Android Development

OOP with Java

SoftUni Team

Teodor Kostadinov

Software University

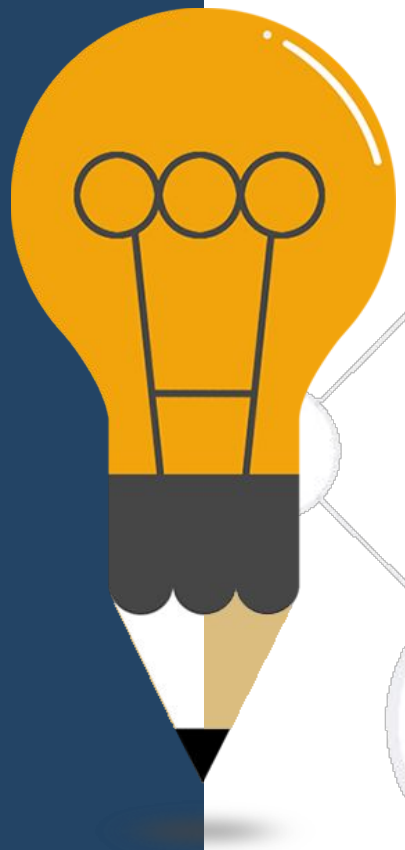
SoftUni Foundation

Software University

http://softuni.bg

Table of Contents

Just a second!



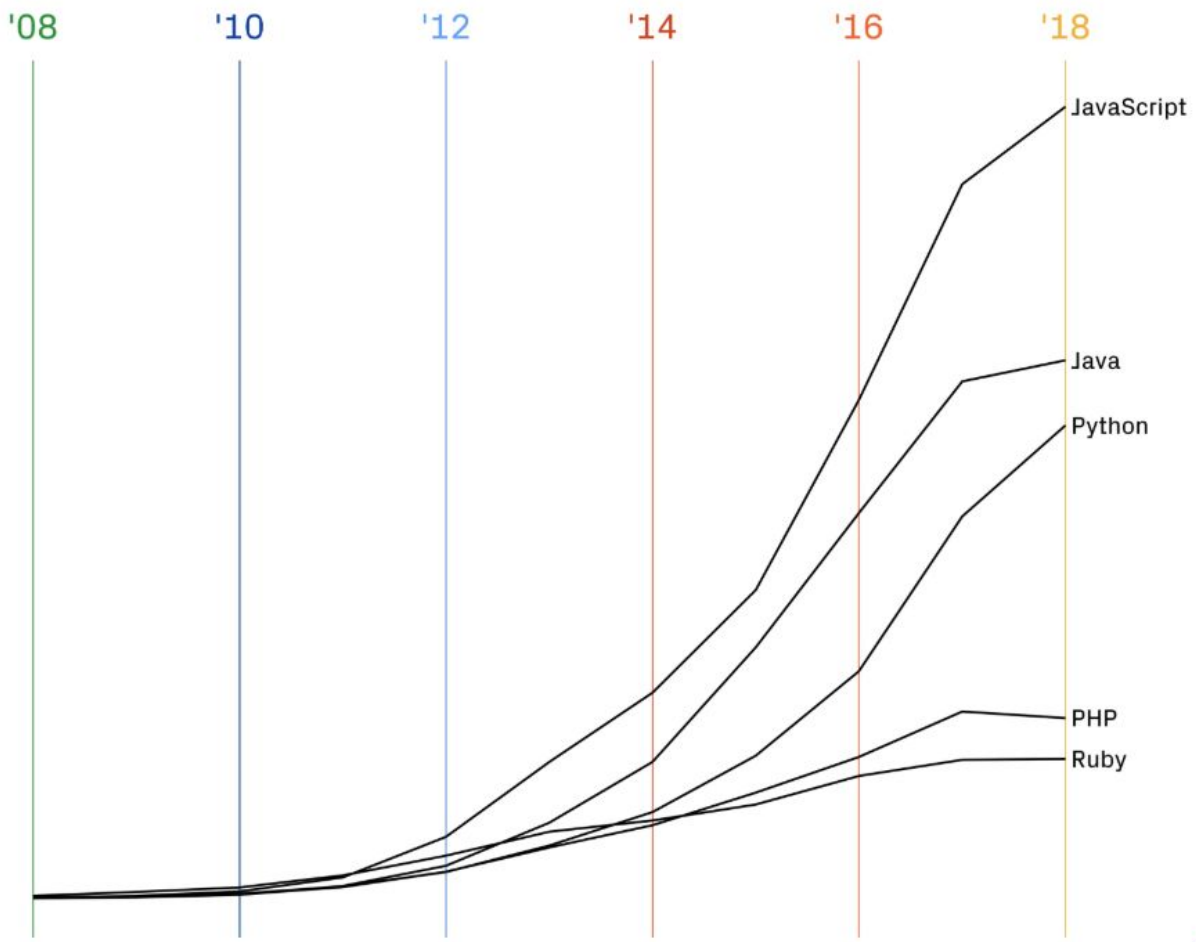
What is your OOP knowledge?

