Übung09: Ausnahmen

Abgabetermin: 24.05.19, 15:30 Uhr

Name: _____ Matrikelnummer: _____

Aufgabe	**schriftlich abzugeben**	**elektronisch abzugeben**	**gelöst**
Aufgabe23	Ausarbeitung zu gestellten Fragen		☐
Aufgabe24	Begründung der Fehlermeldung, Korrigiertes Java Programm	Korrigiertes Java Programm	☐
Aufgabe25	Java Programm, Testausgaben	Java Programm	☐

Achtung!

Bitte auf diesem Deckblatt:

- **Name** und **Matrikelnummer** ausfüllen,
- gelöste Aufgaben **ankreuzen**

und dann der schriftlichen Abgabe als erste Seite anheften.

Aufgabe23: Fragen zu Exceptions (6 Punkte)

Im Folgenden werden Code-Beispiele gezeigt und Verständnisfragen gestellt, die Sie beantworten sollen. Hinweis: Die gezeigten Beispiele sollten in dieser Form **nicht** in richtigen Programmen verwendet werden.

1. Ist folgende Methode gültig? Welcher Wert wird zurückgegeben? Welche Anweisungen haben eine Auswirkung auf den Rückgabewert und warum?

```
int foo() {
  int x = 0;
  try {
    x = x + 1;
    throw new Exception();
  } catch (Exception e) {
    x = x + 1;
    return x;
  } finally {
    x = x + 1;
  }
}
```

2. Welche Fehler können mit folgendem Exception-Handler abgefangen werden? Warum sollte man einen Handler in dieser Form nicht verwenden?

```
catch (Exception e) {
  // …
}
```

3. Ist am folgenden Code-Beispiel etwas falsch? Wenn ja, was und warum?

```
try {
  // …
} catch (Exception e) {
  // …
} catch (IOException e) {
  // …
}
```

4. Überlegen Sie für folgende Fälle, welche Art von Fehler (Error, Checked Exception, Runtime Exception, Compiler Fehler, keine Exception) auftritt bzw. geworfen wird:

a)
```
int[] values;
values[0] = 0;
```
b) Ihr Programm erzeugt bei der Ausführung ein großes temporäres Array. Aufgrund gewisser Umstände steht der Java Virtual Machine nicht genug Speicher zur Verfügung, um das Array zu erzeugen.
c) Ihr Programm liest über einen `FileInputStream` Werte aus einer Datei ein und erreicht das Ende des Streams.
d) Nachdem Ihr Programm den Stream geschlossen hat, versucht es erneut, Werte von diesem Stream einzulesen.
e) Sie übersehen für einen speziellen Fall in Ihrem Programm, einer Variable einen Anfangswert zuzuweisen. Der Fall tritt bei der Ausführung ein und Ihr Programm greift auf die uninitialisierte Variable zu.

Aufgabe24: Fehlerhafte Klasse beheben (9 Punkte)

Gegeben ist folgende Klasse `ListOfNumbers` mit der Methode `writeList`. Diese schreibt den Inhalt des internen Arrays `values` formatiert in eine Datei.

```
// Note: This class does not compile.
import java.io.*;

public class ListOfNumbers {

  private static final int CAPACITY = 10;
  private int[] values;

  public ListOfNumbers () {
    values = new int[CAPACITY];
    for (int i = 0; i < CAPACITY; i++) values[i] = i;
  }

  public void writeList(String fileName) {
    // may throw an IOException
    BufferedWriter out = new BufferedWriter(new FileWriter(fileName));

    for (int i = 0; i < CAPACITY; i++) {
      // may throw an IndexOutOfBoundsException
      int value = values[i];

      // may throw an IOException
      out.write("Value " + value + " written to file");

      // may throw an IOException
      out.newLine();
    }
    // may throw an IOException
    out.close();
  }
}
```

Es sind mehrere Stellen im Code markiert, die eine Exception auslösen können. Wenn Sie die Klasse kompilieren, werden Sie feststellen, dass der Compiler Fehlermeldungen ausgibt. **Begründen Sie**, warum der Compiler an den Stellen, wo `IOExceptions` auftreten können, Fehler ausgibt, dies aber für die Stelle, an der eine `IndexOutOfBoundsException` auftreten könnte, nicht tut. Womit hängt das zusammen?

Schreiben Sie weiters die Methode `writeList` so um, dass die Klasse kompiliert und die Fehlerfälle sinnvoll behandelt werden. Eine `IOException` kann in diesem Code geworfen werden, wenn die Datei nicht beschreibbar ist, oder es während den Schreibvorgängen zu Problemen externer Natur kommt (z.B.: Datei wird beim Schreiben von Außen gelöscht, Festplatte liefert einen Fehler, usw.).

