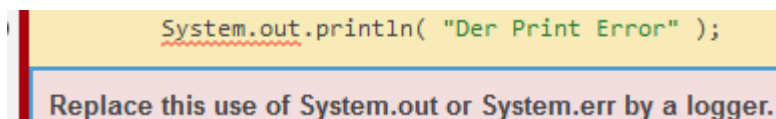


# Clean Code Development

## Einleitung :

Ich habe für diese Aufgabe meine Tic Tac Toe Spiel aus der vorherigen Aufgabe genommen und mit Hilfe von SonarQube den Code korrigiert. Zusätzlich habe ich eine Liste geschrieben, welche die wichtigsten Regeln für mich persönlich und meine Fehler zusammen fasst.

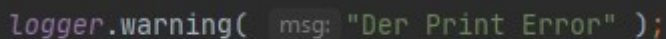
### 1. Regel: Logge keine Fehler mit System.out.print. Benutze die Logging Klasse



```
System.out.println( "Der Print Error" );
```

Replace this use of System.out or System.err by a logger.

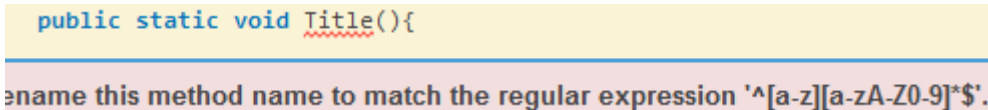
statt dessen



```
logger.warning( msg: "Der Print Error" );
```

Die Logger Klasse wurde dafür entwickelt sie zeigt sogar in Farbe an um was für einen Fehler es sich handelt. Dadurch wird der Konsolen Output deutlich lesbarer.

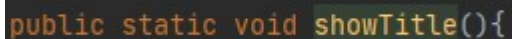
### 2. Regel: Methoden werden immer klein geschrieben und müssen mit ihrem Namen erklären, was sie tun.



```
public static void Title(){
```

Rename this method name to match the regular expression '^[a-z][a-zA-Z0-9]\*\$'.

statt dessen



```
public static void showTitle(){
```

Jeder geschriebene Code soll von jedem verstanden werden können. Ein andere könnte diese Methode als Objekt oder Klasse falsch lesen. Das kann Probleme verursachen. Außerdem ist es auch irgendwo eine Respekt Sache. Ich will deine Code lesen können und du willst meinen Code lesen können.

**3. Regel:** Nur an einer Stelle in einer Methode Return benutzen.

```
public static String startGame(){  
    Refactor this method to not always return the same value.  
    Code Smell | Blocker | Open | Not assigned | 4min  
  
    System.out.println("Wählen 1. Schere 2. Stein");  
  
    Scanner myObj = new Scanner(System.in);  
  
    String choice = myObj.nextLine();  
  
    if (choice.contains("1") | choice.contains("2"))  
        return choice;  
    }else{  
        System.out.println("Keine gültige Auswahl");  
        startGame();  
    }  
    return choice;  
}
```

Statt dessen :

```
String choice = myObj.nextLine();  
String returnChoice = "";  
  
if (choice.contains("1") | choice.contains("2"))  
    returnChoice = choice;  
}else{  
    logger.warning("Keine gültige Auswahl");  
    startGame();  
}  
return returnChoice;
```

Wenn eine Methode mehrere Returns hat und diese den selben Wert zurück geben, so ist dies ein Zeichen von schlechtem Design.

**4.Regel:** Bei Oder Ketten entweder || benutzen oder Switch Case:

```
if (choice.contains("1") | choice.contains("1") | choice.contains("2") | choice.contains("2"))  
    Refactor this "|" to "||" and extract the right operand to a variable if it should always be evaluated.  
    Is this an issue?  
    Code Smell | Blocker | Open | Not assigned | 5min effort | Comment
```

statt dessen

```
if (choice.contains("1") || choice.contains("1") || choice.contains("2") || choice.contains("2"))  
    returnChoice = choice;
```

Bei einem | „non short-circuit“ wird die rechte Seite nicht evaluiert, sofern es nicht nötig ist.  
Bei || „short-circuit“ werden beide Seiten evaluiert.

**5. Regel:** Variablen werden klein geschrieben !

```
String Result = Integer.toString(result);
```

statt dessen :

```
String result = Integer.toString()
```

Man muss sich als Programmierer an die Konventionen halten! Wie bei Regel 2.

**6. Regel:** Keinen Loop ohne End Bedingungschreiben

```
while (true){  
    logger.warning("While True Fehler");  
    System.out.println(text);  
}
```

statt dessen:

```
int i;  
i = 0;  
while (true){  
    logger.warning(msg: "While True Fehler");  
    System.out.println(text);  
  
    i++;  
    if (i == 200) {  
        break;  
    }  
}
```

Ein Loop der nicht beendet werden kann, ist ein Bug, da er das Programm in einer Dauerschleife gefangen halten kann. Dies kann Fatale Folgen haben.

**Regel 7.** Schreibe niemals Code, der einfach nur schnell und dreckig funktioniert. Schreibe immer nach den Regeln und nimm dir für jedes Problem Zeit und versuche nicht schnell durch zu rennen. In der Ruhe liegt die Kraft.

Dies ist meine Persönliche Regel. Die meisten Fehler waren mir klar, es ist mein ungestümes und unruhiges Coden, was die meisten Fehler produziert. Der Gedanke das man endlich fertig sein möchte oder die nächste Idee zum funktionieren bringen will führt dazu, dass man unachtsam wird. Oft sage ich mir, „Ach den Code korrigiere ich später!“ Problem aber ist, das die Fehler die ich getan habe zu Bugs führen mit denen ich mich später befassen muss. Dies hindert mich daran große Probleme zu lösen oder bringt mich dazu Lösungen zuschreiben die Fehler lösen, die wiederum auf schelten Code im voraus basieren. Dsh. Der Code bleibt schlecht. Kurz gesagt aus Sch\*\*\* kann nur Sch\*\*\* entstehen, egal wie viel man versucht zu retten.