Please fill this form: http://bit.ly/oop-and-you

This will determine the level of detail we go into when explaining code constructs in this course.

Java rules the World

- One of the most popular programming languages
- Robust and enterprise-ready
- Can be used for almost everything



Source: https://github.blog/2018-11-15-state-of-the-octoverse-top-programming-languages/

What about Kotlin?

- The future of Android
- JVM language
- Some call it the next generation of Java
- Can have both Java and Kotlin files in same project

		Growth in contributors
1	Kotlin	2.6x
2	HCL	2.2x
3	TypeScript	1.9x
4	PowerShell	1.7x
5	Rust	1.7x
6	CMake	1.6x
7	Go	1.5x
8	Python	1.5x
9	Groovy	1.4x
10	SQLPL	1.4x

Source: https://github.blog/2018-11-15-state-of-the-octoverse-top-programming-languages/

How Java code looks?

Software University

```java
public class MyClass extends MyBaseClass implements MyInterface {
    public static final int SOME_CONSTANT = 42;
    public int publicField;
    private static MyClass sSingleton;
    int mPackagePrivate;
    private int mPrivate;
    protected int mProtected;
    XmlHttpRequest request;

    public int doSomeStuff() {
    }
}
```

```kotlin
class MyClass : MyBaseClass(), MyInterface {
    var publicField: Int = 0
    internal var mPackagePrivate: Int = 0
    private val mPrivate: Int = 0
    protected var mProtected: Int = 0
    internal var request: XmlHttpRequest? = null

    fun doSomeStuff(): Int {}

    companion object {
        val SOME_CONSTANT = 42
        private val sSingleton: MyClass? = null
    }
}
```

Kotlin Corner

Java 101

- Java is strongly-typed language
- Primitive types: byte, short, int, long, float, double, boolean, char
- Methods
  - can return a value
  - void - do not return anything
- Other important classes:
  - String - for text
  - arrays - initialized with []
  - Collections - ArrayList, HashMap, etc
- Constructs
  - loops - for, while
  - if and switch statements

Source: https://www.tutorialspoint.com/java/java_quick_guide.htm

Kotlin is:

- statically typed
- compatible with Java
- no primitive types
- variables can't be null if not explicitly stated so
- variables should be initialized when created
- for, while, if, when

Kotlin Corner

Java Style Guide

- Camel Case
- Variable names start with lower-case letter
- Class names are nouns and start with upper-case letter
- Method names are verbs with lower-case letter
- Constants are all upper-case letters
- Open brackets on the same row
- Leave comments with //

