# Первая часть

# Что скачивать

JAVA SE Standart Edition (Core Java), EE, ME

java 6, 7, 8, 9

IDE Integrated Development Environment
(JetBrains IntelliJ IDEA, Eclipse, NetBeans)

JDK Java Development Kit
  Компилятор (javac.exe)
  ...
  JRE Java RunTime Environment
    JVM Java Virtual Machine (java.exe)
        (prog.class)
    ...
    ...

# Oracle Code Conventions

- Правила оформления кода

# Стиль оформления

- Название класса с большой буквы!
- Название всего остального с маленькой.
- Пустые строчки стараемся не использовать.
- Открывающие фигурные скобки на новой строчке не пишем (это стиль C#).
- Обязательная точка с запятой в конце каждой команды
- Все переменные пишем в camelCase (верблюжьяНотация)
- Автоматическое форматирование: Code — Reformat code

# Комментарии

- Три типа комментариев:

```
// однострочный комментарий

/*
    многострочный
    комментарий
     */

/**
 * javadoc
 * комментарий
 *
 * @author Author!
 */
```

# Class

- Название класса должно совпадать с названием файла (есть тонкости, но для начала следуем этому правилу)

# Метод

- Любой код нужно писать в методе

```
// метод, функция, процедура, подпрограмма -- для нас это практически синонимы

static void bark() {
    //sout + Tab
    //Ctrl-Y удаление строчек
    System.out.println("ruff-ruff");
    //Shift-Cmd-Enter, Shift-Ctrl-Enter – быстрый переход на новую строчку
}

static int aport() {
    int result = max(3, 5);
    //fixme поправить, здесь ошибка!
    return result;
}

static int max(int a, int b) {
    //todo написать вычисление
    return a;
}
```

# Точка входа в программу

- Точка входа:

```
//psvm + Tab
public static void main(String[] args) {
    int temp;
    temp = 3;
    String s = "Java";
    System.out.print("Hello, " + s + "\n");
    System.out.println("Hello, " + s);
    // Ctrl-/ Cmd-/ -- однострочный комментарий
    bark();
    System.out.println("Количество косточек: " + aport());
}
```

# О static

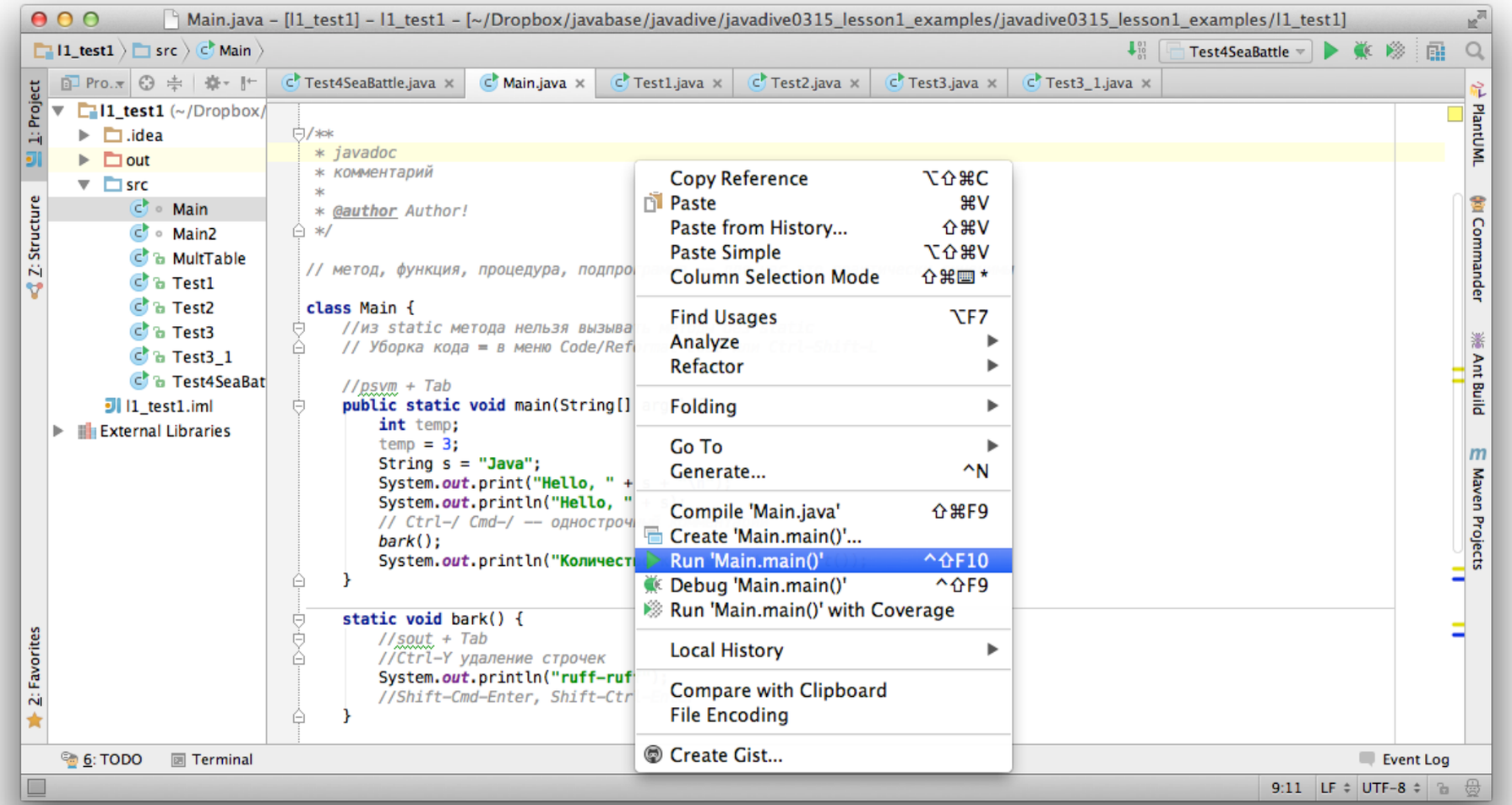
- Основное правило

```
//из static метода нельзя вызывать методы без static
```

