Java Course Lecture 3 - Class, fields and methods



www.pragmatic.bg

Contents



- What's a class
- What's an object
- Fields
- Methods
- Packages



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

Example



Every human:

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

Person example

Peter

- 30 years old
- p.n. 8612025281
- is male
- 100 kg

Maria

- 35 years old
- p.n. 8203301201
- is female
- 53.0 kg

www.pragmatic.bg

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

```
public class Person 4
                                     Class name
     String name;
     int age;
     long personalNumber;
     boolean isWoman;
                                           Fields
     double weight;
```

www.pragmatic.bg

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
- Using keyword new

```
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

www.pragmatic.bg PRAGMATIC IT Learning &

Outsourcing Center

More on classes

- Each class begins with a capital letter and use CamelCase convention
- 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 .class file
- .java is human-readable, .class is virtual-machinereadable

www.pragmatic.bg PRAGMATIC IT Learning &

Outsourcing Center

Accessing fields and modifying the state of the object

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

```
public static void main(String[] args) {
     Person ivan = new Person();
      ivan.name = "Ivan";
      ivan.age = 25;
                                             Accessing
      ivan.isWoman = false;
                                           field with .
      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);
```

Car Example



Let's write class which represents Car

Each car has:

- Max speed
- Current speed
- Color
- Current gear

Car Example



- 1. Write the class Car
- 2. Create class CarDemo with main method
- Create 2 instances of class car and set values to their fields
- Change the gear and current speed of one of the cars

www.pragmatic.bg PRAGMATIC IT Learning &

Outsourcing Center

Car driver/owner

- We want every car to have owner.
- The owner is a <u>person</u>
- 1. Make some changes to class Car to assign owner to every car
- 2. In CarDemo print to the console the name of the owner for every car n.

www.pragmatic.bg PRAGMATIC IT Learning & Outsourcing Center

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.
- There is no problem for a class to have an instance of itself

www.pragmatic.bg

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>
```

www.pragmatic.bg

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

PRAGMATIC A IT Learning & **Outsourcing Center**

```
public class Person {
                                      Method
        String name;
        int age;
                                                           Return
                                       name
        long personalNumber;
                                                            type
        boolean isWoman;
        double weight;
        void eat()
          System.out.println("Eating...");
        void walk() {
          System.out.println(name + " is walking");
                                                                      Parameter
        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.");
```



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

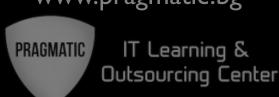


Add methods in class Car:

```
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



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

Calling the methods of class



```
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");
```

www.pragmatic.bg

this key word



- **this** key word is used to reference the current instance.
- this this is always referring to instance that the operation at hand was requested

Constructors



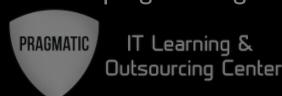
- A special "method" that is called when creating a new instance of a class
- Constructors have the same name as the class. That is mandatory!
- A constructor initializes an object immediately upon creation
- Constructors can have arguments just like any other method
- Constructors have no return type, not even void

www.pragmatic.bg PRAGMATIC IT Learning &

Outsourcing Center

Default Constructor

- A default constructor is a constructor with no arguments.
- All classes have default constructors
- If one is not defined, the JVM will create one for you



this() constructor

- Since constructors are similar to methods it is perfectly correct for one constructor to refer to another constructor of the same class.
- A constructor invokes another constructor of the same class by using the this() constructor



Packages



- The idea behind packages is to group classes together.
- Usually packages are named based on the following convention:
- prominentdomain.domainname.project.functionality all lowercase with no spaces
- Example: edu.pragmatic.domain
- Containing classes for university actors of our program like Professor, Student and so on.

Visibility



- Usually a Class may use ALL classes in the current package of that Class without doing anything.
- When a class needs to use another class defined in different package there are 2.5 options 69
 - Import statement (like we did for the Scanner class)

```
import edu.pragmatic.domain.Student;
```

From now on you may use class Student as it was in the current package

```
import edu.pragmatic.domain.*;
```

From now on you can use ALL classes in this package Inline when you need it just type

```
edu.pragmatic.domain.Student s = new
edu.pragmatic.domain.Student()
```

Summary



- 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

Problems



- What's a class ?
- What's the difference between class and objects?
- Name all manners in which a class is instantiated?
- What's my favourite color?
- Where in the memory are objects stored?
- What's the connection between the class and the java file it resides?
- What's an instance ?
- What's the difference between a method and a constructor?
- How can we access a property of a class if we have an object of that class?
- Can we use this() in a method?
- Can we call methods within a constructor?
- Can we call methods within other methods?
- Write a small program which models a school. Write the program in a way that whomever uses your classes, they will be able to create a new classroom, a new course. Set the teacher and students for that course.