Source: https://google.github.io/styleguide/javaguide.html

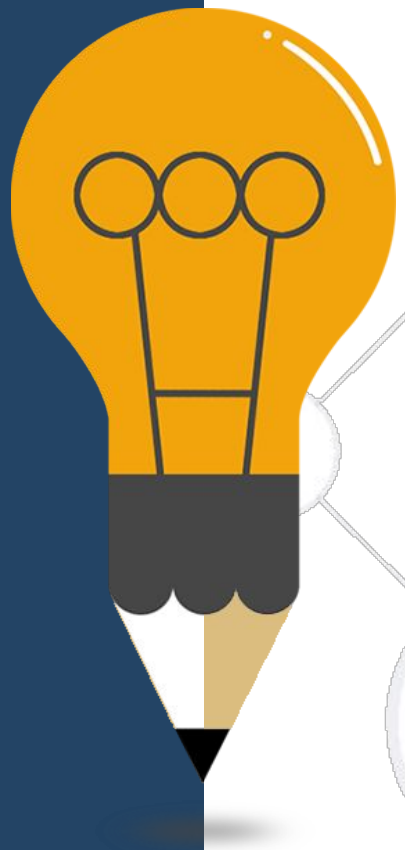
Kotlin style guide:
https://kotlinlang.org/docs/reference/coding-conventions.html

Kotlin Corner

Demo

Let’s write our first? (hope not) Java program

Let's do this!



Write the following program:

Given 2 int values, return whichever value is nearest to the value 10, or return 0 in the event of a tie. Note that Math.abs(n) returns the absolute value of a number.

(8, 13) → 8

(13, 8) → 8

(13, 7) → 0

Write it here: https://codingbat.com/prob/p172021

OOP Time

- OOP is a way to structure your code
- It tells you how to divide code in different files (classes)
- It tells you how to make these different files (classes) work together
- It tells you how to make your code secure, easy to change, easy to bugfix
- It helps you reuse your code and not duplicate it
- It gives general guidelines so everyone reading your code can easily understand it (and do it fast)

Source: http://www.introprogramming.info/intro-java-book/readonline/glava11-sazdavane-i-izpolzvanena-obekti/

OOP Time

- OOP has 4 main principles
- Encapsulation - don’t allow outside forces to change you
- Abstraction - don’t tell exactly what your object is
- Inheritance - allow one object to pass its features to its kids
- Polymorphism - make so that you can do the same thing over different objects

Source: http://codebetter.com/raymondlewallen/2005/07/19/4-major-principles-of-object-oriented-programming/

Everything is Object in Java

```
class Chair {
    String material = “wood”;
    int positionX = 5;
    int positionY = 4;
    Color matColor = Color.RED;
    void moveChair ( int x, int y){
        System.out.println(“Chair moved”);
    }
}
```

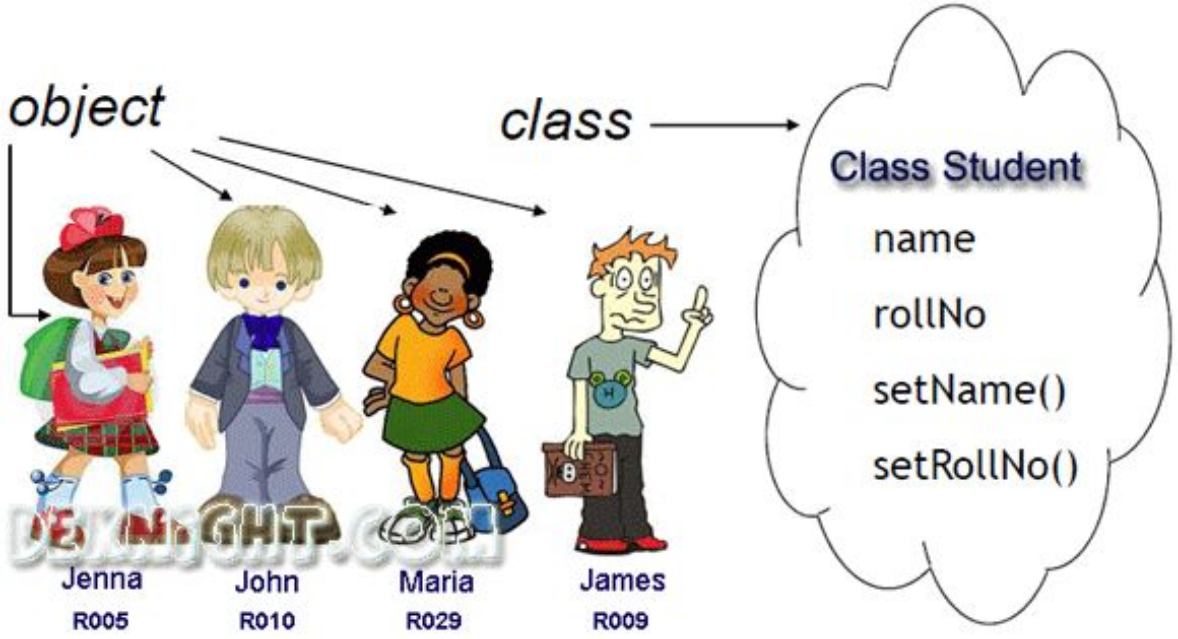
```
class Chair {
    internal var material = "wood"
    internal var positionX = 5
    internal var positionY = 4
    internal var matColor: Color = Color.RED

    internal fun moveChair(x: Int, y: Int) {
        println("Chair moved")
    }
}
```