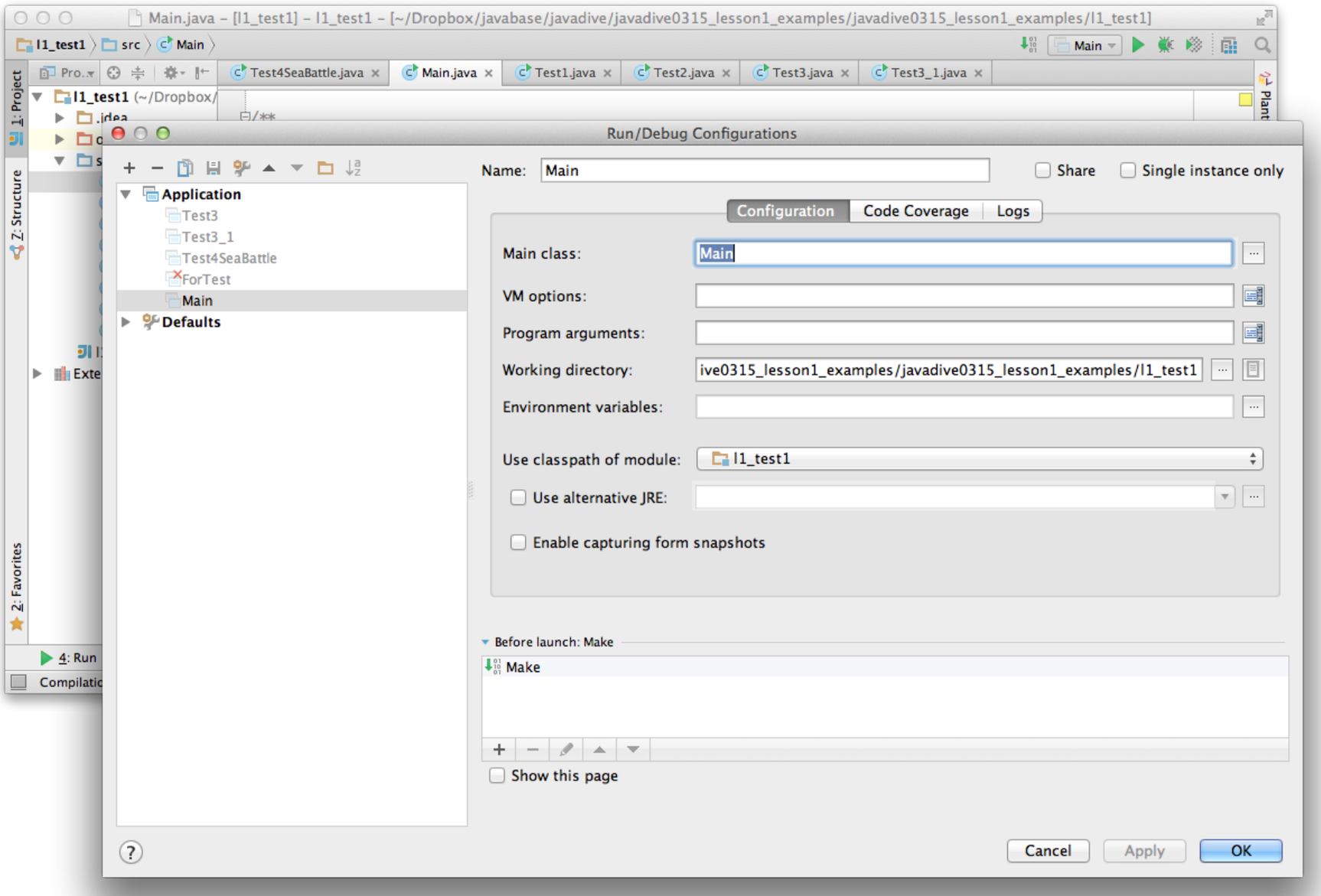
```
// метод, функция, процедура, подпрограмма

class Main {

    //psvm + Tab
    public static void main(String[] args) {
        String s = "Java";
        System.out.print("Hello, " + s + "\n");
        bark();
    }

    void bark() {
        //sout + Tab
        //Ctrl-Y удаление строчек
        System.out.println("ruff-ruff");
        //Shift-Cmd-Enter, Shift-Ctrl-Enter
    }

    int aport() {
        int result = max(3, 5);
        //fixme поправить, здесь ошибка!
        return result;
    }

    int max(int a, int b) {
        //todo написать вычисление
        return a;
    }
```

