Lecture 5

Introduction to Strings

Object Oriented Programming

Contents

- Introduction to Strings
- Object Oriented Programming (OOP)
- Classes and objects
- Fields
- Manipulating object state
- Using methods

Strings

- What is String?
- How to create String

```
String firstName;

firstName = "Ivan";

String lastName = new String("Petrov");
```

Declare variable of type String

Initialization

Another way for initialization

Concatenation of strings

+ is used for concatenation

```
String firstName = "Ivan";
String lastName = "Petrov";

String name = firstName + " " + lastName;

System.out.println(name);
```

Prints the value of name to the console

Comparing strings

.equals() should be used because String is reference type

```
String firstName = "Ivan";
String lastName = "Petrov";

String name = firstName + " " + lastName;

System.out.println(name == "Ivan Petrov");
System.out.println(name.equals("Ivan Petrov"));
```

The output is:

```
false
true
```

More about Strings

- \ should be used for escaping special characters
- .length() return the length of the string
- String has many features(methods) for manipulating the text value

```
String welcome = "Welcome to learning center \"SoftAcad\"";

System.out.println(welcome.length());
```

The output is 37

Object Oriented Programming

- OOP is concept in programming
- It enable software engineers to write reusable, easy for understanding and maintaining code
- The heart of OOP consist of objects and classes

Objects

- Software objects are used to model the real-world and abstract objects that you find in everyday life
- Real-world objects share two characteristics: They all have state and behavior

Each person has name, age, personal number... (state)

Each person can eat, sleep, walk... (behavior)

Mobile phone – Have memory, has color, is switched on or off. Can ring, can send SMS, can be switched off

Classes

Main idea

- The class acts as the template for building object
- The class defines the properties of the object and its behavior

Person example

Every human:

- Has name
- Has age
- Has personal number
- Has sex
- Has weight

Person example

Ivan

- 25 years old
- p.n. 8612025281
- is male
- 80.5 kg

Maria

- 21 years old
- p.n. 8203301201
- is female
- 55.0 kg

Writing simple classes

- Each starts with *class <name of the class>*
- The properties are called fields. They hold the state of each object
- The fields has type and name

```java
public class Person {
    String name;
    int age;
    long personalNumber;
    boolean isWoman;
    double weight;
}
```

Class name

Fields

Objects in Java

- Objects are the presentation of a class
- Each class can have more than one object instances
- Objects of same classes have the same properties, but they may differ by the values of these properties
- Objects exists in heap memory
- Objects can be created and their state can be changed

Creating objects of class Person

- A variable of type Person should be declared
- Objects are created via constructors (we'll talk more about them in the next lesson)
- Using keyword *new*

```java
public class PersonTest {

    public static void main(String[] args) {
        Person ivan = new Person();
        Person maria = new Person();
    }

}
```

Differences between classes and objects

- Object is the concrete representation of a class.
- Class is the „model“ for creating an object
- Each object has the properties that its class owns
- Objects have the same properties, but they may differ by the values of these properties
- One class can have more than one objects, but an object can't be instance of more than one classes

More on classes

- Each class begins with a capital letter and use CamelCase convension
- Each class has the same name as the file it is declared in
- The programmer creates the classes in a file .java, Java compiles .java-files and creates .classes
- .java is human-readable, .class is machine-readable

Accessing fields and modifying the state of the object

<object>.<fieldname> is used to access fields

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

    Person ivan = new Person();
    ivan.name = "Ivan";
    ivan.age = 25;
    ivan.isWoman = false;
    ivan.personalNumber = 861202528;
    ivan.weight = 80.5;

    System.out.print("Ivan is " + ivan.age + " years old ");
    System.out.print("and his weight is " + ivan.weight);

}
```

Accessing field with .

Car Example

Let's write class which represents Car

Each car has:

- Model
- Max speed
- Current speed
- Color
- Current gear

Car Example

1. Write the class Car

2. Create class CarDemo with main method

3. Create 2 instances of class car and set values to their fields

4. Change the gear and current speed of one of the cars

Car owner

We want every car to have owner.
The owner is a person.

1. Make some changes to class Car to assign owner to every car

2. In CarDemo set owner to one of the objects of type Car and print to the console the owner's name and owner's age.

Add friend to class Person

Each person has a friend, who is a person as well.

Friend is a field of type Person in class Person.

No problem for a class to have and instance of itself

Methods

- Methods are features of the object
- Can manipulate the data of a specified object
- Can perform any other task
- Have name
- Have body, enclosed between braces { } - code
- Have parameters
- Have return type (for now we'll use only void)

```
<return type> <method name> (<parameters>) {
  <body>
}
```

Methods in class Person

Each human eat food, can walk, can drink water and increase his age every year.

- eat ()
- walk()
- growUp() - modify the field age
- drinkWater(double liters)

Methods in class Person

```java
public class Person {
    String name;
    int age;
    long personalNumber;
    boolean isWoman;
    double weight;

    void eat() {
        System.out.println("Eating...");
    }
    void walk() {
        System.out.println(name + " is walking");
    }
    void growUp() {
        age++;
    }
    void drinkWater(double liters) {
        if(liters > 1) {
            System.out.println("This is too much water!!!");
        } else {
            System.out.println(name + " is drinking " + liters + " water.");
        }
    }
}
```

Return type

Method name

Parameter

Calling methods

(non static) methods are called by instance of the class using .

<instance>.<method name>(<parameters list>);

```
public static void main(String[] args) {
    Person ivan = new Person();
    ivan.name = "Ivan";
    ivan.age = 25;
    ivan.isWoman = false;
    ivan.personalNumber = 861202528;
    ivan.weight = 80.5;

    ivan.walk();
    double literWater = 0.3;
    ivan.drinkWater(literWater);
}
```

Exercise

Add methods in class Car:

```
void accelerate()

void changeGearUp()

void changeGearDown()

void changeGear(int nextGear)

void changeColor(String newColor)
```

Write logic in methods which change gear

(validate the gear before changing - min is 1, max is 5)

Invoke them in CarDemo class

Methods in class Car

```java
void changeGearUp() {
    if(gear < 5) {
        gear++;
    }
}
void changeGearDown() {
    if(gear > 0 ) {
        gear--;
    } else {
        System.out.println("You are now on 1st gear!!!);
    }
}
void changeGear(int nextGear) {
    if(nextGear > 0 && nextGear < 6) {
        gear = nextGear;
    }
}
void changeColor(String newColor) {
    color = newColor;
}
```

Calling the methods of class Car

```java
public static void main(String[] args) {
     Car golf = new Car();
     golf.speed = 100;
     golf.color = "Red";
     golf.gear = 5;
     golf.maxSpeed = 320.5;

     Car honda = new Car();
     honda.gear = 5;
     honda.changeGearUp();

     System.out.println("The current speed of the golf is " + golf.speed);
     golf.accelerate();
     System.out.println("The current speed of the golf is " + golf.speed);

     System.out.println("The current gear is " + golf.gear);
     for (int i = 0; i < 10; i++) {
          golf.changeGearUp();
     }
     System.out.println("The current gear is " + golf.gear);

     System.out.println("The Honda's current gear is " + honda.gear);
     honda.changeGear(1);
     System.out.println("The Honda's current gear is " + honda.gear);

     golf.changeColor("Blue");
     golf.changeColor("Red");
}
```

Summary

- What is String and how we can to use it?
- What is a class?
- What is an object?
- What's the differences between classes and object
- How to declare property of a class
- Use objects as fields
- How to create an object
- How to declare and call methods