Prof. Dr. Eicke Godehardt
Frankfurt University of Applied Sciences

Exercise Sheet 2

October 26th: Template and Command Pattern

Exercise 1

Finding Patterns in real code:

1. Have a look into the `java.io` package (get the sources)!
2. Where do you find examples of the Template Pattern?

Exercise 2

Have a look at the `Macro Command` from Head First, i.e., package `headfirst.command.party`.

1. Compile the packages!
2. Run the `RemoteLoader` and analyze, debug and understand the code!
3. Change the `main` of the `RemoteLoader` to record a different Macro!

Exercise 3

Implement Command Patterns for bank-accounts:

1. Implement a class `BankAccount` with a single attribute `balance`!
2. Implement an interface `Command` with two methods `void execute()` and `void undo()`!
3. Implement this interface with two classes `DepositCommand` and `WithDrawalCommand`!
4. Write a test case!
5. What happens in case of failures?
6. Can you execute a `DepositCommand` multiple times? How to avoid this?

Exercise 4

Have a look at the `java.util.concurrent.Callable` package. Compile the following code

```
package command.concurrent;

import java.util.concurrent.Callable;

public class CallableImpl implements Callable<Integer> {

    private int myName;

    CallableImpl(int i) {
        myName = i;
    }

    public Integer call() {
        for (int i = 0; i < 10; i++) {
            System.out.println("Thread : " + getMyName() + " I is : " + i);
        }
        return new Integer(getMyName());

    }
```

```
    public int getMyName() {
        return myName;
    }

    public void setMyName(int myName) {
        this.myName = myName;
    }

}
```

and

```
public class Test {

    public static void main(String[] args) {
        Callable<Integer> callable1 = new CallableImpl(1);
        Callable<Integer> callable2 = new CallableImpl(2);
        ExecutorService executor = new ScheduledThreadPoolExecutor(5);
        Future<Integer> future1 = executor.submit(callable1);
        Future<Integer> future2 = executor.submit(callable2);
        try {
            System.out.println("Future1 value: " + future1.get());
            System.out.println("Future2 value: " + future2.get());
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

1. Understand the code!
2. What is the difference of a `Callable` as compared to a `Command`?

Hints

- Consult the literature!
- You can work in pairs, if you want!
- If you want to learn a Java API, look into the java docs!
- Always use the same familiar IDE (suggestion Eclipse)!