

Принципы проектирования и дизайна ПО

Лекция №3

Агошков Илья 2016

В предыдущих сериях...

В предыдущих сериях...

**Sequence, state machine, activity
diagrams**

Circle, Rectangle, Square

Сценарии использования (Use Cases).

Идентификация классов/объектов и их обязанностей.

Обзор UML диаграмм.

Основы ООП.

Принципы SOLID. High cohesion, loose coupling.

Dependency Inversion Principle, Inversion of Control,

Dependency Injection

Шаблоны GoF (12-14 шаблонов).

Архитектурные стили:

- Client-server, SOA, Event sourcing, Layered Systems, Ports & Adapters (hexagonal architecture), CQRS

Монолитная архитектура и микросервисы.



Abstraction

Polymorphism

Inheritance

Encapsulation

Unit tests

```
public class ArrayMax {  
  
    public static int max(int[] arr) {  
        int max = arr[0];  
        for (int a : arr) {  
            if (a > max) max = a;  
        }  
        return max;  
    }  
}
```

Test 1

@Test

public void

testMaxForArrayWithSingleElement() {

int testArr[] = { 10 };

assertEquals(10, ArrayMax.max(testArr));

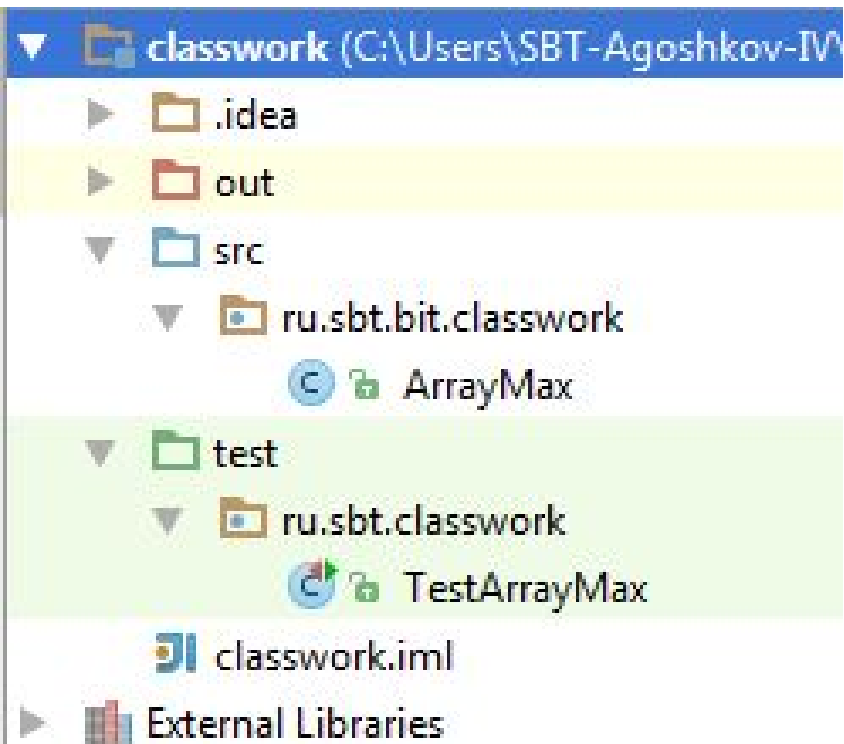
}


Test 2

@Test

```
public void testMaxForArrayWithFewElements()  
{  
    int testArr[] = { 12, 255, -23, 4 };  
    assertEquals(255, ArrayMax.max(testArr));  
}
```


jUnit



 [Back to Dashboard](#)

 [Status](#)


 [Changes](#)


 [Workspace](#)

 [Build Now](#)

 [Delete Project](#)

 [Configure](#)

 [Robot Results](#)

 [Email Template Testing](#)

Project robot



[Workspace](#)



[Recent Changes](#)



[Latest Test Result](#) (no failures)

 Build History	(trend)
 #6	Aug 20, 2013 2:26:16 PM
 #5	Aug 20, 2013 2:24:49 PM
 #3	Aug 20, 2013 2:14:06 PM
 #2	Aug 15, 2013 4:24:27 PM
 #1	Aug 15, 2013 4:21:47 PM
 RSS for all	 RSS for failures

Практика (10-15 минут)

Создать по одному unit-тесту на каждую фигуру.

