Занятие 4

Anastasiya Solodkaya, Denis Stepulenok

LevelUP

18 апреля 2017 г.

1. Сигнатура метода
2. Классы как типы данных
3. Расположение классов
4. Объекты как переменные
5. Состояние и поведение объекта, поля и методы.
6. Жизнь и смерть объекта
7. Ссылочные типы данных
8. Практика

Что такое сигнатура метода?

- Имя метода
- Параметры метода (в порядке объявления)

Сигнатура точки входа

Сигнатура: main (String[])

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

Возвращаемое значение не входит в сигнатуру!

Примеры сигнатур

```
// m (int, String)
public static int m (int i, String s) {}
// m (String, int)
public static int m (String s, int i) {}
// m (int, String, Object)
public static int m (int i, String s, Object o) {}
```

Возвращаемое значение не входит в сигнатуру!

Классы

- Object
- String
- Integer, Long, Float, Boolean...
- Random
- Scanner

Классы vs Примитивы

Классы vs Примитивы

Классы и примитивы - разные типы, хотя их названия могут быть очень похожи! Примитивы - это не значит, что это что-то "базовое просто они создаются и хранятся совершенно другим образом. Как правило java напрямую в байт-коде пишет их значения. Объекты же нуждаются в указании, как их создать.

Классы vs Примитивы

Класс	Примитив
Long	long
Integer	int
Short	short
Byte	byte
Character	char
Double	double
Float	float
Boolean	boolean

Классы vs Массивы

Классы vs Массивы

Массивы - встроенный тип данных, и его обычно считают видом класса. Да, вы можете создавать массивы объектов (но об этом позже).

