Poznan University of Technology

Object Oriented Programming

Jakub Piotr Hamerliński, M.Eng.

Object Oriented Programming

Agenda

- ~~getters, setters~~
- Methods naming convention
- Encapsulation
- Task

Before we start

Object Oriented Programming

Some basic rules

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

~~getters, setters~~

"It is convenient to have these ~~getters~~, you may say. We are all used to them. There are many examples, where ~~getters~~ are being actively used. This is not because they are so effective. This is because we are so procedural in our way of thinking. We do not trust our objects. We only trust the data they store."

Yegor Bugayenko

Object Oriented Programming

~~getters, setters~~

Problems they create:

- **Violated Encapsulation Principle**
- **Exposed Implementation Details**
- **Telling Instead Of Asking**

Object Oriented Programming

Exposed Implementation Details

One of the biggest sins of the OOP.
Usage of ~~getters~~, ~~setters~~ leads to data structures/bags, not to classes.
Bad example below:

```
struct Cash {
  int dollars;
}
printf("Cash value is %d", cash.dollars);
```

Object Oriented Programming

Exposed Implementation Details

How we should do it:

```java
public class Cash {
  private int dollars;
  Cash(int mny) { this.dollars = mny; }
  int balance() { return dollars; }
  public static void main(String[] args) {
    Cash cash = new Cash(69);
    System.out.println("Cash value is " + cash.balance());
  }
}
```

Object Oriented Programming

Violated Encapsulation Principle

~~getters & setters~~ break encapsulation.
Bad examples below:

```
int t;
t = 85;
printf("The temp is %d F", t);
```

```
class Temperature {
  private int t;
  public int getT() { return this.t; }
  public void setT(int t) { this.t = t; }
}
```

Object Oriented Programming

Violated Encapsulation Principle

How we should do it:

```java
class Temperature {
  private int t;
  public String toString() {
    return String.format("%d F", this.t);
  }
}
```

Object Oriented Programming

Telling Instead Of Asking - remediation

Use printers instead of ~~getters~~.
How we should do it:

```java
public class Book {
  private final String isbn =
    "0735619654";
  private final String title =
    "Object Thinking";
  public String toJSON() {
    return String.format(
      "{\"isbn\":\"%s\", \"title\":\"%s\"}",
      this.isbn, this.title
    );
  }
}
```

Methods naming convention

Object Oriented Programming
Methods naming convention

Methods divided into two groups:

- builders,
- manipulators.

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

Booleans are exception - they are builders which are **adjectives**.

Object Oriented Programming

Methods naming convention

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

boolean empty();
boolean equalTo();
boolean present();
```

Encapsulation

encapsulation (noun, BrE /ɪnˌkæpsjuˈleɪʃn/):
the act of expressing the most important parts of something in a few words,
a small space or a single object

Oxford Dictionary

Object Oriented Programming

Encapsulation

Proper class should encapsulate no more than 4 properties.

```
public class Book {
  private String isbn;
  private String author;
  private String title;
  private Integer pages;
  (…)
 }
```

```
public class Book {
  private ISBN isbn;
  private Author[] author;
  private Title title;
  private Integer pages;
  (…)
 }
```

Object Oriented Programming
Task

Write a **class (choose one from examples)** in Java, which will encapsulates correct (~4) amount of properties. If your class need more properties, encapsulates other objects instead of plain types. Each class should have at least one constructor and at least one sensible, well named, method.

Class examples: 1) Movie, 2) Vehicle, 3) Book, 4) Animal, 5) Plant, and 6) Game.

Questions?

Fin