



AARHUS UNIVERSITET

# **Software Engineering and Architecture**

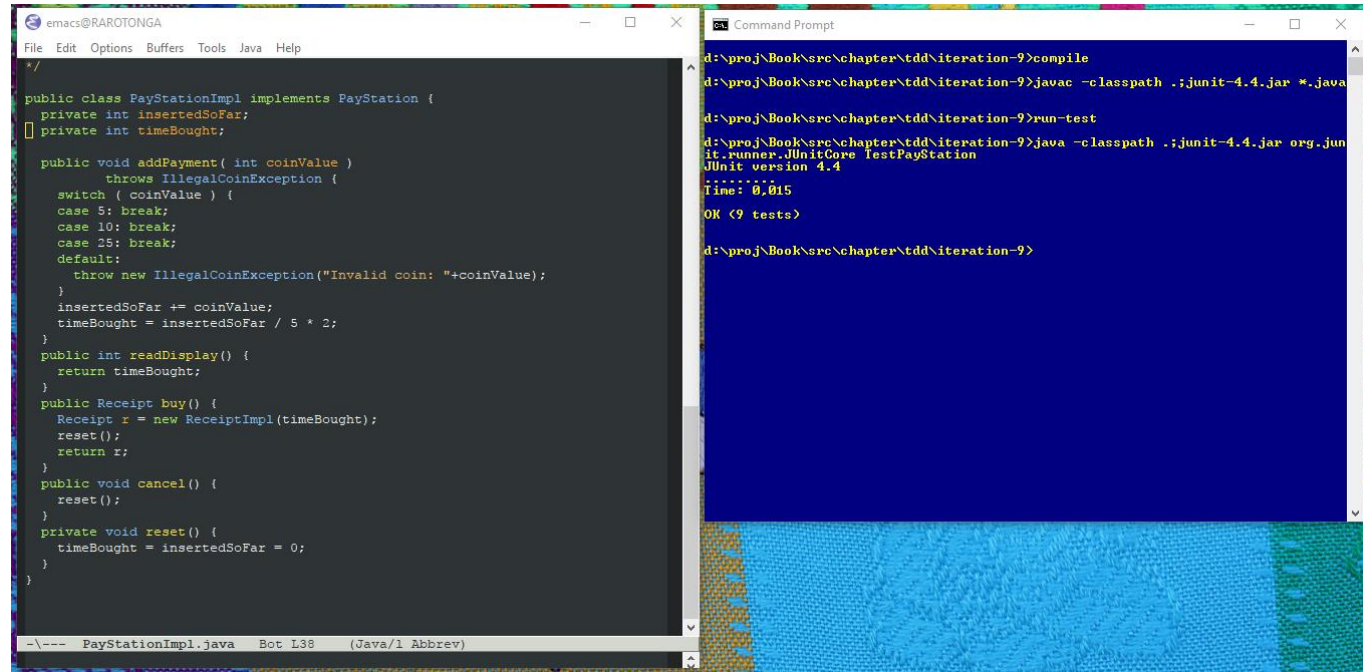
Integrated Development Environment

IntelliJ



# Looking Back in Time

- All you actually need is
  - An editor
  - A compiler



# However...

- Modern IDE's have a deep understanding of the programming language that helps a lot in development
  - Context sensitive help and **completion**
  - **Refactoring** support, especially *renaming identifiers*
  - Integration with **SCM tools**
  - Strong **browsing** of large package/source file hierarchies
- And of course things that *others* 😊 use
  - Debugger
- (Why do I almost never use a debugger?)
  - Because I take very small steps!
    - If I cannot figure a defect out, a few *temporary* System.out does it 😊

The screenshot displays the IntelliJ IDEA IDE interface. The main editor window shows the file `TestPayStation.java` with the following code:

```
1 package paystation.domain;
2
3 import ...
4
5 /** Testcases for the Pay Station system.
6
7 This source code is from the book
8 "Flexible, Reliable Software:
9 Using Patterns and Agile Development"
10 published 2010 by CRC Press.
11 Author:
12 Henrik B Christensen
13 Computer Science Department
14 Aarhus University
15
16 This source code is provided WITHOUT ANY WARRANTY either
17 expressed or implied. You may study, use, modify, and
18 distribute it for non-commercial purposes. For any
19 commercial use, see http://www.baerbak.com/
20 */
21
22 public class TestPayStation {
23
24     /** No a test of the paystation, just an example of
25     what matches the hamcrest library has... */
26     @Test
27     public void shouldDefinitelyBeRemoved() {
28         assertThat("This is a dummy test", containsString("dummy"));
29     }
30 }
31
```

The Run tool window at the bottom shows the execution of the test `TestPayStation.shouldDefinitelyBeRemoved`, which passed successfully. The output indicates that the process finished with exit code 0. A green notification bubble at the bottom center of the IDE states "Tests Passed: 1 passed".




