Semesterprojekt
Implementierung eines Brettspiels
(inklusive computergesteuerter Spieler)

Wintersemester 16/17

Introduction to Coding Conventions

Patrick Schäfer
patrick.schaefer@hu-berlin.de

Marc Bux
buxmarcn@informatik.hu-berlin.de

Good or Bad Coding Style?

```
int i = 0;
while (i < 10)
      System.out.println(i);
      i++;
```

- What is the expected output?

Good or Bad Coding Style?

```
int i = 0;
for (i = 0; i < 10; i++);
        System.out.println(i);
```

- What is the expected output?

Bad Coding Style?

```
HashMap<String, Movie> actorsMap = new HashMap<>();
String actor = "";
if (  actorsMap.containsKey(
        actor
        )
        )
{ actorsMap.get(
        actor
        ).movieCount++
        ;} else
        { actorsMap.put(
            actor, new Actor(actor)
            )
            ;}
```

- What is the functionality of this code?

Motivation

- Goal: Self-documenting code that is easy to read.

- A coding convention defines the style of your source code.

- A team should use the same standard practices for:
  - **naming** classes, variables or functions.
  - **commenting** and **formatting** (indentation and brackets).

- Different conventions reasonable for every language (JAVA; C#; C++) and team.

Indentation and Bracket Placement Examples

```
 if (actorsMap.containsKey(actor))
{
            actorsMap.get(actor).movieCount++;
 }
else
{
            actorsMap.put(actor, new Actor(actor));
}
```

- ANSI C Style

Indentation and Bracket Placement Examples

```
if (actorsMap.containsKey(actor)) {
        actorsMap.get(actor).movieCount++;
 } else {
        actorsMap.put(actor, new Actor(actor));
}
```

- Kernighan and Ritchie Style

Benefits

- No need to **reformat** code and **rename** variables and methods whenever working on code written by **others**.

- Source code is much **easier to understand** when reasonably formatted:

```
if(condition)
    // statement;
otherStatement;
```

```
if (condition)
    statement;
    otherStatement;
```

```
if (condition) {
    statement;
    otherStatement;
}
```

Identifiers

- Upper Case with Underscores:

  THIS_IS_AN_EXAMPLE

- Lower Camel Case:

  thisIsAnExample

- Upper Camel Case:

  ThisIsAnExample

- Lower Case with Underscores:

  this_is_an_example

Typical Conventions

- Classes use Upper Camel Case
    - eg: MovieFactory
- Functions use lower Camel Case
    - eg: readMovie
- Variables use lower Camel Case
    - ex: movies

Indentation, Spaces and Tabs

- Don’t mix tabs and spaces! Best is to set your editor to replace tabs by 2-4 spaces when you enter a tab.

- Do (spaces or tabs):

```
if (condition) {
    block;
} else {
    block;
}
```

- Don‘t (space + tabs):

```
if (condition) {
        block;
    }
        else {
    block;
        }
```

Conclusion

- Agree on a coding guideline within your team stick to it!

- Good starting points:
  - JAVA:
    - https://google.github.io/styleguide/javaguide.html
    - https://github.com/twitter/commons/blob/master/src/java/com/twitter/common/styleguide.md
  - C#
    - http://www.dofactory.com/reference/csharp-coding-standards