 length ATOM: 205 aminos: 146 hetatms: 59 nucleotides: 0
chain 2: >C< HEMOGLOBIN (DEOXY) (ALPHA CHAIN)
 length SEQRES: 141 length ATOM: 201 aminos: 141 hetatms: 60 nucleotides: 0
chain 3: >D< HEMOGLOBIN (DEOXY) (BETA CHAIN)
 length SEQRES: 146 length ATOM: 197 aminos: 146 hetatms: 51 nucleotides: 0
DBRefs: 4
DBREF  4HHB A    1   141  UNP    P69905   HBA_HUMAN        1    141
DBREF  4HHB B    1   146  UNP    P68871   HBB_HUMAN        1    146
DBREF  4HHB C    1   141  UNP    P69905   HBA_HUMAN        1    141
DBREF  4HHB D    1   146  UNP    P68871   HBB_HUMAN        1    146
Molecules:
Compound: 1 HEMOGLOBIN (DEOXY) (ALPHA CHAIN) Chains: ChainId: A C Engineered: YES OrganismScientific: HOMO SAPIENS OrganismTaxId:
9606 OrganismCommon: HUMAN
Compound: 2 HEMOGLOBIN (DEOXY) (BETA CHAIN) Chains: ChainId: B D Engineered: YES OrganismScientific: HOMO SAPIENS OrganismTaxId:
9606 OrganismCommon: HUMAN

Chain A details:
Atom ligands: [Hetatom 142 HEM true atoms: 43]
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFL
ASVSTVLTSKYR
Chain B details:
Atom ligands: [Hetatom 147 PO4 true atoms: 1, Hetatom 148 HEM true atoms: 43]
VHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAA
YQKVVAGVANALAHKYH
Chain C details:
Atom ligands: [Hetatom 142 HEM true atoms: 43]
VLSPADKTNVKAAWGKVGAHAGEYGAEALERMFLSFPTTKTYFPHFDLSHGSAQVKGHGKKVADALTNAVAHVDDMPNALSALSDLHAHKLRVDPVNFKLLSHCLLVTLAAHLPAEFTPAVHASLDKFL
ASVSTVLTSKYR
Chain D details:
Atom ligands: [Hetatom 147 PO4 true atoms: 1, Hetatom 148 HEM true atoms: 43]
VHLTPEEKSAVTALWGKVNVDEVGGEALGRLLVVYPWTQRFFESFGDLSTPDAVMGNPKVKAHGKKVLGAFSDGLAHLDNLKGTFATLSELHCDKLHVDPENFRLLGNVLVCVLAHHFGKEFTPPVQAA
YQKVVAGVANALAHKYH
```

http://biojava.org/wiki/BioJava:CookBook:PDB:read3.0

Форматы представления биологических данных: SBML

- SBML (System Biology Markup Language – Язык Разметки для Системной Биологии) – создан на основе языка разметки XML [Hucka et al., 2003], www.sbml.org .
- Имеет хорошо документированную структуру и набор программных компонент (представленных для разных языков моделирования) осуществляющих как чтение, так и манипуляцию моделями.
- Позволяет использовать большой спектр стандартных программных разборщиков, без необходимости разработки собственной программы.
- На сегодня данный стандарт поддерживают более 150 программных систем, решающие задачи моделирования и анализа различных аспектов функционирования молекулярно-генетических систем.