Test Driven Development

Live coding (Stack)

Практика (10-15 минут)

В TDD стиле начать разрабатывать класс ShapeUtils, функцию max, выбирающую максимальную (по площади) из двух фигур.

Создать класс TestShapeUtils. Создать один тест, создающий две фигуры и вызывающий функцию max. Создать класс ShapeUtils и функцию max через подсказку IDE.

Interface

Интерфейс определяет контракт, без реализации.

```
public interface Shape {  
    double getArea();  
}
```

max

```
public Shape max(Shape a, Shape b) {  
    ...  
}
```

```
public interface Collection<E> extends Iterable<E> {  
    int size();  
    boolean isEmpty();  
    boolean contains(Object o);  
    Iterator<E> iterator();  
    Object[] toArray();  
    <T> T[] toArray(T[] a);  
    boolean add(E e);  
    boolean remove(Object o);  
    boolean containsAll(Collection<?> c);  
    boolean addAll(Collection<? extends E> c);  
    boolean removeAll(Collection<?> c);  
    boolean retainAll(Collection<?> c);  
    void clear();  
    boolean equals(Object o);  
    int hashCode();  
}
```


Практика (10 минут)

Доделать ShapeUtils.max с использованием интерфейса.

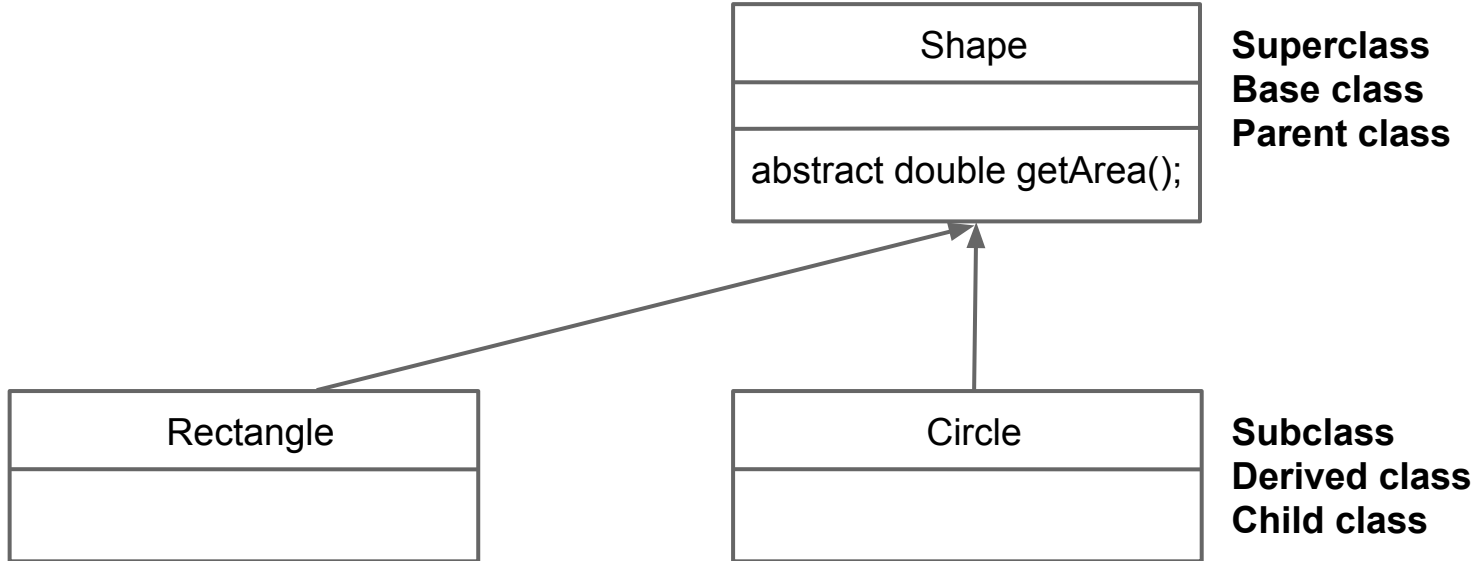
Inheritance

```
public abstract class Shape {  
    public abstract double getArea();  
}
```

Inheritance

```
public class Circle extends Shape {  
    private final double radius;  
    public Circle(double r) {  
        this.radius = r;  
    }  
    @Override  
    public double getArea() {  
        return Math.PI * radius * radius;  
    }  
}
```

Inheritance



Circle inherits from Shape

Problems with inheritance?

