3. Классы

Программирование на Java

Автор материалов курса: Федор Лаврентьев

Лектор: Виктор Яковлев

МФТИ, 2016-2017

Пакеты (packages)

Файл java/lang/Integer.java

```
package java.lang;

public class Integer {
  public static final int MAX_VALUE = ...;
  ...
}
```

Файлы и директории

```
ru/
\-  mipt/
    \- java/
       |-  lection3/
       |   |-  Child.java
       |   |-  Example.java
       |   \-  Parent.java
       \-  lection4/
           \-  Utility.java
java/
|-  io/
|   |-  IOException.java
|   \-  InputStream.java
\-  lang/
    \-  String.java
```

Прямое наследование

Наследование

```
class Rectangle {
  final double width, height;

  Rectangle(double width, double height) {
    this.width = width;
    this.height = height;
  }
}

class Square extends Rectangle {

  Square(double side) {
    super(side, side);
  }
}
```

Приведение и проверка типов

```
public void draw(Shape shape) {
  if (shape instanceof Rectangle) {
    Rectangle rect = (Rectangle) shape;
    drawRectangle(rect);

  } else if (shape instanceof Circle) {
    Circle circle = (Circle) shape;
    drawCircle(circle);

  } else if (shape instanceof Line) {
    drawLine((Line) shape);
  }
}
```

Абстрактные классы

```
abstract class Shape {
  abstract double calcArea();
}

class Rectangle extends Shape {
  ...
  @Override
  double calcArea() {
    return getWidth() * getHeight();
  }
}
```

Переопределение метода

```
class Object {
  public int hashCode() { ... }
}

class Long [extends Object] {
  private final long value;

  @Override
  public int hashCode() {
    return (int) (value ^ (value >>> 32))
  }
}
```

Вызов метода родительского класса

```
class Circle {
  public void draw() {
    Canvas.circle(x, y, radius);
  }
}

class Sun extends Circle {
  @Override
  public void draw() {
    super.draw(); // Draw circle
    drawSunlight();
  }
}
```

Инстанцирование абстрактного класса

```
Shape a = new Rectangle(3, 4);
a.calcArea(); // 12

Shape b = new Square(7);
b.calcArea(); // 49

Shape c = new Shape();
c.calcArea(); // ?
```

Инстанцирование абстрактного класса

```
Shape a = new Rectangle(3, 4);
a.calcArea(); // 12

Shape b = new Square(7);
b.calcArea(); // 49

Shape c = new Shape(); // WRONG!
c.calcArea();
```

Анонимные классы

```
Shape a = new Rectangle(3, 4);
a.calcArea(); // 12

Shape b = new Square(7);
b.calcArea(); // 49

Shape c = new Shape() {
  double calcArea() { return 0; }
}
c.calcArea(); // 0
```

Модификаторы доступа

Инкапсуляция - private и public

```
public class Animal {
  private double x;
  private double y;
  private double bearing;

  public double getX() { return x; }
  public double getY() { return y; }
  public double getBearing() { return bearing; }

  public void rotate(double angle) {
    bearing = modulo(bearing + angle, 2 * Math.PI);
  }

  public void stepUp(double distance) {
    x += distance * Math.cos(bearing);
    y += distance * Math.sin(bearing);
  }
}
```

Точки расширения - protected

```
public class Animal {
  ...
  private boolean hungry;
  public boolean isHungry() { return hungry;}
  protected void setHungry(boolean hungry) {
    this.hungry = hungry;
  }
}

public class LazyAnimal {
  @Override
  public void stepUp(double distance) {
    if (isHungry()) throw new IllegalStateException();
    super.stepUp(distance);
    setHungry(true);
  }
}
```

Доступ по умолчанию (package-local)

```
package edu.phystech.java.lection3;

public class Animal {
  ...
  String name;
}

public class Dog extends Animal {
  public String tellName() { return “Bark!”; }
}

public class Student extends Animal {
  public String tellName() { return name; }
}

public class Zookeeper {
  public String nameAnimal(Animal a) {
    return a.name;
  }
}
```

Хорошая, годная инкапсуляция

```
public class Wallet {
  private double money = 0.0;

  public void putMoney(double amount) {
    if (amount < 0) throw new IllegalArgumentException();
    money += amount
  }

  public void takeMoney(double amount) {
    if (amount < 0) throw new IllegalArgumentException();
    if (amount > money) throw new IllegalArgumentException();
    money -= amount;
  }

  public double getMoney() {
    return money;
  }
}
```

Нарушение инкапсуляции

```
public class Wallet {
  public double money = 0.0;

  public void putMoney(double amount) {
    if (amount < 0) throw new IllegalArgumentException();
    money += amount
  }

  public void takeMoney(double amount) {
    if (amount < 0) throw new IllegalArgumentException();
    if (amount > money) throw new IllegalArgumentException();
    money -= amount;
  }

  public double getMoney() {
    return money;
  }
}
```