Форматы представления биологических данных: SBML

Начало определения модели
- список функций (опционально)
- список единиц измерения (опционально)
- список типов компартментов (опционально)
- список типов соединений (опционально)
- список компартментов (опционально)
- список соединений (опционально)
- список параметров (опционально)
- список начальных концентраций (опционально)
- список правил (опционально)
- список ограничений (опционально)
- список реакций (опционально)
- список событий (опционально)

Конец определения модели

Схема SBML-документа

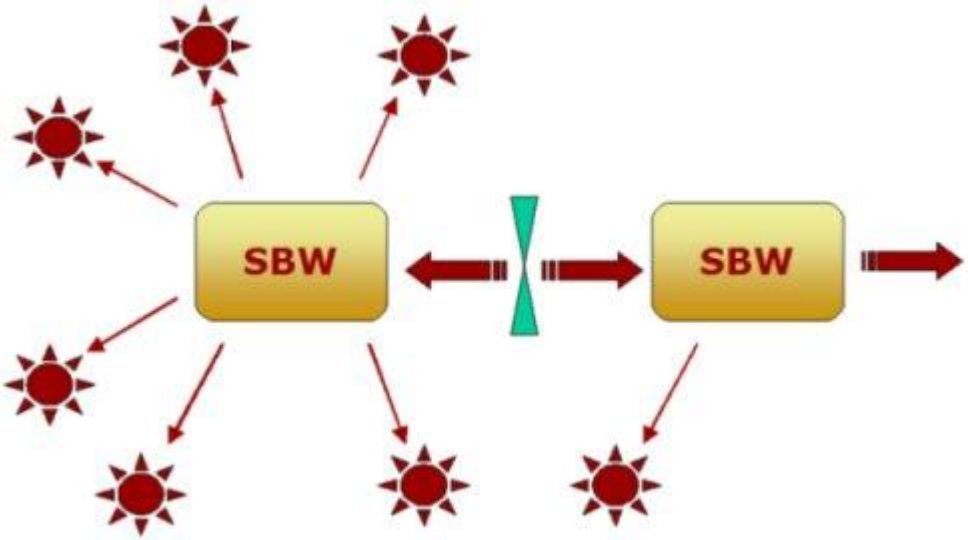
Форматы представления биологических данных: SBML

Фрагмент SBML-документа

```
<?xml version="1.0" encoding="UTF-8" ?>
- <sbml level="2" version="3" xmlns="http://www.sbml.org/sbml/level2/version3">
  - <model name="EnzymaticReaction">
    - <listOfUnitDefinitions>
      - <unitDefinition id="per_second">
        - <listOfUnits>
            <unit kind="second" exponent="-1" />
          </listOfUnits>
        </unitDefinition>
      - <unitDefinition id="litre_per_mole_per_second">
        - <listOfUnits>
            <unit kind="mole" exponent="-1" />
            <unit kind="litre" exponent="1" />
            <unit kind="second" exponent="-1" />
          </listOfUnits>
        </unitDefinition>
      </listOfUnitDefinitions>
    - <listOfCompartments>
        <compartment id="cytosol" size="1e-14" />
      </listOfCompartments>
    - <listOfSpecies>
        <species compartment="cytosol" id="ES" initialAmount="0" name="ES" />
        <species compartment="cytosol" id="P" initialAmount="0" name="P" />
        <species compartment="cytosol" id="S" initialAmount="1e-20" name="S" />
        <species compartment="cytosol" id="E" initialAmount="5e-21" name="E" />
      </listOfSpecies>
    - <listOfReactions>
      - <reaction id="veq">
        - <listOfReactants>
            <speciesReference species="E" />
            <speciesReference species="S" />
          </listOfReactants>
        - <listOfProducts>
            <speciesReference species="ES" />
          </listOfProducts>
```

Форматы представления биологических данных: SBW

- SBW (Systems Biology Workbench, www.sys-bio.org) – платформа, позволяющая обмениваться данными и математическими моделями между программами, написанными на разных языках (и для разных платформа)
- Использует SBML в качестве базового формата данных
- Предоставляет API для следующих языков
  - Java
  - C++/C
  - Perl
  - Delphi/Kylix
  - Python



© http://sbw.sourceforge.net/

Конвейерная обработка данных с помощью платформы Taverna

- Платформа Taverna (http://www.taverna.org.uk/) – свободно распространяемая программная платформа для проектирования и выполнения *программных конвейеров* (workflow) обработки данных.
- Ядро Taverna позволяет добавлять в систему новые программные ресурсы, используя технологию *веб-сервисов*.
- В числе основных поставщиков таких ресурсов:
  - Национальный Центр Биотехнологической Информации США (NCBI)
  - Европейский Институт Биоинформатики (EBI)
  - Японская База Данных ДНК (DDBJ)
- В настоящее время эта платформа, помимо биоинформатики, используется также в задачах хемоинформатики, астрономии, социальных и других наук.

Конвейерная обработка данных с помощью платформы Taverna