13.12 Übungsblatt 12

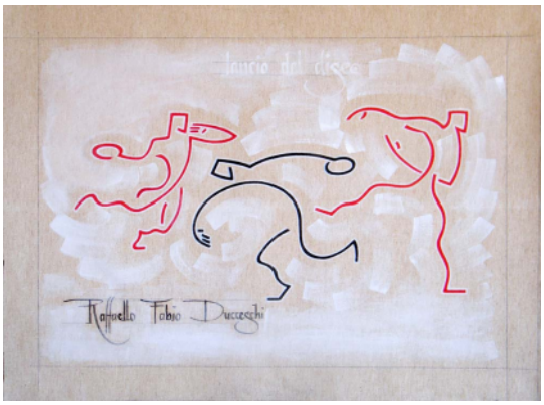

Bild 13.8: Diskuswurf - Leichtathletik, Gemälde von Raffaello Fabio Ducceschi

In diesem Blatt soll mit Ausnahmen (Exceptions) gearbeitet werden.

13.12.1 Ausnahmen fangen

Die Division durch 0 führt zu einer `ArithmeticException`. Die Klasse `ArithmeticException` ist von `RuntimeException` abgeleitet und damit eine *Unchecked Exception*.

Man kann also, ohne sich um Ausnahmen zu kümmern, eine Methode zur Division von zwei Werten so aufschreiben:

```
/**
 * Zwei Werte dividieren
 * @param z, Zahl die dividiert wird (Zähler)
 * @param n, Zahl durch die dividiert wird (Nenner)
 * @return z / n
 */
public static int div(int z, int n) {
        return z / n;
}
```

Dies hat aber zur Folge, dass bei der ersten Division durch 0 das Programm abbricht und zuvor ein Stack-Trace ausgegeben wird:

```
Exception in thread "main" java.lang.ArithmeticException: / by zero
       at blattxx_sols.ExampleForDivByZeroException.div(ExampleForDivByZeroException.java:14)
       at blattxx_sols.ExampleForDivByZeroException.main(ExampleForDivByZeroException.java:26)
```

Dieses strenge Abbrechen des Programms ist meist nicht erwünscht. Eine Möglichkeit, etwas sanfter mit der Division durch 0 umzugehen, ist, im Falle der Division durch 0 den Wert ∞ (unendlich) zurückzugeben. Bei dem Datentyp `double` ist dies mit den Konstanten

`Double.POSITIVE_INFINITY` und `Double.NEGATIVE_INFINITY` direkt möglich (daher gibt es dort auch keine Ausnahme bei Division durch 0). Bei dem Datentyp `int` kann ersatzweise die größte vorhandene Zahl (`Integer.MAX_VALUE`) oder die kleinste vorhandene Zahl (`Integer.MIN_VALUE`) zurückgegeben werden.

Aufgabe 12.1: ***Ausnahmen mit try-catch fangen***

Die obige Methode `div` soll wie folgt geändert werden:

1. Arithmetische Ausnahmen sollen mit `try-catch` abgefangen werden.
2. Tritt eine Ausnahme auf, soll Folgendes passieren:
   - Es soll der Grund für die Ausnahme nach `System.err` ausgegeben werden. **Hinweis:** Der Grund kann über die geworfene Ausnahme ermittelt werden.
   - Ein Stack-Trace ausgegeben wird. Dies kann ebenfalls über die geworfene Ausnahme getan werden.
   - Die Ausgabe bei einer Ausnahme sieht dann zum Beispiel so aus:

```
Fehlergrund: "/ by zero"
java.lang.ArithmeticException: / by zero
        at
blatt12.ExampleForDivByZeroException.div(ExampleForDivByZeroException.java:17)
        at
blatt12.ExampleForDivByZeroException.main(ExampleForDivByZeroException.java:31)
```

   - Es soll ein Ersatzwert zurückgegeben werden.

13.12.2 Mit Ausnahmen arbeiten, Ausnahmen werfen

Viele Bibliotheken verwenden Exceptions, um Fehlerzustände zu signalisieren. So auch die Bibliotheken, die sich mit Datenbanken und Dateien befassen.

Seit Java 1.7 gibt es beispielsweise die `java.nio.file`-Bibliothek, die das Schreiben und Lesen auf/von Dateien sehr vereinfacht hat.

Aufgabe 12.2: ***Behandlung von in Bibliotheken gegebenen Exceptions ändern***

Diese Bibliothek wird in der Klasse `blatt12.Statistik` verwendet, um eine Statistik in einer Datei auszugeben. Der Java-Quellcode ist in ILIAS verfügbar. Die Klasse `blatt12.Statistik` ist dabei eine sehr einfache Statistik-Klasse, der mit der Methode `addWert` neue Werte hinzugefügt werden können. Für diese Werte kann dann mit der Methode `getDurchschnitt` der Durchschnitt dieser Werte berechnet werden. Beispiel:

```
Statistik statistik = new Statistik("Teststatistik);
statistik.addWert(2.0);
statistik.addWert(3.0);
System.out.println(statistik.getDurchschnitt());
// -> 2.5 wird ausgegeben
```

Diese Klasse verfügt über die Methode `writeToFile`, die eine solche einfache Statistik als Datei ausgibt:

```
/** Statistik in Datei mit dem Namen der Statistik und der Endung ".statistik"
ausgeben
 * @see https://stackoverflow.com/questions/2885173/how-do-i-create-a-file-and-
write-to-it-in-java
 */
public void writeToFile()  {
        try {
                Files.write(Paths.get(this.name + ".statistik"),
this.toString().getBytes());
        } catch (IOException e) {
                System.out.println(e);
        }
}
```

Damit wird eine Datei erzeugt, die den Namen der Statistik hat, der um die Dateiendung ".statistik" erweitert wird. Wenn der Name der Statistik kein gültiger Dateiname ist, weil er zum Beispiel das Zeichen "\" unter Windows enthält, so wird von `File.write` eine Ausnahme geworfen.