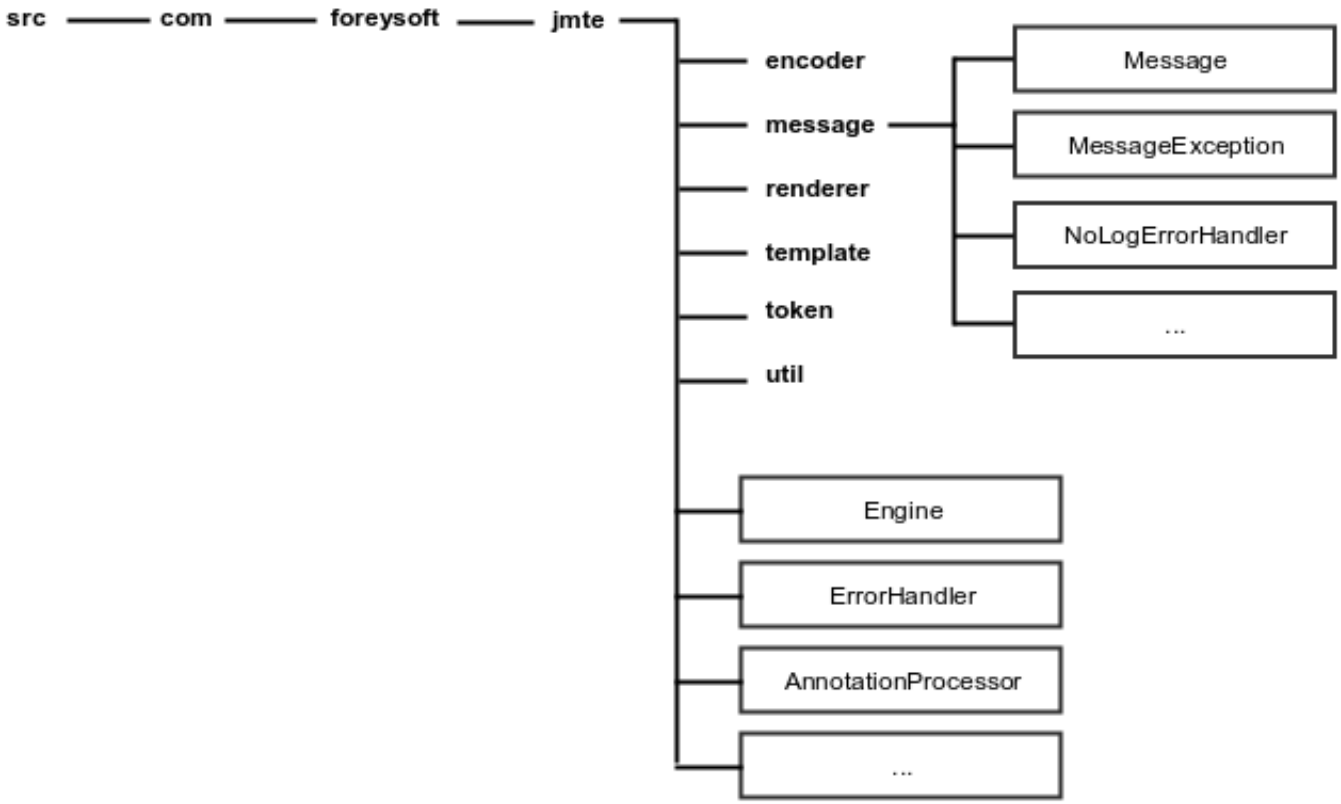
Как устроен исходный код программ Java

- Классы лежат в различных пакетах. Обычно классы разнесены по пакетам на основе какого-то принципа, выбранного разработчиками.
- Обычно есть "главный"пакет, например **com.levelp.beginners**
- Пакеты составляются как доменные имена, здесь важен вопрос уникальности
- Точно так же важен вопрос осмысленности имени!
- Поддерживается вложенность (пакеты вложены один в другой)
- Названия пакетов пишутся обычно с маленькой буквы, желательно не делать их очень длинными
- Обычно они начинаются с "первого"пакета: ru, su, com, org, net, edu, gov
- Если вы не указываете пакет, то это "пакет по умолчанию"

Примеры пакетов из JDK

- **java.util** - коллекции, структуры данных
- **java.math** - математика
- **java.io** - работа с файлами
- **java.nio** - работа с Input/Output

Пример реального приложения



Как указать пакет класса

Допустим, класс лежит в **com.levelp.app.mine**

```
package com.levelp.app.mine;

public class MyClass {
    ...
}
```

Как использовать класс в том же пакете

Допустим, класс лежит в **com.levelp.app.mine**

```
package com.levelp.app.mine;

public class MyClass2 {
    public static void main(String[] args) {
        MyClass c = new MyClass();
    }
}
```

Как использовать класс из другого пакета

Допустим, класс лежит в **com.levelp.csl**

```
package com.levelp.csl;

import com.levelp.app.mine.MyClass;
// Import number of classes from one package:
// import com.levelp.app.mine.*;

public class MyClass3 {
    public static void main(String[] args) {
        MyClass c = new MyClass();
    }
}
```

Как скомпилировать и запустить файл с пакетом?

1. Переходим в директорию, где лежит папка **src**
2. Создаем папку **classes**
3. Выполняем компиляцию, указав папку для скомпилированных классов через опцию **-d**:
   javac -d classes ./src/com/levelp/app/mine/MyClass.java
4. В папке **classes** появился файл ./classes/com/levelp/app/mine/MyClass.class
5. При запуске программы нам необходимо указать то место, где искать наш класс. У нас это класс com.levelp.app.mine.MyClass, то есть java будет искать в com/levelp/app/mine/. То есть место для поиска - папка **classes**:
   java -cp classes com.levelp.app.mine.MyClass

Переменные

Что такое объект?

Объект - это сконструированный экземпляр класса.
Если **класс** - это чертеж, то **объект** - это сделанная по нему табуретка.

Что хранится в переменной?

В любом случае то, что вы записываете как значение переменной (не примитивного типа) - является объектом, а не классом.
Однако, стоит помнить, что существует именно класс **Class**, и у него есть экземпляры.

