Building Blocks

Class Structure

Classes

- classes are basic building blocks of every Java program
- to design a class means to describe parts and characteristics of these blocks
- in order to use a class you need to create *an object* (most of the times)
    - you can think about a class as a blueprint, and an object as realization
- an *object* is a single representation of the class, also called *instance* of a class
- a *reference* is a variable that points to an object

Fields and Methods

- two main elements (members) of Java classes are *fields* and *methods*
- fields are sometimes referred to *variables*
  - to be precise: all fields are variables, but not all variables are fields
- fields hold the information about the *state* of an object or a class
- methods describe some action or operation on that state
  - methods are similar to *functions* in some older programming languages
- let's write some code...

```java
// simplest Java class
class Student { }

// in file Student.java
public class Student {
```
return type
```java
  String name;
  public String getName() {
    return name;
    }
```
signature
```java
  public void setName(String theName) {
    name = theName;
  }
}
```

Comments

- comments are used to make a code more readable
  - they are ignored by the compiler
- there are three ways to comment out the text
  - comment until the end of a line: `//`
  - comment everything within `/*` and `*/`
  - comment starting with `/**` (Javadoc)

```
// this is one-line comment

System.out.println(a); // this will print a

System.out.println(a); /* this will print a */


/*

 * usual way to write multiline comments

 * it's very readable like this

*/


/**

 * Javadoc style offers you some nice features

 * @author Luka Popov

*/
```

Classes and source files

- it's a good practice to have each class in it's own .java file
- it's possible to have more classes in one file
    - but only one of them is *top-level class*
- top-level class is almost always marked as `public`, but it's not necessary
- if you mark the top-level class with `public`, then the filename must match the class name
- only one class in the can be marked as `public`

```
// in file Student.java
public class Student { }


// in file Item.java
public class Item { }
class SomeOtherItem { }


// in file Customer.java
public class Customer { }
public class Client { } // DOES NOT COMPILE
```