# Запуск программы

- Первый запуск — через правый клик мыши (конфигурация за нас будет создана автоматически)

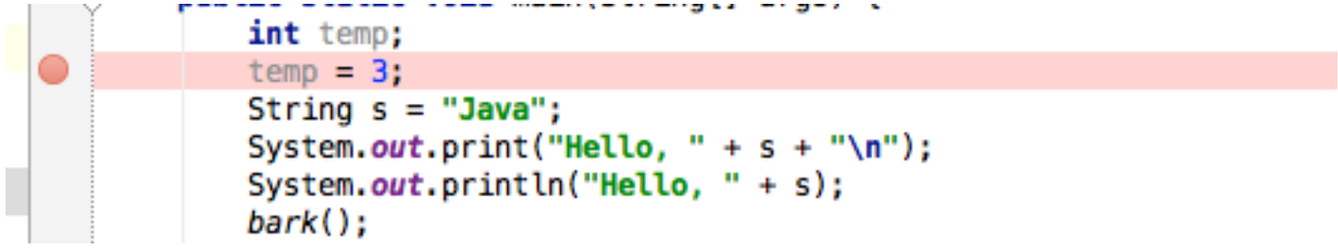


# Конфигурация

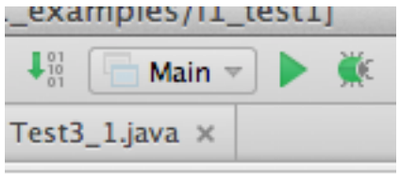


# Debug

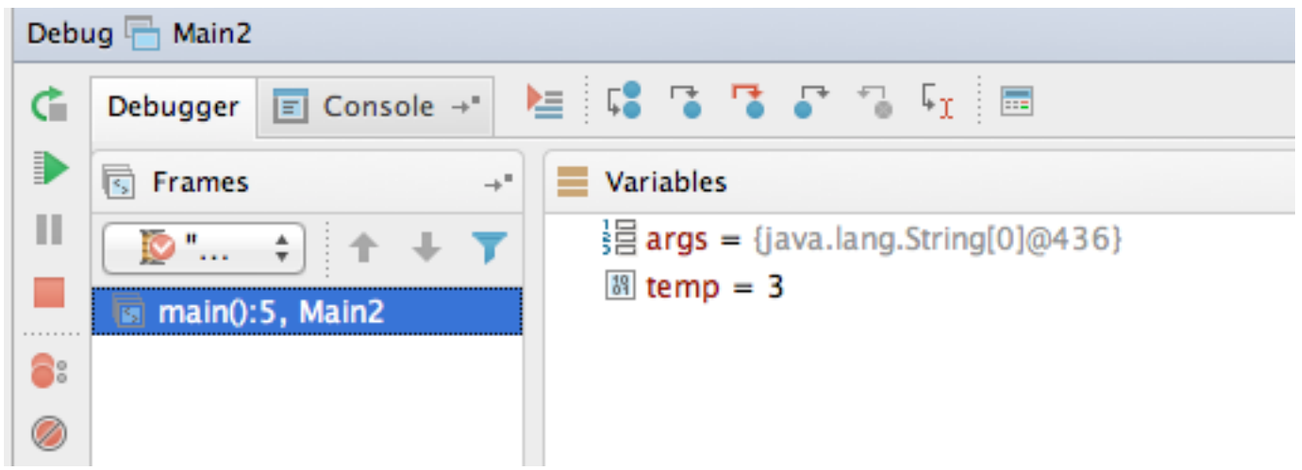
- Пошаговое исполнение: слева от кода нажимаем мышкой — текущая строчка выделяется, это так называемая точка останова