Объекты

Экземпляры классов:

```
Integer i = new Integer(0); // of class Integer
String s = "abc"; // of class String
Class clazz = String.class; // ! of class Class
```

А что тогда классы?

А классы - это чертежи:

```
public class MyClass {
    ...
}
```

Вы можете заглянуть в исходный код таких классов, как **Integer**, **Long**, **String** и увидеть, как написаны классы. Текст, который вы читаете - и есть класс.

Состояние объекта

- У класса могут быть определены поля, в которые у объекта будут записываться определенные данные
- Эти поля определяют внутреннее состояние объекта

Состояние объекта - табуретка

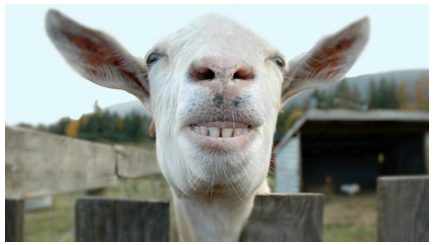
Внутреннее состояние табуретки

- Количество ножек
- Количество перекладин
- Высота
- Ширина седушки
- Длина седушки
- Форма седушки
- Материал седушки

Поведение объекта

- Объект имеет не только внутреннее состояние, но и поведение
- Более того, объекты умеют общаться между собой
- Это общение и поведение обеспечивается методами объекта

Состояние и поведение объекта - животное



Внутреннее состояние животного

- Название
- Имя
- Пол
- Цвет
- Место обитания
- Издаваемые звуки
- Скорость перемещения
- Время сна
- Чем питается
- Впадает ли в спячку?

Поведение?

Ваши варианты?

Поведение животного

- Поесть
- Подать голос
- Бежать
- Спать
- Линять
- Повзрослеть на год
- Умереть

Внутреннее состояние - поля

Очень похожи на переменные. Внимание! Все поля и методы, у которых написано **static** не имеют непосредственного отношения к созданному **объекту**, поэтому не храните в них свойства и данные объекта!

```
public class Animal {
    String name;
    String voice;
    String type;
    int age;
    int color;
    boolean vermint;
}
```

Внутреннее состояние - поля

Получить доступ к полю внутри объекта можно таким образом:

```
public class Animal {
    String name;

    public void run(){
        // two options:
        name = "A";
        this.name = "B";
    }
}
```

Внутреннее состояние - поля

Однако, если параметры метода имеют те же названия, что поля, остается только один вариант (параметры метода перекрывают поле)):

```
public class Animal {
    String name;

    public void updateName(String name) {
        this.name = name;
    }
}
```

Внутреннее состояние - поля

Получить доступ к полю извне объекта можно таким образом:

```
public class Animal {
    String name;
}
public static void main(String[] args) {
    Animal a = ...;
    System.out.println(a.name);
}
```

Однако, обычно так не делают, вместо этого пишут специальные методы для записи в поля.

Поведение - методы

Методы должны иметь разные сигнатуры!

```
public class Animal {
    ...
    void run() { ... }
    String makeSound() { ... }
    int getColor() { ... }
    boolean iVermint() { ... }
    void eat(Object food) { ... }
    void incrementAge() { ... }
    ...
}
```

Поведение - методы

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

    Animal cat = ...;
    Animal mouse = ...;

    cat.makeSound(); // meow!
    cat.run(); // stomping
    cat.eat(mouse);

}
```

Немного реального мира

```
DataStorage storage = ...;
User user = ...;

user.login("john", "123");
long id = user.getId();
System.out.println("User logged in: " + id + ";");

long friendId =
    storage.findFriendByEmail("boris666@gmail.com");
user.sendMessage("Hello!", friendId);

user.updateLastSeenInformation(new Date()); // now
```

Посмотрим еще раз на то, как работают объекты

Но как создать объект?

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

    Animal cat = ...;
    Animal mouse = ...;

    cat.makeSound(); // meow!
    cat.run(); // stomping
    cat.eat(mouse);

}
```