AARHUS UNIVERSITET

# Highlights in *Power Coding*

Past the 'Run All Tests' right-click

- 'Quick Fix' programming
  - Write immediately what you want
  - Let IDE fill in the details under your guidance **'Alt+Enter'**

```
@Test
public void shouldShow2MinFor5CentEntered() throws IllegalArgumentException {
    assertThat(ps.readDisplay(), is( value: 2));
}
```



```
@Test
public void shouldShow2MinFor5CentEntered() throws IllegalArgumentException {
    assertThat(ps.readDisplay(), is( value: 2));
}

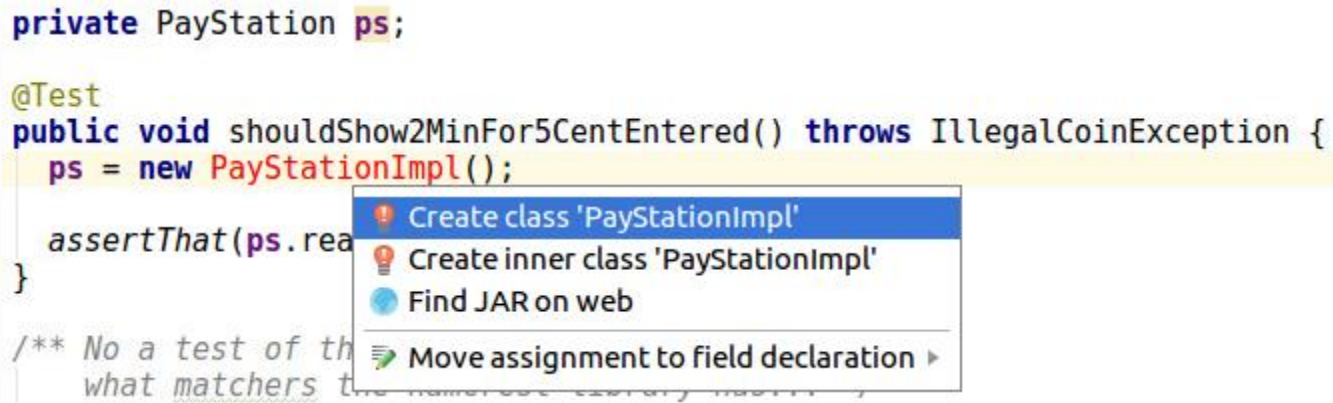
/** No a test
    what match
@Test
public void sh
assertThat( actual. THIS IS A DUMMY TEST , containsString( substring.
```

- Create local variable 'ps'
- Create field 'ps'
- Create parameter 'ps'
- Import static constant 'sun.plugin.services.PlatformService.ps'
- Rename reference

# Never Write Classes

- IDE does it quicker **‘Alt-Enter’**

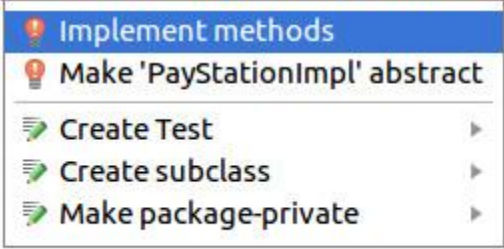
```
private PayStation ps;  
  
@Test  
public void shouldShow2MinFor5CentEntered() throws IllegalCoinException {  
    ps = new PayStationImpl();  
  
    assertThat(ps.read(), is(2));  
}  
  
/** No a test of the number of what matches the number of 2 cent coins */
```





- All classes in TDD starts as 'temporary stubs'
  - All methods are 'null' implementations:
    - Empty or 'return null;' or 'return 0;' etc.
- **'Alt-Enter'** does it faster than you

```
package paystation.domain;  
  
/**  
 * Created by csdev on 6/9/17.  
 */  
public class PayStationImpl implements PayStation {  
}
```





# Code Completion

- Never write method names in full ‘Ctrl-space’

```
@Test
public void shouldShow2MinFor5CentEntered() throws IllegalArgumentException {
    ps = new PayStationImpl();
    ps.ad
    m addPayment(int coinValue) void
    m readDisplay() int
} Press Ctrl+Period to choose the selected (or first) suggestion and insert a dot afterwards >>
```