- Запуск отладки не через зеленую стрелку, а через картинку с жучком




Внизу панель управления — пошагового выполнения



# Модификаторы доступа

- Public, private — как в соц. сетях — это права доступа

# Вторая часть

# Типы данных

- Краткое введение

```
public class Test1 {
    public static void main(String[] args) {

        byte b1 = 127; // 255, -128 + 127
//        short sh = (short) 32700; // casting // 2 байта 65535; -32768 +32767
        short sh = 32700; // casting // 2 байта 65535; -32768 +32767
        System.out.println(sh);
        int number = 2000000000; // 4 байта // по умолчанию все целые числа -- это int
        long l = 2000000000000000000L; // 8 байт

        double d = 111.2; // по умолчанию все числовые значения с точкой -- это double
        float f = 11.2f;

        char ch = 65535;
        char ch2 = 'D';

        String str = "hello";

        boolean bool = true; //false

        System.out.println(Byte.MAX_VALUE);
        System.out.println((int)Character.MAX_VALUE);

    }
}
```

# Условные выражения

```
public class Test2 {
    public static void main(String[] args) {
        int a = 100;
        int b = 200;
        if (a < b) {
            System.out.println("True");
        } else {
            System.out.println("False");
        }

        int grade = 8;
        switch (grade) {
            case 5:
                System.out.println("Best!");
                break; // прерывает выполнение switch, без break выполнились бы и все последующие строчки
            case 4:
                System.out.println("Good!");
                break;
            case 3:
                System.out.println("norm");
                break;
            case 2:
                System.out.println("bad");
                break;
            default:
                System.out.println("???? what");
        }

    }
}
```

# Циклы

```
public class Test2 {
    public static void main(String[] args) {
       int i = 0;
        while (i < 3) {
            System.out.println(i);
            i++;
        }

        for (int j = 0; j < 3; j++) {
            System.out.println(j);
        }

        // fori + Tab
        for (int j = 0; j < 3; j++) {
            System.out.println(j);
        }

        int i1 = 0;
        do {
            System.out.println(i1);
            i1++;
        } while (i1 < 3);

        System.out.println("------");
    }
}
```

# Таблица умножения

```java
public class MultTable {
    public static void main(String[] args) {
        for (int i = 1; i <= 9; i++) {
            for (int j = 1; j <= 9; j++) {
                System.out.printf("%d * %d = %d\n", i, j, i * j);
            }
            System.out.println();
        }

        // souf
        System.out.printf("число %d, строка %s, дробное %.2f", 4, "hello", 45.5);

        // http://programador.ru/printf/
    }
}
```

# Массивы

```java
public class Test3 {
    public static void main(String[] args) {
        int a = 10;
        int[] numbers = {23, 24, 11, 324234, 233};
//        System.out.println(numbers[1]);
        System.out.printf("length %d\n", numbers.length);
        // itar + Tab
        for (int i = 0; i < numbers.length; i++) {
            int number = numbers[i];
            System.out.println(number);
        }

        // название: for each
        // iter + Tab
        for (int number : numbers) {
            System.out.println(number);
        }
    }
}
```

# Куча и Стек

- Примитивные типы хранятся в Стеке, а объектные — в Куче.
- Если что-то хотим разместить в Куче — нужно выделить память с помощью new, вот место для массива, как раз нужно выделять в Куче

```
        // в стеке
        int a;
        a = 10;

        // в куче
        int[] numbers;
//          numbers = выделить место для нашего массива
        numbers = new int[4];
        numbers[3] = 555;
```

# Двумерные массивы

- Двумерные массивы похожи на простые таблицы в Excel. Чтобы обратиться к какому-то элементу нужно указать столбец и строчку

```
char[][] cells2;
cells2 = new char[3][3];
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        cells2[i][j] = '.';
    }
}

cells2[1][1] = 'X';

for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        System.out.print(cells2[i][j]);
    }
    System.out.println();
}
```