Kotlin Corner

Classes

- A class is just a mold, which can be used to create objects.
- An object is a specific element, created using a class. It is also called an instance of the class.

```
Student pesho = new Student();
Chair zelen = new Chair();
zelen.color = Color.Green;
```



Classes

- A class can contain:
  - fields
    - static or not, or constants
  - methods
    - static or not
  - constructors

```
internal class Chair {
    var material = "wood"
    fun moveChair(x: Int, y: Int) {
        println("Chair moved")
    }
    init {
        print("A new chair was created")
    }
    companion object {
        var matColor: Color = Color.RED
    }
}
```

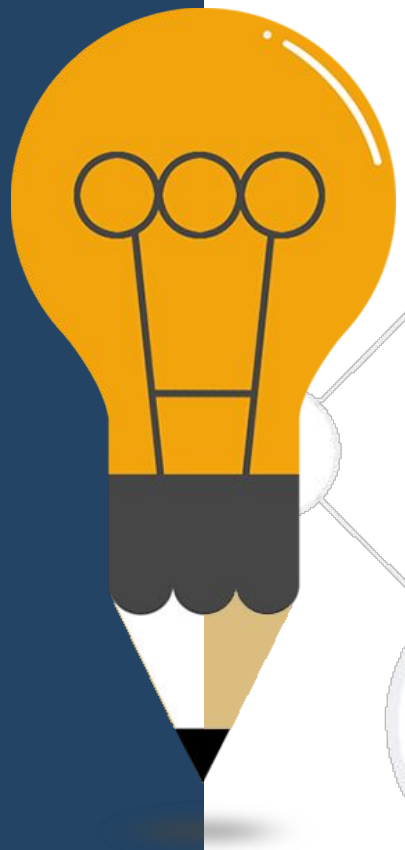
Kotlin Corner

```
class Chair {
    String material = “wood”;
    static Color matColor = Color.RED;
    void moveChair ( int x, int y){
        System.out.println(“Chair moved”);
    }
    Chair() {
        System.out.print(“A new chair was created”);
    }
}
```

Demo

Let’s create a dog shelter in Java

Let’s do this!



Write the following classes:

Circle - it should have a radius and center coordinates, getArea and getPerimeter methods

Rectangle - you decide what to have

Privacy

- in front of each field and method we can specify who can access it

Modifier	Class	Package	Subclass	World
public	✓	✓	✓	✓
protected	✓	✓	✓	✗
no modifier	✓	✓	✗	✗
private	✓	✗	✗	✗

Inheritance

- A class can inherit another class
- It gets all the fields and methods of its father
- A child-class can decide to overwrite (override) some of the methods of its father

```
class Human {
  int age;
  void getOld() {
      this.age++;
  }
}
```

```
class Student extends Human {
  int schoolNumber;
  void doHW(){
      this.age++;
  }
}
```

```
internal class Student : Human() {
  var schoolNumber = 0
  fun doHW() {
    this.age++
  }
```