# Review the Docs

- Hit the 'Ctrl-Q' to see the JavaDocs

```
@Test
public void shouldShow2MinFor5CentEntered() throws IllegalCoinException {
    ps = new PayStationImpl();
    ps.addPayment( coinValue: 5);

    assertTha
}

/** No a te
    what ma

@Test
public void
    assertTha
    String s
    assertTha
    "OK";
}
```

Documentation for addPayment(int)

paystation.domain.PayStation

```
public void addPayment(int coinValue)
    throws IllegalCoinException
```

Insert coin into the pay station and adjust state accordingly.

**Parameters:**

- coinValue - is an integer value representing the coin in cent. That is, a quarter is coinValue=25, etc.

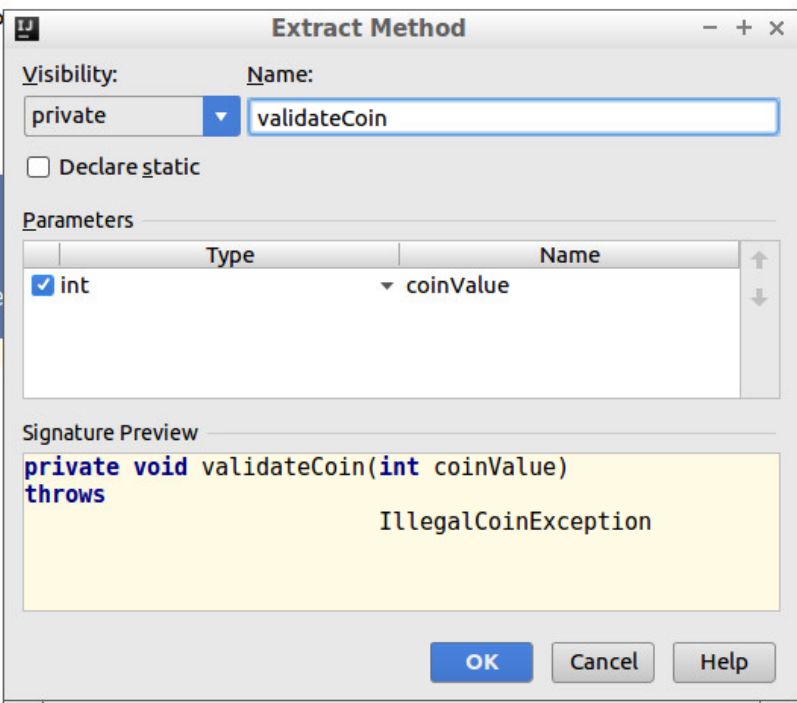
- We will look at that later, but – example
  - Extract Method **‘Ctrl-Alt-M’**

```
public class PayStationImpl implements P
private int insertedSoFar;
private int timeBought;

@Override
public void addPayment(int coinValue)
    switch (coinValue) {
        case 5: break;
        case 10: break;
        case 15: break;
        default: throw new IllegalCoinExce
    }
    insertedSoFar += coinValue;
    timeBought = insertedSoFar / 5 * 2;
}

@Override
public int readDisplay() {
    return 0;
}

@Override
public Receipt buy() {
    return null;
}
```



**Extract Method**

Visibility: private    Name: validateCoin

Declare static

Parameters

	Type	Name
<input checked="" type="checkbox"/>	int	coinValue

Signature Preview

```
private void validateCoin(int coinValue)
throws
    IllegalCoinException
```

OK    Cancel    Help

- Hit the button and you get...

```
@Override
public void addPayment(int coinValue) throws IllegalCoinException {
    validateCoin(coinValue);
    insertedSoFar += coinValue;
    timeBought = insertedSoFar / 5 * 2;
}

private void validateCoin(int coinValue) throws IllegalCoinException {
    switch (coinValue) {
        case 5: break;
        case 10: break;
        case 15: break;
        default: throw new IllegalCoinException("Invalid coin, value is "+coinValue);
    }
}
```



- IDE's automatically notice that you add source files and can keep the SCM system up-to-date!
  - You cannot compile your friends code after pulling it from a git repo if one file was not added to the repo!



# Summary

- You can code in notepad/vi/gedit and the Java compiler
  - However modern IDEs help you code *much faster*
  - Drawback
    - Steep learning curve ☹️
      - Find a 'cheat sheet' /default keymap on the web!
- At the exam, there is no IntelliJ!