# Scanner

- Класс Scanner помогает нам считать с клавиатуры ввод пользователя

```
Scanner scanner = new Scanner(System.in);
String s = scanner.nextLine(); // здесь программа будет ждать, пока пользователь не нажмет Enter
System.out.printf("Вы ввели: %s\n", s);
```

# Random

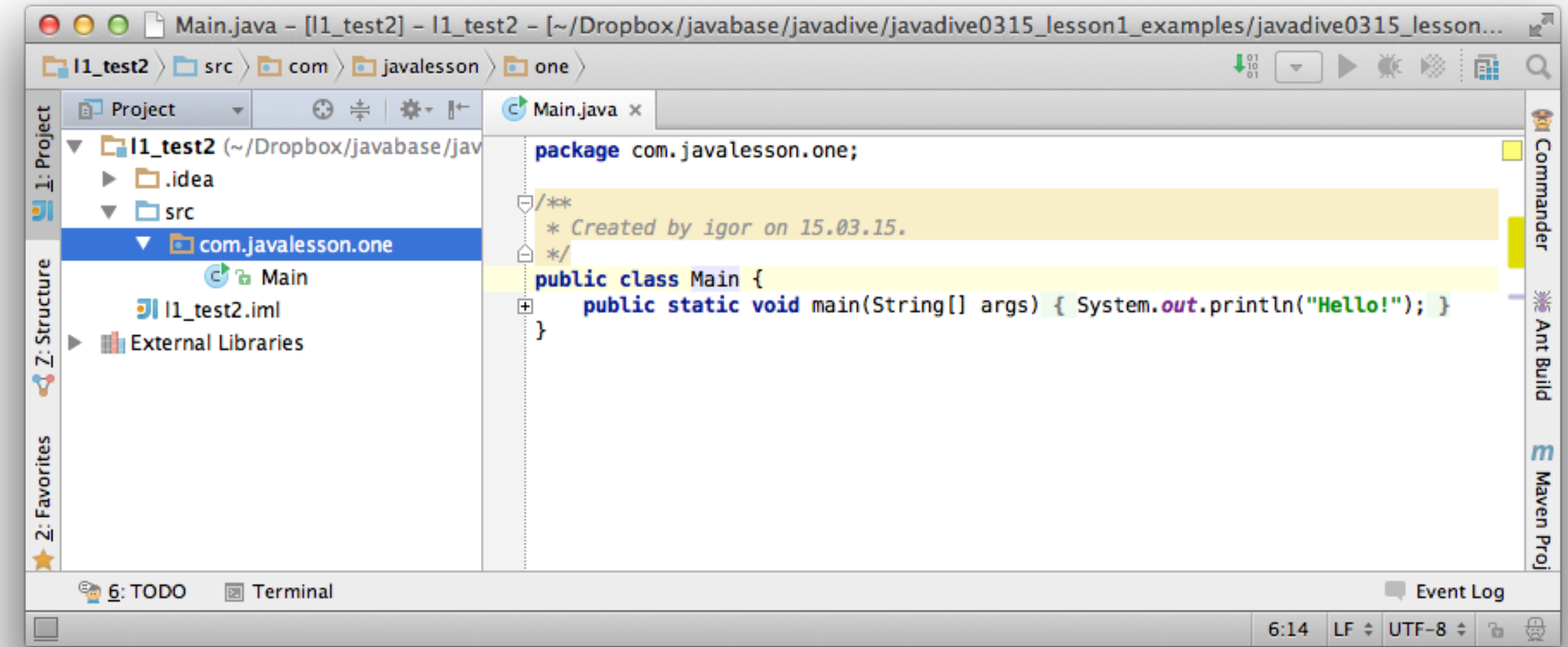
- Класс Random предоставляет нам случайные значения (если мы объявим переменную этого класса, а потом выделим для него память)

```
Random random = new Random();

int i = random.nextInt(100); // Случайное число от 0 до 99 (включительно)

int k = random.nextInt(10) + 1; // Случайное число от 1 до 10 (включительно)

boolean b = random.nextBoolean(); // Случайное значение: true или false
```

# Третья часть

# Пакеты

- Чтобы у разных программистов имена классов не путались, не совпадали, придумали Пакеты. Это просто обычные директории (папки), в которых лежат наши Классы. И имя каждого класса тогда получается длинное и уникальное



# ООП — введение в объектно-ориентированное программирование

- Класс (class) — представляет собой обобщенное, шаблонное описание какого-либо объекта из предметной области.

- Название класса как правило пишется в единственном числе (и с большой буквы!).

- Класс — это не какой-то конкретный объект, а всего лишь схема, по которой будет создан уже конкретный объект. Так же как по схеме космического корабля будут созданы реальные космические корабли.

- Тело класса заключено в фигурные скобки {}.

- Если попытаться описать любой класс, то эти описания можно разбить на две группы: 1) Характеристики, свойства этого класса (цвет, пол, возраст, имя…) и 2) Действия, умения, способности, то, что класс умеет делать (прыгать, бегать, выводить себя на экран, проверять конец игры или нет, расставлять корабли…)

- Характеристики (поля) записываются в переменные, а действия — в методы.

- По конвенции принято сначала описывать поля класса, а затем методы.

# Пример класса

```
class Cat {
    // Характеристики. Поля класса. Переменных

    String color; // = null;
    int age; // = 0;
    String name; // = null;
    boolean isMale; // = false;


    // Что класс умеет делать. Действия. Скилы. Методы

    void eat() {
        System.out.println("nyam-nyam");
        int i; // обратите внимание, что переменные внутри методов не инциализуруются нулем!
        // поэтому следующая строчка выдаст ошибку
//        System.out.println(i);
    }

    void drink() {
        System.out.println("пью воду");
    }

    void hunt() {
        System.out.println("охочусь");
    }

    void play() {
        System.out.println("играю с мышкой");
    }

    void about() {
        System.out.printf("name %s age %d\n", name, age);
    }
}
```

# Создание экземпляров

- Для создания экземпляров по шаблону класса нужно выделить память с помощью new (так же, как с массивами делали)

```
public class Main {
    public static void main(String[] args) {
        // Память в Java: Стек (для примитивных переменных) и Куча (для объектов)

        Cat cat1 = new Cat();

        cat1.age = 4;
        cat1.name = "Tom";
        cat1.about();

        Cat cat2 = new Cat();
        cat2.age = 3;
        cat2.name = "Murka";
        cat2.about();
    }
}
```

# Конструкторы

- Круглые скобки после названия класса, когда создаем экземпляр, это вызов спецального метода — Конструктора. Он автоматически создается в классе, если вручную его не описать

- Cat cat1 = **new** Cat();

# Описания конструкторов

- Мы можем и свои конструкторы в классе описать, но тогда Java не создаст за нас пустой конструктор! Если он нужен — его придется описывать самостоятельно!

```
class Cat {
    // Характеристики. Поля класса. Переменных
    String color; // = null;
    int age; // = 0;
    String name; // = null;


    // Конструктор
    public Cat(String name, int age, String color) {
        this.name = name;
        this.age = age;
        this.color = color;
    }

    public Cat(int age2, String name2) {
        age = age2;
        name = name2;
    }

    // Пустой конструктор создается автоматически, если нет ни одного конструктора
    // Но если хоть один конструктор есть, то пустой конструктор не будет автоматически создаваться!
    public Cat() {
    }

    // Что класс умеет делать. Действия. Скилы. Методы
}
```

# O this

- Специально слово this помогает нам отличить поля класса от параметров метода, когда у них одинаковые названия!

- Можно так:

```
public Cat(int age2, String name2) {
    age = age2;
    name = name2;
 }
```

- А можно через this:

```
// Конструктор
public Cat(String name, int age, String color) {
    this.name = name;
    this.age = age;
    this.color = color;
}
```

# Создание экземпляров с помощью своего конструктора

- Создание экземпляра класса с инициализацией полей может выглядеть короче, чем без конструктора:

```
public class Main {
    public static void main(String[] args) {
        Cat cat3 = new Cat(4, "Masya");
        cat3.about();

        Cat cat1 = new Cat();
        cat1.age = 4;
        cat1.name = "Tom";
        cat1.about();
    }
}
```

# Сборщик мусора

- Память, выделенная для экземпляра класса, будет освобождена Сборщиком мусора (Garbage collector) в тот момент (примерно), когда он обнаружит, что на наш объект больше никто не ссылается, то есть его адрес не хранится ни в одной переменной

```java
public class Main {
    public static void main(String[] args) {
        // Память в Java: Стек (для примитивных переменных) и Куча (для объектов)

        Cat cat1 = new Cat();
        cat1.age = 4;
        cat1.name = "Tom";
        cat1.about();

        // Garbage collector. Сборщик мусора Java. Удаляет объекты, если на них нет никаких ссылок.

//        cat1 = null;
//        cat1 = new Cat();
//        cat1.about();
    }
}
```