Kotlin Corner

Abstraction

- Can be achieved with interfaces
- Interfaces are like promises, you know what the class can do, but you don't know what class it is exactly

```
interface CanBark {
  void bark();
}
```

```
class Dog implements CanBark {
  @Override
  void bark(){
      //bau bau
  }
}
```

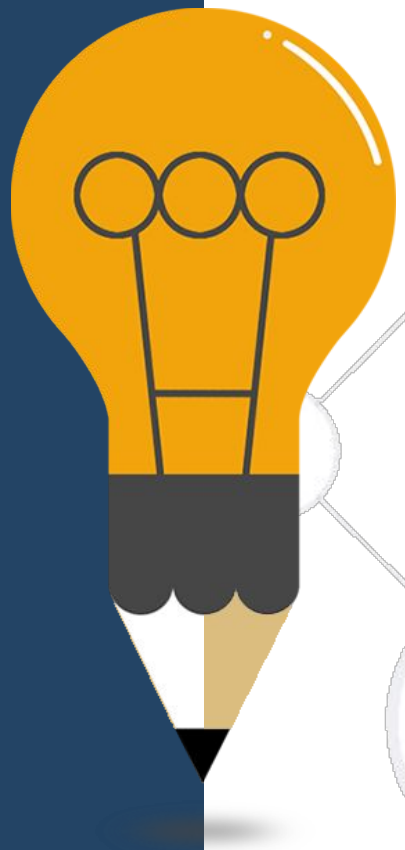
```
internal class Dog : CanBark {
  override fun bark() {
    //bau bau
  }
```

Kotlin Corner

Demo

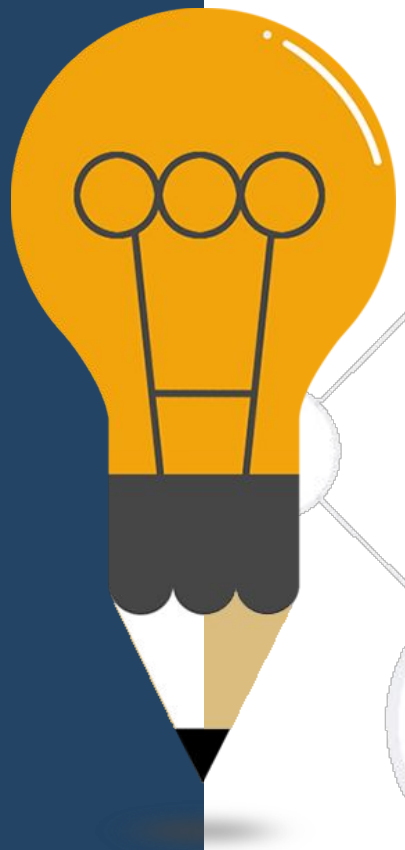
Let’s improve our dog shelter

Test Time



Let's together answer the following questions and understand why things are the way they are.

Test Time



```
class Triangle{
    public int base;
    public int height;
    private static double ANGLE;
    public static double getAngle();
    public static void Main(String[] args) {
        System.out.println(getAngle());
    }
}
```

Identify the one correct statements:

A. It will not compile because ANGLE cannot be private.

B. It will not compile because getAngle() has no body.

C. It will not compile because it does not implement setAngle method.

D. It will not compile because ANGLE field is not initialized.

E. It will not compile because of the name of the method Main instead of main.

Test Time

```
public int setVar(int a, int b, float c) {
    ...
}
```

Which two of the following methods correctly overload the above method?

```
A. public int setVar(int a, float b, int c){return (int)(a + b + c);}
B. public int setVar(int a, float b, int c){return this(a, c, b);}
C. public int setVar(int x, int y, float z){return x+y;}
D. public float setVar(int a, int b, float c){return c*a;}
E. public float setVar(int a){return a;}
```

