Poznan University of Technology

Object Oriented Programming

Jakub Piotr Hamerliński, M.Eng.

Object Oriented Programming

Agenda

- Exception type
- Project structure
- Introduction to unit testing
- Task

Exception type

Object Oriented Programming

Exception

Even though we have many different exception, I recommend to use only one type, the most generic, **Exception**, **std::exception**

```
public void pay(double repayment) throws Exception;
```

```
CreditDebt debt = new CreditDebt(period, firstPay, amount);
try {
  debt.pay(1000.0);
} catch (Exception ex) {
  System.out.print("Unable to pay debt: " + ex.getMessage());
}
```

Project structure

Object Oriented Programming
Project structure

Project structure Java

```
└── src
    ├── Application.java
    └── pl
        └── poznan
            └── put
                ├── content
                │   ├── Character.java
                │   ├── Content.java
                │   ├── FileContent.java
                │   └── Symbol.java
                └── log
                    ├── Logarithm.java
                    └── Number.java
```

Object Oriented Programming

Project structure

Project structure C++

```
├── include
│   ├── content
│   │   ├── Content.h
│   │   └── FileContent.h
│   └── log
│       ├── Logarithm.h
│       └── Number.h
└── src
    ├── Application.cpp
    ├── content
    │   └── FileContent.cpp
    └── log
        └── Logarithm.cpp
```

Introduction to unit testing

Object Oriented Programming

Introduction to unit testing

In testing world it's popular to use mocks for unit testing. One of the flaws of mocking is that these tests became verbose, and because of that, the maintainability is decreased. I think this approach is bad and instead of mocks, we should use fakes. Fake is a class that mimic behavior of the original one.

```java
public class Cash {
  private final Exchange exchange;
  private final float amount;
  public Cash(Exchange exch, float amnt) {
    this.exchange = exch;
    this.amount = amnt;
  }
  public Cash exchangedCash(String currency) {
    return new Cash(this.exchange, (this.amount * this.exchange.rate("USD", currency)));
  }
  @Override
  public String toString() {
    return Float.toString(amount);
  }
}
```

Object Oriented Programming

Introduction to unit testing

```java
public interface Exchange {
  float rate(String origin, String target);
  final class Fake implements Exchange {
    @Override
    public float rate(String origin, String target) {
      return 1.2345f;
    }
  }
}
```

Object Oriented Programming

Introduction to unit testing

```java
public class Application {

  public static void main(String[] args) {
    Exchange exchange = new Exchange.Fake();
    Cash dollar = new Cash(exchange, 500);
    Cash euro = dollar.exchangedCash("EUR");
    assert "617.25".equals(euro.toString());
  }
}
```

Object Oriented Programming

Task

Create class Weather, which encapsulates interface Forecast, which declares method returning todays temperature. Create Fake class inside Forecast, which will return some, always the same, value. Show usage of the class Weather.