3rd party libraries

code from other package/module

Polymorphism

$a + b$

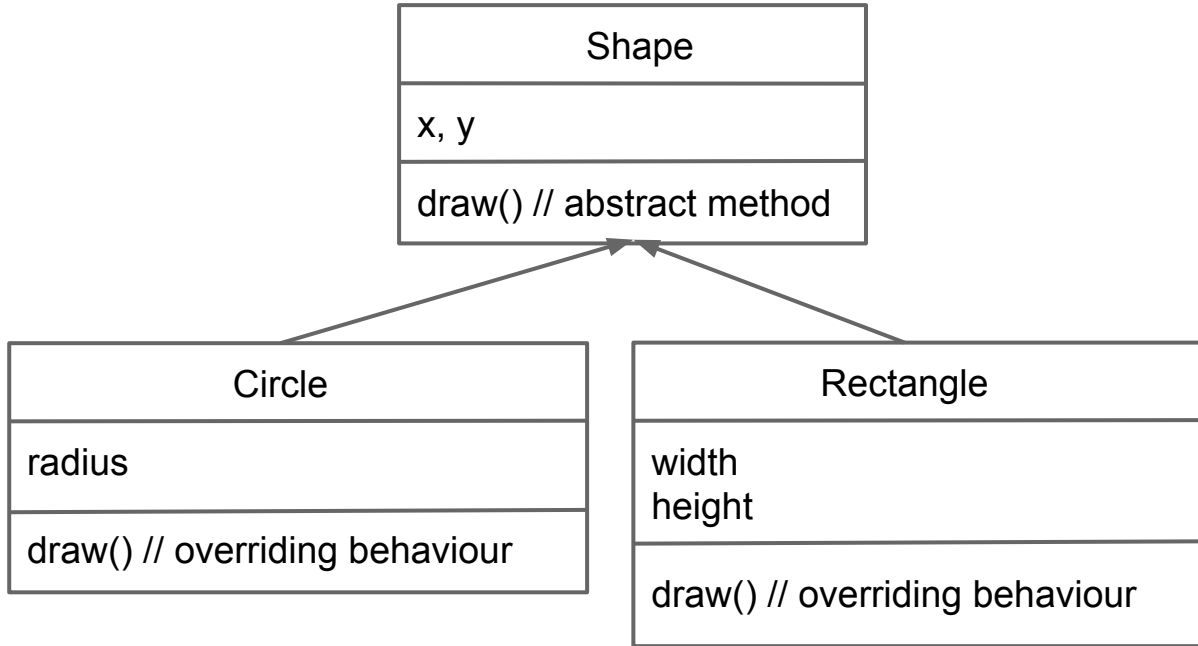
$3 + 5 = 8$

Polymorphism

$a + b$

“Hello ”+”World”=”Hello World”

Polymorphism



Polymorphism

```
public static Shape max(Collection<Shape> shapes) {  
    Shape max = shapes.isEmpty() ? null :  
                                   shapes.iterator().next();  
  
    for (Shape shape : shapes) {  
        if (shape.getArea() > max.getArea()) max = shape;  
    }  
    return max;  
}
```

Polymorphism

```
public static Shape max(Collection<Shape> shapes) {  
    Shape max = shapes.isEmpty() ? null :  
                                   shapes.iterator().next();  
  
    for (Shape shape : shapes) {  
        if (shape.getArea() > max.getArea()) max = shape;  
    }  
    return max;  
}
```

Polymorphism

```
public static Shape max(Collection<Shape> shapes) {  
    Shape max = shapes.isEmpty() ? null :  
                                shapes.iterator().next();  
  
    for (Shape shape : shapes) {  
        if (shape.getArea() > max.getArea()) max = shape;  
    }  
    return max;  
}
```

Практика (10-15 минут)

В TDD стиле реализовать Comparator для сортировки коллекции объектов типа Shape через `Collections.sort(list, comparator)`.

Начать с `assertEquals`.