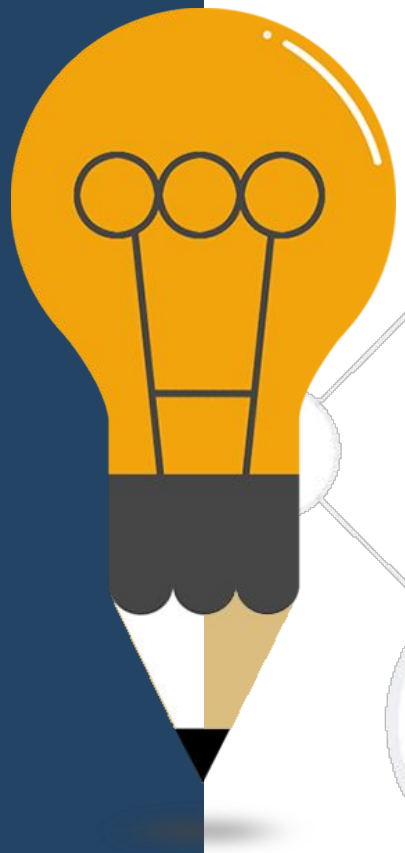
Test Time

```
interface MyIface{};
class A {};
class B extends A implements MyIface{};
class C implements MyIface{};

and the following object instantiations:
A a = new A();
B b = new B();
C c = new C();
```

Which of the following assignments are legal at compile time?

A. b = c;
B. c = b;
C. MyIface i = c;
D. c = (C) b;
E. b = a;

Test Time



```
public class TestClass{
public TestClass(int a, int b) { } // 1
public void TestClass(int a) { } // 2
public TestClass(String s); // 3
private TestClass(String s, int a) { } //4
public TestClass(String s1, String s2) { } //5
}
```

Which lines contain a valid constructor in the following code?

Select 3 options

Test Time

What will the following code print when compiled and run?

```
class ABCD {
int x = 10;
static int y = 20;
}

class MNOP extends ABCD {
int x = 30;
static int y = 40;
}
```

```
public class TestClass {

public static void
main(String[] args) {

System.out.println(new
MNOP().x+", "+new
MNOP().y);

}

}
```

Test Time

Which of the following statements is/are true?

A. Subclasses must define all the abstract methods that the superclass defines.
B. A class implementing an interface must define all the methods of that interface.
C. A class cannot override the super class's constructor.
D. It is possible for two classes to be the superclass of each other.
E. An interface can implement multiple interfaces.

Test Time

```
public class TestClass{

    class MyException extends Exception {}

    public void myMethod() throws XXXX{
        throw new MyException();
    }
}
```

What can replace XXXX?
Select 3 options

A. MyException
B. Exception
C. No throws clause is necessary
D. Throwable
E. RuntimeException

Test Time

```
public class TestClass {

    static String str = "Hello World";

    public static void changeIt(String s){
        s = "Good bye world";
    }
    public static void main(String[] args){
        changeIt(str);
        System.out.println(str);
    }
}
```

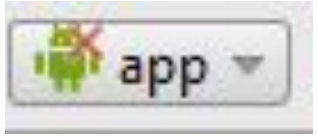
What will the following program print?

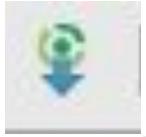
Select 1 option
A. "Hello World"
B. "Good bye world"
C. It will not compile.
D. It will throw an exception at runtime.
E. None

Android Studio

- The IDE for Android Development made by Google
- Shortcuts
  - ctrl + space -> auto-complete
  - alt + enter -> automatic fixes
  - ctrl + alt + L -> rearrange code
  - ctrl + F -> search in file
  - alt + F9 -> find usage
  - alt + insert -> insert getters/setters/constructors
  - shift + F6 -> rename