# Инициализация переменных

При создании экземпляра класса
(то есть реального объекта), его поля инициализуются в нули, null или false
Но при создании простой переменной в методе — инициализации не происходит!

```
class Cat {
    // Характеристики. Поля класса. Переменных
    String color; // = null;
    int age; // = 0;
    String name; // = null;
    boolean isMale; // = false;


    // Что класс умеет делать. Действия. Скилы. Методы

    void eat() {
        System.out.println("nyam-nyam");
        int i; // обратите внимание, что переменные внутри методов не инциализуруются нулем!
        // поэтому следующая строчка выдаст ошибку
//          System.out.println(i);
    }
}
```

# Создание массива объектов

- Пусть у нас будет класс Horse (Лошадь)
- Если мы хотим несколько лошадей хранить в массиве, то нам придется выделять память несколько раз:
- Первый раз для самого массива (стойла, загона):

```
Horse[] horses = new Horse[14];
```

- А затем уже для каждой лошади отдельно:

```
for (int i = 0; i < horses.length; i++) {
    horses[i] = new Horse();
    horses[i].speed = 10 + random.nextInt(100); // Ctrl-J Документация
    horses[i].name = "Буцефал " + i;
    horses[i].age = random.nextInt(10) + 1;
    horses[i].isMale = random.nextBoolean();
}
```

- Зато в дальнейшем удобно пробежать по загону и сказать каждой лошади: «скачи»

```
for (Horse horse : horses) {
    horse.ride();
}
```

# Класс Лошадь (целиком)

```java
class Horse {
    // Характеристики. Поля класса. Переменных
    String color;
    int age;
    String name;
    int speed;
    boolean isMale;

    // Что класс умеет делать. Действия. Скилы. Методы

//    Horse() {
//    }

    void ride() {
        about();
        System.out.println("riding at speed " + speed);
    }

    void eat() {
        System.out.println("brrrrr");
    }

    void drink() {
        System.out.println("пью воду");
    }

    void play() {
        System.out.println("играю");
    }

    void about() {
        String sex = (isMale) ? "Male" : "Female"; // тернарный оператор
        System.out.printf("name %s age %d sex %s ", name, age, sex);
    }
}
```

# Класс Менеджер скачек (Main.java)

```java
import java.util.Random;

public class Main2 {
    public static void main(String[] args) {

        Random random = new Random();
        Horse[] horses = new Horse[14];
        for (int i = 0; i < horses.length; i++) {
            horses[i] = new Horse();
            horses[i].speed = 10 + random.nextInt(100); // Ctrl-J Документация
            horses[i].name = "Буцефал " + i;
            horses[i].age = random.nextInt(10) + 1;
            horses[i].isMale = random.nextBoolean();
        }

        for (Horse horse : horses) {
            horse.ride();
        }
    }
}
```

# Константы

- Константы — это переменные, которые нельзя поменять, в них удобно хранить какие-то настройки, например, количество ячеек на поле

```
final int SIZE = 10;
```

- По соглашению они полностью пишутся большими буквами

- Слово final и обозначает, что они неизменны

# Тернарный оператор

- Просто сокращенный if — служит для присвоения значения, в зависимости от выражения до знака вопроса:

```
String sex = (isMale) ? "Male" : "Female"; // тернарный оператор
```

# Бонус

ускорение работы в IDEA

# Шаблоны psvm и sout

- psvm + Tab создает точку входа в программу

```
class Main {
    //psvm + Tab
    public static void main(String[] args) {
    }
}
```

- sout + Tab быстро пишет команду вывода в консоль

```
//sout + Tab
System.out.println("ruff-ruff");
```

# Шаблоны itar и iter

- Шаблоны itar и iter позволяют быстро набрать цикл, который пробежит по ближайшему массиву (который мы объявили выше)

```java
int[] numbers = {23, 24, 11, 324234, 233};

System.out.printf("length %d\n", numbers.length);
// itar + Tab
for (int i = 0; i < numbers.length; i++) {
    int number = numbers[i];
    System.out.println(number);
}

// название: for each
// iter + Tab
for (int number : numbers) {
    System.out.println(number);
}
```

