Java

Loops and Object Oriented Programing

Christoph Baitis

2. November 2022

Overview

1. Loops
   - for
   - while
2. OOP in Java
   - General information
   - Methods
   - Return Value
   - Constructor

Loops

for

```
for(initial value, condition, change) {
    // do code while condition is true
}
```

for example

```java
public class ForExample {

    public static void main(String[] args) {
        for(int i = 0; i <= 10; i++) {
            System.out.print("na ");
        }
        System.out.println("BATMAN!");
    }

}
```

while

```
while(condition) {
    // do code while condition is true
}
```

while example

```
1  public class WhileExample {
2
3      public static void main(String[] args) {
4          int a = 0;
5          while(a <= 10) {
6              System.out.println(a);
7              a++; // Otherwise you would get an endless loop
8          }
9      }
10
11 }
```

Try it yourself

Your first JAVA game!

Write a game which first calculates a random number and lets the user guess the number afterwards.

The user should be promoted to enter a number

- if the number is larger than the random number the program should output *"To large!*
- if the number is small than the random number the program should output *"To small!*
- if the number is the random number the program should output *"You got it!*
- also output the number of tires the user took to guess the number

```
1 //Generating a random number between a min and a max value
2 int min = 5
3 int max = 10;
4 int random = ((Math.random() * (max - min)) + min);
```

OOP in Java

Object Oriented Programming

Class Student

```java
public class Student {

    // Attributes
    private String name;
    private int matriculationNumber;


    // Methods
    public void setName(String name) {
        this.name = name;
    }

    public int getMatriculationNumber() {
        return matriculationNumber;
    }

}
```

Creation

We learned how to declare and assign a primitive datatype.

```
1          int a; // declare a
2          a = 273; // assign 273 to a
3
```

The creation of an object works similar.

```
1          Student example = new Student();
2          // create an instance of Student
3
```

The **object** derived from a **class** is also called **instance**. The variable is called the **reference**.

Calling a Method

```
1   public class Student {
2
3       private String name;
4
5       public String getName() {
6           return name;
7       }
8
9       public void setName(String newName) {
10          name = newName;
11      }
12
13  }
14
```

The class *Student* has two methods: *String getName()* and *void setName()*.

Calling a Method

```java
public class Main {

    public static void main(String[] args) {
        Student example = new Student(); // creation
        example.setName("Jane"); // method call
        String name = example.getName();
        System.out.println(name); // Prints "Jane"
    }

}

```

You can call a method of an object after its creation with **reference.methodName();**.

Calling a Method

```
1   public class Student {
2
3       private String name;
4
5       public void setName(String newName) {
6           name = newName;
7           printName();   // Call own method
8           this.printName(); // Or this way
9       }
10
11      public void printName() {
12          System.out.println(name);
13      }
14
15  }
16
```

You can call a method of the own object by simply writing **methodName();** or **this.methodName();**

Methods with Arguments

```
1  public class Calc {
2
3      public void add(int summand1, int summand2) {
4          System.out.println(summand1 + summand2);
5      }
6
7      public static void main(String[] args) {
8          int summandA = 1;
9          int summandB = 2;
10         Calc calculator = new Calc();
11         System.out.print("1 + 2 = ");
12         calculator.add(summandA, summandB);
13         // prints: 3
14     }
15
16 }
17
```

Methods with Return Value

A method without a return value is indicated by **void**:

```
1     public void add(int summand1, int summand2) {
2         System.out.println(summand1 + summand2);
3     }
4
```

A method with an **int** as return value:

```
1     public int add(int summand1, int summand2) {
2         return summand1 + summand2;
3     }
4
```

Calling Methods with a return value

```
1   public class Calc {
2
3       public int add(int summand1, int summand2) {
4           return summand1 + summand2;
5       }
6
7       public static void main(String[] args) {
8           Calc calculator = new Calc();
9           int sum = calculator.add(3, 8);
10          System.out.print("3 + 8 = " + sum);
11          // prints: 3 + 8 = 11
12      }
13
14  }
15
```

Constructors

```
1   public class Calc {
2
3       private int summand1;
4       private int summand2;
5
6       public Calc() {
7           summand1 = 0;
8           summand2 = 0;
9       }
10
11  }
12
```

A constructor gets called upon creation of the object

Constructors with Arguments

```
1   public class Calc {
2
3       private int summand1;
4       private int summand2;
5
6       public Calc(int x, int y) {
7           summand1 = x;
8           summand2 = y;
9       }
10
11  }
12
```

```
1   [...]
2   Calc myCalc = new Calc(7, 9);
3
```

A constructor can have arguments as well!

Try it yourself - 1

Task

Models a cube as a class.
Cubes have the following attributes:

- Side length

Cubes have the following methods:

- one that calculates the volume
- one that calculates the surface area

Write a program that creates an object of type Cube. It should output which volume and which surface area is calculated by the cube object.

Task (advanced)

- write a function "cutSideInHalf" which cuts the sidelength of the cube in half
- write a new class Cuboid whicht has the same methods as the cube class but has a baseSideLength and a Height

References

```
1  public class Reference {
2      public static void main(String[] args) {
3          int var1 = 4;
4          int var2 = var1;
5
6          var1 = 8;
7          System.out.println(var2);
8
9          Number ref1 = new Number(4);
10         Number ref2 = ref1;
11         ref1.number = 8;
12         System.out.println(ref2.number);
13 }
14 }
```

Task

- Write the class number that matches the source code above.
- Run the program. What do you observe? What can you conclude from it?

That's it!

- Be encouraged to keep working on the tasks
- Feel free to reach out
  - to send your results
  - to tell me about problems you ran into

Next lesson

- Visibilities
- Arrays
- Inheritance
- more practical examples!

Self study

For those of you who are bored because you can finish the tasks much quicker than others challenge yourself!

You can find other tasks at
`https://ein-christoph.github.io/java-tud/self-study.html`

Caution!
The tasks you'll be confronted with in INLOOP are for people already knowing the basics of Java. Do not worry if you can not solve the tasks right now. You will be able to after this course!