Android Studio





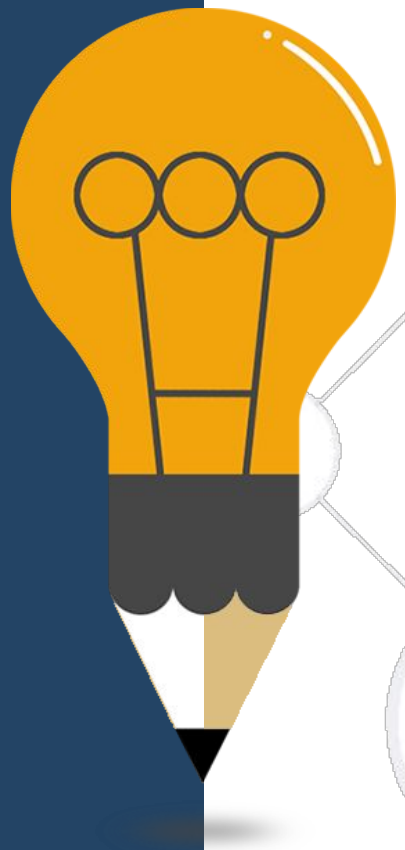
- билдва наново проекта
- конфигурация за изпълнение
- пуска проекта в нормален или дебъг режим
- синхронизира gradle файла
- създава и контролира виртуалните устройства - емулаторите
- sdk manager - отговаря за обновяването на андроид библиотеките и другите компоненти

Summary and Resources

OOP is just some rules to order and divide your code.

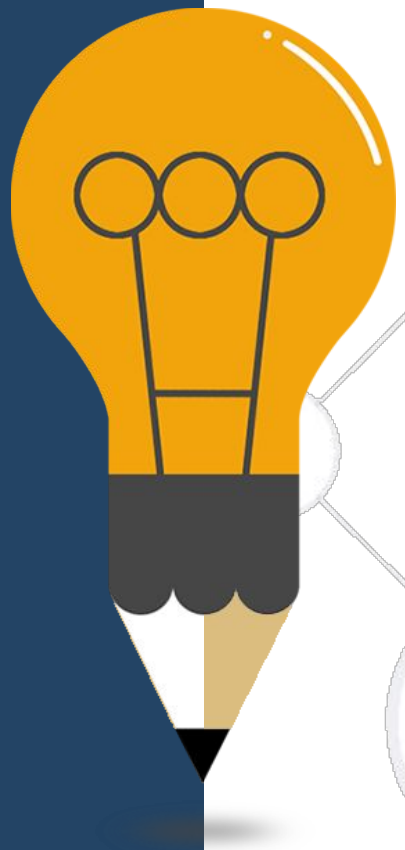
Resources:
http://www.headfirstlabs.com/books/hfjava/
https://source.android.com/source/code-style.html

Homework (1)



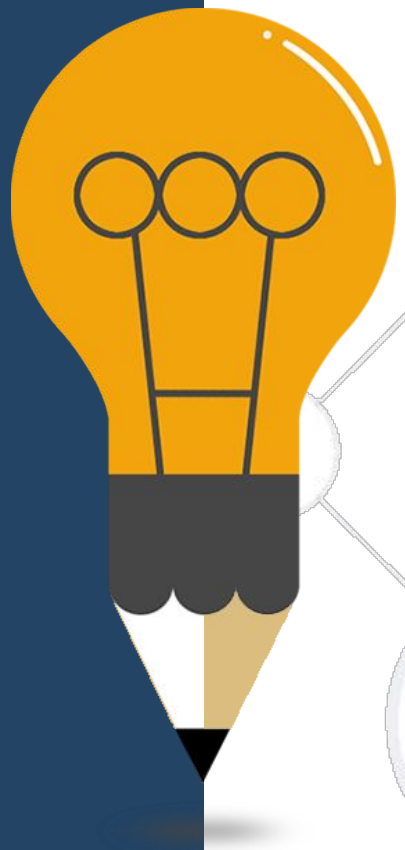
- Extend the Date class in Java
- Add a check if the day is a Bulgarian holiday or not
- Test it

Homework (2)