# Шаблон souf

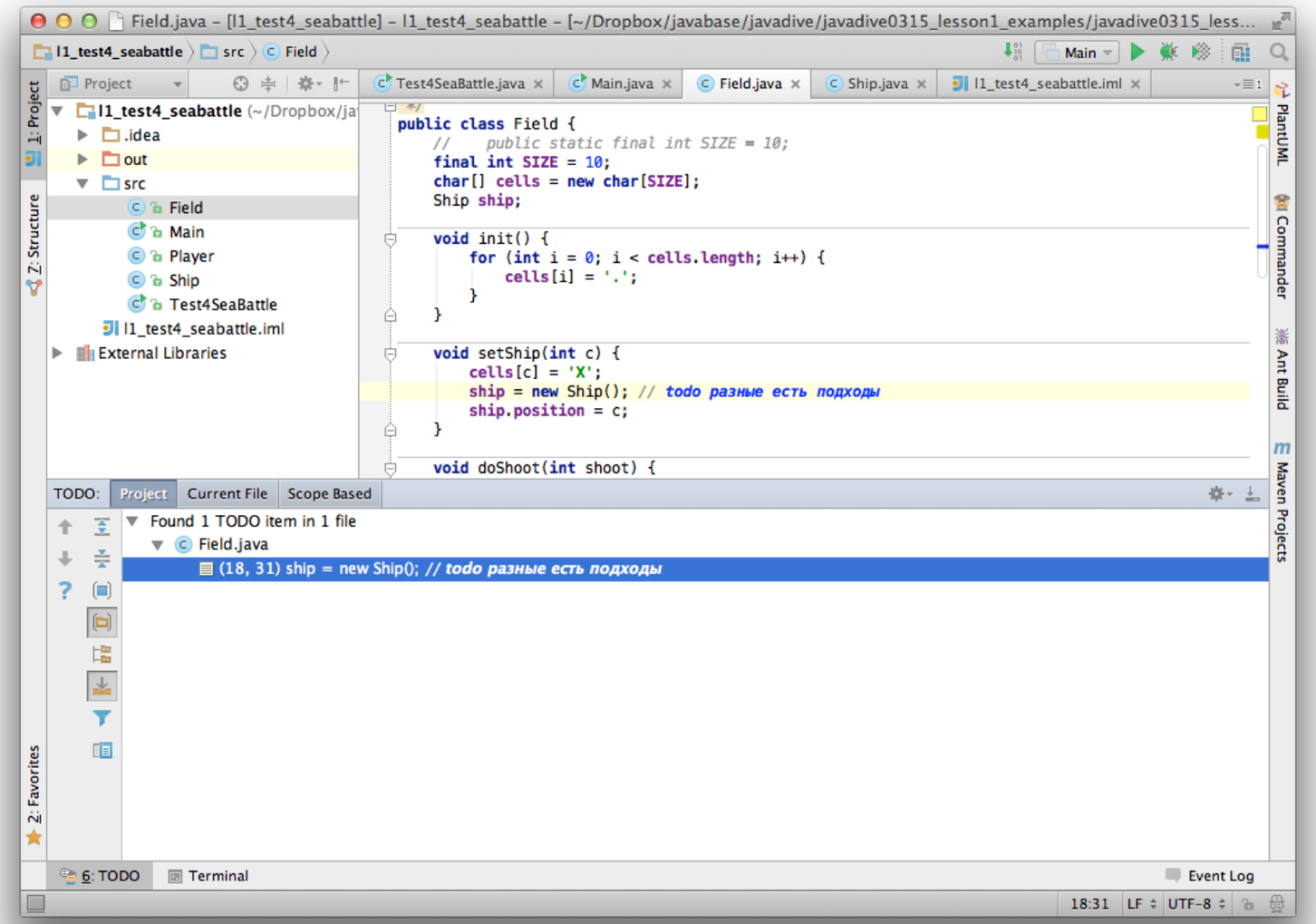
- souf + Tab пишет метод форматированного вывода в консоль

```
// souf
System.out.printf("число %d, строка %s, дробное %.2f", 4, "hello", 45.5);
```

Подробнее: http://programador.ru/printf/

# ТуДушки (// todo)

- Комментарии, которые начинаются с // todo — называются заметками, и в IDEA к ним удобно обращаться из специального окна TODO (сразу видно, сколько заметок мы создали в программе)



# Вспомогательные комбинации в IDEA

- Alt-Ins на Win, Ctrl-Enter на Mac — вызов меню быстрой генерации Конструктора (и др.), если курсор в области кода. А если выделен файл слева в списке файлов, то — вызов быстрого меню создания нового класса

- Shift-Cmd-Enter на Mac, Shift-Ctrl-Enter на Win,— автозавершение if, for или текущего кода (расставляет фигурные скобки и переводит на следующую строчку)

- Уборка кода = в меню Code/Reformat Code или Ctrl-Shift-L

- Ctrl-/ или Cmd-/ -- однострочный комментарий (включает или выключает)

- Ctrl-Y удаление текущей строчки

- Шаблон fori + Tab — просто быстро набирает цикл

- Ctrl-J — документация по текущему методу, на котором стоит курсор