Внутренние классы

```
public class Dog {
  private double bearing; // Направление

  public class Head {
    private final int length;
    public String smash() {}
  }
}
```

Вложенные классы

```
public class Observable {

  public static class Event {
    private final Observable source;
    private final String type;
    private final String cause;
  }


  public static class Observer {
    void onEvent(Event e) { ... }
  }
}
```

Множественное наследование

Интерфейсы

```
public interface Planar {
  [public] void setLocation(Location l);
  [public] Location getLocation();
}

public class Rectangle implements Planar {
  @Override
  void setLocation(Location l) {
    this.left = l.getX();
    this.top = l.getY();
  }

  @Override
  void getLocation() {
    return new Location(getLeft(), getTop());
  }
}
```

Абстрактная имплементация интерфейса

```
public interface Planar {
  void setLocation(Location l);
  Location getLocation();
}

public abstract class Shape implements Planar {
  // setLocation() unimplemented
  // getLocation() unimplemented
}

public class Rectangle extends Shape {
  ...
}
```

Наследование интерфейсов

```
public interface Planar {
  void setLocation(Location l);
  Location getLocation();
}

public interface Drawable extends Planar {
  void draw(Canvas c);
}

public class Shape implements Drawable {
  ...
}
```

Множественное наследование

```
public class Geometry {
  ...
}

public class Circle
    extends Geometry
    implements Drawable, Comparable, Closeable {
  ...
}
```

Реализация по умолчанию

```
public interface Observable {
  List<Observer> getObservers();

  void addObserver(Observer o) default {
    getObservers().add(o);
  }

  Observer removeObserver(Observer o) default {
    getObservers().remove(o);
  }

  void notifyObservers(Event e) default {
    List<Observer> observers = getObservers();
    for (int i = 0; i <= observers.size(); ++i) {
      observers.get(i).onEvent(e);
    }
  }
}
```

Сравнение объектов

Сравнение по ссылке

```
Student s1 = new Student(“Vasya”, 21);
Student s2 = new Student(“Vasya”, 21);
Student s3 = s1;

assert s1 == s1; // true
assert s1 == s2; // false
assert s1 == s3; // true
```

Сравнение по ссылке

```
Integer a = 100500;
Integer b = Integer.valueOf(100500);
Integer c = Integer.valueOf(100500);

assert a == b;
assert a == c;
assert b == c;

Integer d = 42;
Integer e = Integer.valueOf(42)

assert d == e;
```

Сравнение по ссылке

```
Integer a = 100500;
Integer b = Integer.valueOf(100500);
Integer c = Integer.valueOf(100500);

assert a == b; // WRONG
assert a == c; // WRONG
assert b == c; // WRONG

Integer d = 42;
Integer e = Integer.valueOf(42)

assert d == e;
```

Сравнение по ссылке

```
Integer a = 100500;
Integer b = Integer.valueOf(100500);
Integer c = Integer.valueOf(100500);

assert a == b; // WRONG
assert a == c; // WRONG
assert b == c; // WRONG

Integer d = 42;
Integer e = Integer.valueOf(42)

assert d == e; // CORRECT?!
```

Сравнение объектов – equals

```
class Student {
  ...
  public boolean equals(Object o) {
    if (o == null) return false;
    if (this == o) return true;
    if (this.getClass() != o.getClass()) return false;

    Student s = (Student) o;
    if (this.age != s.age) return false;
    if (this.name == null) return s.name == null;
    return this.name.equals(s.name);
  }
}
```

Сравнение объектов – equals (в Java 7)

```
class Student {
  ...
  public boolean equals(Object o) {
    if (o == null) return false;
    if (this == o) return true;
    if (this.getClass() != o.getClass()) return false;

    Student s = (Student) o;
    if (this.age != s.age) return false;

    return Objects.equals(this.name, s.name);
  }
}
```

Сравнение объектов – хеш-код

```
class Student {
  ...
  public int hashCode() {
    final int prime = 37;
    return age * prime + name.hashCode();
  }
}
```

Сравнение объектов – по умолчанию

```
public class Object {
  ...
  public native int hashCode();
      // адрес в памяти

  public boolean equals(Object o) {
    return this == o;
  }
}
```

Контракт на сравнение объектов

- a.equals(a) == true; // Рефлексивность
- a.equals(b) == b.equals(a); // Симметричность
- a.equals(b) && b.equals(c) => a.equals(c); // Транзитивность
- a.equals(b) == a.equals(b); //Консистентность
- a.hashCode() == a.hashCode(); //Консистентность
- a.equals(b) == true => a.hashCode() == b.hashCode();
- a.equals(null) == false;