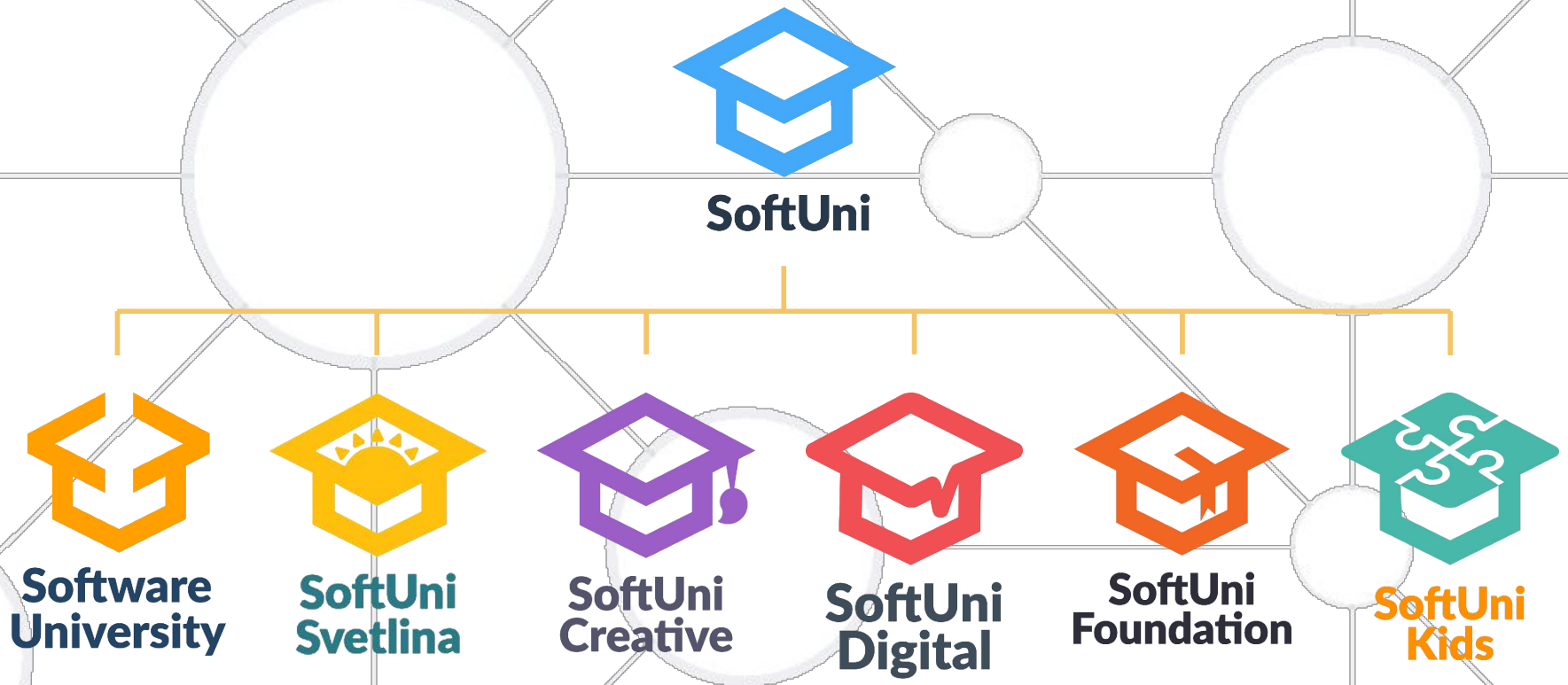
- Create a library. Define classes for a library and for books. The library should have a name and a list of books. Books should have title, author, publisher, year, ISBN.
- You should be able to add a book
- You should be able to search a book by author or title
- You should be able to see the details of a book
- You should be able to delete a book
- Create a test class that initializes a library object, adds books to it, shows information for all books. Then search for a book and show results, delete a book and again show information about all books.

Homework (3)



- Install Android Studio
- Create an empty project
- Run it on your phone or in an emulator

Questions?


SoftUni
Software University
SoftUni Svetlina
SoftUni Creative
SoftUni Digital
SoftUni Foundation
SoftUni Kids

СофтУни диамантени партньори

Software University

INDEAVR
Serving the high achievers

INFRAGISTICS®

SoftwareGroup
doing it right

xssoftware

NETPEAK

SUPER HOSTING .BG®

СофтУни диамантени партньори

Software University

SBTech we know sports

telenor

codexio

SmartIT

LIEBHERR

æternity

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg