Übungen zur Vorlesung Programmieren 2
FHWS – Fakultät für Informatik und Wirtschaftsinformatik

Übungsblatt 15

Aufgabe 1: (Exceptions)

a) Schreiben Sie eine Klasse Bruch mit sinnvollen Attributen. Der Konstruktor soll übergebene Attribute übernehmen und eine `ArithmeticException` (aus der Java-Klassenbibilothek) werfen, wenn der übergebene Nenner 0 ist.

b) Gegeben sei folgende Klasse:

```
public class Punkt
{
  int x;
  int y;

  public void verschiebePunkt(int zielX, int zielY) {
    x = zielX;
    y = zielY;
  }
}
```

Ergänzen Sie die Methode dahingehend, dass sie eine checked Exception namens `PunktNichtAufDemBildschirmException` wirft, wenn dem Punkt

- eine negative x oder y-Koordinate
- eine x-Koordinate größer 1920 oder
- eine y-Koordinate größer 1080

zugewiesen wird.

Aufgabe 2:
Gegeben sei ein `OutputStream` mit der folgenden `write`-Methode:

```
public void write(byte b[], int off, int len) throws IOException
{ … }
```

Welche Vorbedingungen sollten Sie an die Parameter stellen und welche Fehlerfälle sollten sie dementsprechend in der Methode abfangen? Zeigen Sie den Quellcode.

Aufgabe 3:
Schreiben Sie eine Klasse `Person` mit den Attributen `vorname`, `nachname` und `adresse`. Eine Adresse bestehe wiederum aus `strasse`, `hausnummer`, `postleitzahl` und `ort`.
Wenn eine `Person` angelegt wird, sollen folgende Regeln gelten:

- `vorname`, `strasse` und `ort` sollen mit einem Großbuchstaben beginnen.
- `hausnummer` muss mit einer Ziffer beginnen

Wenn eine dieser Regeln verletzt wird, soll eine `ValidationException` geworfen werden.

Aufgabe 4: (Stream, Dateien, Buffered Streams)
Die meisten Stream-Beispiele benutzten bisher nicht gepufferte Streams. read- und write-Aufrufe werden bei nicht gepufferten Streams direkt an das Betriebssystem weitergereicht. Dies führt bei häufigen Aufrufen zu Performanceeinbußen. Bspw. sind Dateizugriff, Netzzugriff, etc. relativ teure Operationen.

Buffered Streams verwenden einen Puffer. Ein Puffer ist ein Zwischenspeicher im Hauptspeicher.
Buffered Input Streams greifen nur auf das Betriebssystem zu, wenn der Puffer leer ist.
Buffered Output Streams greifen nur auf das Betriebssystem zu, wenn der Puffer voll ist.

Ihre Aufgabe ist es eine Musikdatei einzulesen und eine Kopie

a) mit Hilfe von ungepufferten Streams
b) mit Hilfe von gepufferten Streams

zu erstellen.

Der Nutzer soll den Pfad zu der Musikdatei, die kopiert werden soll, eingeben können. Stellen Sie in Ihrer Implementierung sicher, dass Sie eine FileNotFoundException fangen und sinnvoll den Fehler beheben.

Messen Sie für a) und b) jeweils die Zeit! Um welchen Faktor (bzw. um wie viel Prozent) unterscheidet sich der Zeitaufwand zwischen der Verwendung eines ungepufferten und gepufferten Streams in ihrer Implementierung?

Hinweis:
System.currentTimeMillis() liefert die Anzahl der vergangenen Millisekunden seit 1. Januar, 1970 UTC (Coordinated Universal Time) zurück.

```
long startTime = System.currentTimeMillis();
//Code, dessen Dauer gemessen werden soll
//…
//…
long endTime = System.currentTimeMillis();
long duration = endTime - startTime;
```

Für kleinere Zeitauflösungen (Nanosekunden) kann auch **System.nanoTime()** anstelle von **System.currentTimeMillis()** verwendet werden.