Die Methode `writeToFile` verwendet `try-catch`-Blöcke um eine solche Ausnahme zu behandeln. Im `catch`-Block wird dabei einfach die Ausnahme am Bildschirm ausgegeben.

Dies ist für eine Klasse, die von anderen Klassen wiederverwendet werden soll, ein schlechtes Verhalten, da Nutzer der Methode `writeToFile` der Klasse `Statistik` gar keine Möglichkeit haben, auf einen solchen Fehler zu reagieren, zum Beispiel durch die Wahl eines anderen Namens, der keine verbotenen Sonderzeichen enthält.

In dieser Aufgabe soll daher die Methode `writeToFile` so geändert werden, dass sie eine Ausnahme wirft.

Zum Testen Ihrer Änderung steht die Methode `testDateiAusgabe` in der Klasse `blatt12.TestStatistikAusnahmen` bereit, die ebenfalls in ILIAS vorliegt.

Das Ergebnis nach erfolgreichem Einbau des Werfens von Ausnahmen in die Methode `writeToFile` sieht dann so aus:

```
1. TEST DER EXCEPTION BEI DATEIAUSGABE:

1.1. Test, ob Exception bei nicht gültigem Dateinamen (== Name der Statistik)
kommt
Erwartete Exception kam.
```

Aufgabe 12.3: ***Selbst Exceptions werfen, wenn Klasse in schlechtem Zustand ist***

Die gegebene Klasse `Statistik` ist zustandsabhängig: Solange diese Klasse keine Werte über die Methode `addWert` bekommen hat, ist sie in einem ungültigen Zustand. Sie kann in diesem Zustand weder einen Durchschnitt berechnen, noch sollte sie sich als Zeichenkette ausgegeben, da sie ja noch gar keine Werte hat.

Ändern Sie daher bitte die Methoden

- `getDurchschnitt` und
- `toString`

so ab, dass Sie für leere Statistiken eine `RuntimeException`-Ausnahme werfen.

Verwenden Sie bitte die Methode `testLeereStatistiken` der Klasse `blatt12.TestStatistikAusnahmen`,
um das korrekte Werfen zu prüfen.

Das Ergebnis der Methode `testLeereStatistiken` nach erfolgreichem Einbau des Werfens von Ausnahmen bei leeren Statistiken sollte so aussehen:

```
2. EINIGE TESTS FÜR AUSNAHMEN BEI "LEEREN" STATISTIKEN

2.1. Test: Leere Statistik soll nicht in String gewandelt werden
Empty statistic cannot be serialized.

2.2. Test: Mittelwert einer leeren Statistik soll nicht ausgegeben werden
No average for an empty statistic available.

2.3. Test: Keine Ausnahme erwartet ...
Teststatistik_2: n==         1; μ==2,000000
```

Hinweis: Die englischen Fehlermeldungen, wie zum Beispiel `"Empty statistic cannot be serialized."`, kann dadurch erzeugt werden, dass man diese Zeichenkette dem Konstruktor der `RuntimeException`-Ausnahme als Konstruktor-Parameter mitgibt.

Aufgabe 12.4: *Selbst Exceptions werfen, wenn falsche Parameter verwendet werden*

Der Konstruktor der gegebenen Klasse `Statistik` ist ebenfalls fehleranfällig. Es wird nicht geprüft, ob der Parameter `name` korrekt ist. Er sollte:

1. Keine `null`-Referenz sein.
2. Nicht aus 0 Zeichen bestehen ("").
3. Der erste Buchstabe sollte in alphabetisches Zeichen sein.

Bauen Sie daher bitte den Konstruktor so um, dass er für die obigen Fälle eine `NullPointerException` oder eine `RuntimeException` wirft, wenn der Parameter `name` nicht korrekt ist.

Verwenden Sie bitte die Methode `testNames` der Klasse
`blatt12.TestStatistikAusnahmen`,
um das korrekte Werfen zu prüfen.

Das Ergebnis der Methode `testNames` nach erfolgreichem Einbau des Werfens von Ausnahmen im Konstruktor sollte so aussehen:

```
3. TESTFÄLLE FÜR DIE NAMEN DER STATISTIK:

3.1. Test, ob Ausnahme bei: Null Zeiger nicht erlaubt
Name of statistic must not be null.

3.2. Test, ob Ausnahme bei: Leerer Name nicht erlaubt
Name of statistic must not be empty.

3.3. Test, ob Ausnahme bei: Name der Statistik soll mit Buchstaben beginnen
Name of statistic must start with a letter.

3.4. Test, ob Ausnahme bei: Name, der keine Ausnahme werfen soll ...
OK-Statistik: n==        10; μ==0,541322

3.5. Test, ob Ausnahme bei: Griechischer Name, der keine Ausnahme werfen soll ...
Ελληνικές στατιστικές: n==        10; μ==0,528927
```

Hinweis: Die Prüfung, ob ein Zeichen ein *Buchstabe* ist, also zu einem Alphabet gehört, sollte man besser nicht selbst codieren. Sonst müsste man sich um alle Alphabete kümmern. Schauen Sie sich daher bitte in der Wrapper-Klasse `Character` an, welche statischen Methoden diese für einzelne Zeichen zur Verfügung stellt.