Object Oriented Programming

Computer Science
project

Jakub Piotr Hamerliński, M.Eng. | 2022-10-24

Before we start

Let’s clean up repository’s structure

+

Basic Java introduction

Repository

Structure

1. Repository name: **oop-put-course**
2. For each lesson create a dir:
   lesson-<*number*>
3. Each exercise should be named:
   ex-<*number*>**.**<*extension*>
4. Create a **README.md** file with your **student id**

Object Oriented Programming

Java class example

```
final class Dog {
  int weight;
  String name;
  String breed;
}
```

Object Oriented Programming

Java object example

```
Dog loyalFriend = new Dog();
loyalFriend.name = "Hachikō";
loyalFriend.weight = 13;
loyalFriend.breed = "Akita Inu";
```

Object Oriented Programming

Java method example

```
final class Dog {
  (...)
  Ball dogsBall() {
    return ball;
  }
}
```

Object Oriented Programming

Java Hello world example

```
public class HelloWorld {

  public static void main(String[] args) {
    System.out.println("Hello world");
  }

}
```

Object Oriented Programming

Agenda

1. Following conventions
2. OOP glossary - part II
    2.1. Constructors
3. Exercises

Object Oriented Programming

Method naming convention

Methods divided into two groups:

- builders,

- manipulators.

Builders are **nouns** and manipulators are **verbs**.

Object Oriented Programming

Method naming convention

```
int length(…);
String parsedText(…);
void save();
void print();
```

Object Oriented Programming

Conventions

google.github.io/styleguide/javaguide

google.github.io/styleguide/cppguide

Object Oriented Programming

Constructors

It’s an entry point to a new object.

It accepts some arguments, does something with them, and prepare the object to perform their duties.

Properly constructed class should have many constructors and only a few methods.

Number of constructors > **Number of methods**

Object Oriented Programming

Constructors

```
new Money(420);
new Money(“15.22€”);
new Money(150.69f);
new Money(9.99, “PLN”);
```

```
Money defaultCash;
Money floatCash(140.15f);
Money stringCash("128");
Money intCash(144);
```

Object Oriented Programming

Constructors

Constructor's main task is to initialise encapsulated properties, using the provided arguments.

One primary constructor with initialisation.

Many secondaries constructors, which call the primary one.

This approach reduces complexity and helps to avoid duplication.

Object Oriented Programming

Constructors

```java
public class Money {
  private int amount;
  Money(float mny) {
    this((int) mny);
  }
  Money(String mny) {
    this(Integer.parseInt(mny));
  }
  Money(int mny) {
    this.amount = mny;
  }
  int balance() {
    return amount;
  }
}
```

```cpp
class Money {
 private:
  float amount;

 public:
  Money() { this->amount = 0.0f; }
  Money(float mny) { this->amount = mny; }
  Money(int mny) { this->amount = mny; }
  Money(std::string mny) { this->amount = std::stof(mny); }
  float balance() { return amount; }
};
```

Object Oriented Programming

Constructors

Constructs must be code-free.

Any computations done inside a constructor is a bad practice, because they can create side effects that are not requested by the object owner.

Object Oriented Programming

Constructors

```java
interface Name {
  String firstName();
}

public class EnglishName implements Name{
  private final CharSequence text;
  public EnglishName(final CharSequence txt) {
    this.text = txt;
  }
  @Override
  public String firstName() {
    return this.text.toString().split("", 2)[0];
  }
}
```

```cpp
class Name {
  virtual std::string firstName() = 0;
};

class EnglishName : public Name {
 private:
  std::string text;

 public:
  EnglishName(std::string txt) { this->text = txt; }

 public:
  std::string firstName() {
    return text.substr(0, text.find(" "));
  }
};
```