Как рождается объект

- Если вы объявили переменную с определенным классом, у вас еще нет там объекта
- Для создания объекта необходимо использовать ключевое слово **new** и специальный метод - **конструктор**.

```
Integer i = new Integer(5);
```

- Исключение составляют объекты с **autoboxing** - Integer, Long и прочиие. Их можно создать просто через

```
Integer i = 4;
// int temp = 4;
// Integer i = new Integer(temp);
```

Конструктор

- Специальный тип метода. Его имя **всегда совпадает** с именем класса, а возвращаемое значение не указывается:

```
public class Animal {
    public Animal() { // empty constructor
        ...
    }

    public Animal(int age, String name) {
        ...
    }
}

public static void main(String[] args) {
    Animal a = new Animal();
    Animal b = new Animal(1, "John");
}
```

Конструктор

- Класс может содержать несколько конструкторов. У них должна быть разная сигнатура!
- Класс может не содержать конструкторов вообще. В таком случае в нем **автоматически** присутствует конструктор **по умолчанию**, то есть пустой.

Конструктор

- Главная цель конструктора - создать объект и инициализировать его так, чтобы его состояние было целостным
- Если говорить более простыми словами, то он должен задать внутреннее состояние таким образом, чтобы объект был полноценным.
- Например, при рождении котенка у нас нет необходимости задавать имя, зато цвет шерсти, глаз и пол уже определены и должны быть заданы конструктором.

Как умирают объекты?

В java есть garbage collector (сборщик мусора). Он сам определяет, когда объекты больше не нужны (программа не ссылается на них) и удаляет их, очищая память.

Как хранятся объекты в переменных

На самом деле перменные содержат не сам объект (кусочек памяти), а ссылку на эту память.

```
Cat cat0 = new Cat("Barsik");
Cat cat1 = cat0;
Cat cat1 = new Cat("Murka");

cat0 = ?
```

Как передаются аргументы в метод

Что напечатает этот метод?

```
public static void main(String[] args) {
    int i = 10;
    increment(i);
    System.out.println(i);
}

public static void increment(int i) {
    i = i + 2;
}
```

Как передаются аргументы в метод

Ответ: 10

Как передаются аргументы в метод

Теперь посмотрим на объект:

```
public class Cat {
    String name;
    public Cat(String name) {
        this.name = name;
    }
}
public static void main(String[] args) {
    Cat cat = new Cat("Barsik");
    replace(cat);
    System.out.println(cat.name);
}
public static void rename(Cat cat) {
    cat = new Cat("Murka");
}
```

Как передаются аргументы в метод

Ответ: Barsik

Как передаются аргументы в метод

```
public class Cat {
    String name;
    public Cat(String n) {
        name = n;
    }
}
public static void main(String[] args) {
    Cat cat = new Cat("Barsik");
    rename(cat, "Murka");
    System.out.println(cat.name);
}
public static void rename(Cat cat, String n) {
    cat.name = n;
}
```

Как передаются аргументы в метод

Ответ: Murka

Класс - "Круг"

- Для чего мы его создаем?
- Какое название выбрать?
- Какие данные хранить?
- Какие возможности можно добавить?

Circle - данные

- x, y - координаты центра, double.
- r - радиус, double.

Circle - методы

- move(x, y)
- zoom(z)
- isPointInside(x, y)

Зоопарк

- Создайте класс Animal, который умел бы выводить себя на экран, "издавать звуки сообщать, хищный ли он и каков его вес.
- На основе массива создайте свой зоопарк.
- С помощью класса Random выберите случайным образом 10 раз животное и попросите его сообщить все, что он может сообщить о себе.
- Можно добавить свои возможности

TODO-List (проект)

Задачи по проекту

Модифицировать свой проект так, чтобы вместо строк в массиве хранились объекты. Объекты должны содержать данные, которые вам необходимы.