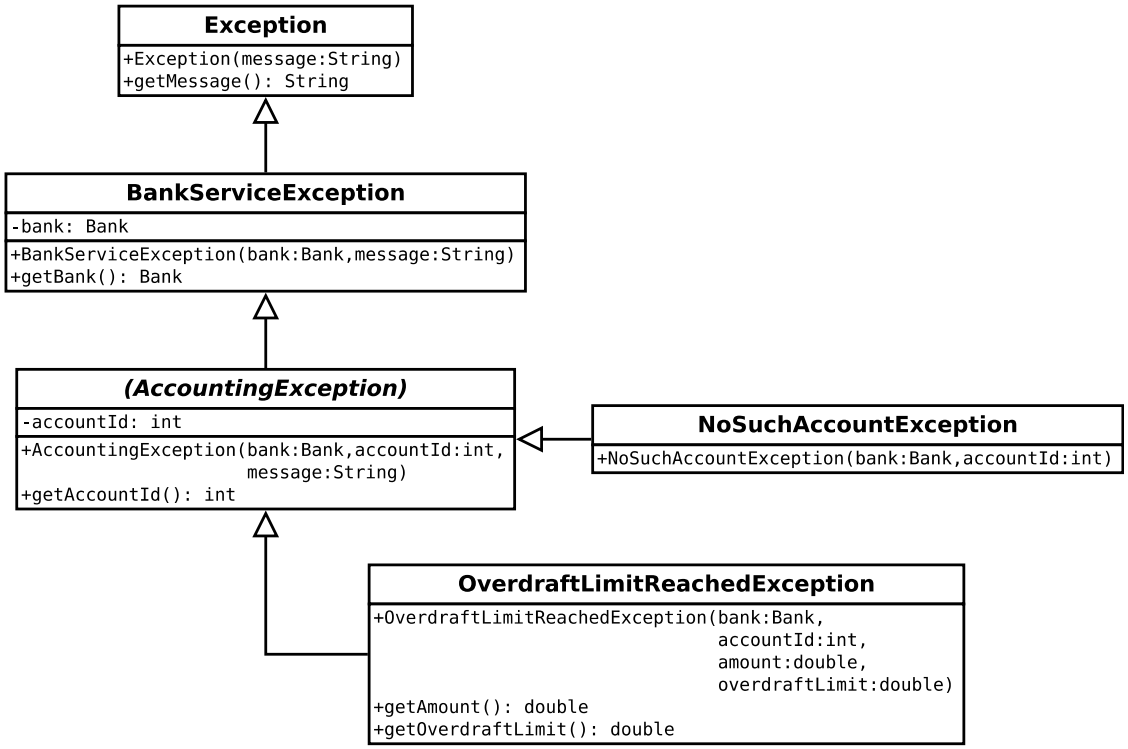
Da die Klasse `ListOfNumbers` in unterschiedlichen Umgebungen einsetzbar sein soll, ist die Methode selbst ein schlechter Ort, um die Exceptions zu behandeln. Der Aufrufer soll entscheiden, was im Fehlerfall passiert. Leiten Sie daher eine Exception im Falle eines Eintritts an den Aufrufer weiter. Sorgen Sie jedoch dafür, dass die Ressourcen **immer** freigegeben werden, indem Sie sicherstellen, dass `out.close()` richtig aufgerufen wird. **Achtung:** Auch `out.close()` kann eine `IOException` werfen.

Abzugeben sind:

- Begründung der Fehlermeldung
- Korrigiertes Java-Programm

Aufgabe25: Banksystem mit eigenen Exceptions (9 Punkte)

Gegeben ist die Musterlösung der *Aufgabe18: Verwaltung von Bankkonten* (zu finden in Moodle in der Datei `Banking.zip`). In der Musterlösung werden spezielle Rückgabewerte verwendet, um den Erfolg gewisser Operationen zu signalisieren (z.B. `false` wenn beim Abheben der Überziehungsrahmen erreicht wird, -1 wenn beim Erzeugen neuer Accounts die maximal mögliche Anzahl an Accounts erreicht wurde, etc.). Erweitern Sie die Musterlösung aus Aufgabe18 um folgende neue Klassenhierarchie:



Klasse `BankServiceException`:

- Basisklasse für Exceptions, die in der Klasse `Bank` geworfen werden.
- Besitzt eine Referenz auf das entsprechende `Bank`-Objekt.
- Definiert einen Konstruktor, über den ein `Bank`-Objekt und eine Fehlermeldung übergeben werden kann.

Klasse `AccountingException`:

- Abstrakte Klasse, die von `BankServiceException` erbt und als Basisklasse für Fehler dient, die ein konkretes Konto betreffen.
- Besitzt ein Feld für die betroffene Kontonummer.
- Definiert einen Konstruktor, der neben dem `Bank`-Objekt und der Fehlermeldung auch die betroffene Kontonummer setzt.

Klasse `NoSuchAccountException`:

- Erbt von `AccountingException` und wird geworfen, wenn ein Konto mit gegebener Nummer nicht existiert.
- Definiert einen Konstruktor, der nur das `Bank`-Objekt und die betroffene Kontonummer setzt. Eine definierte Fehlermeldung soll über einen Superkonstruktor-Aufruf gesetzt werden.

Klasse `OverdraftLimitReachedException`:

- Erbt von `AccountingException` und wird geworfen, wenn beim Abheben eines Betrags von einem Konto der Überziehungsrahmen erreicht wird.
- Definiert einen Konstruktor, der neben dem `Bank`-Objekt und der betroffenen Kontonummer auch den Betrag und Überziehungsrahmen setzt. Eine definierte Fehlermeldung soll über einen Superkonstruktor-Aufruf gesetzt werden.

Schreiben Sie weiters die gegebene Musterlösung so um, dass in der Klasse `Bank` die eingeführten Exceptions in den entsprechenden Methoden bei Bedarf geworfen werden.

Werfen Sie für den Fall, dass beim Erstellen neuer Accounts die maximale Anzahl an möglichen Accounts erreicht wurde, eine allgemeine `BankServiceException` mit einer entsprechenden Fehlermeldung.

Für den Fall, dass ein gegebener Betrag negativ ist, soll eine `IllegalArgumentException` der Java-Bibliothek mit entsprechender Fehlermeldung geworfen werfen. Sorgen Sie jedoch dafür, dass vom Benutzer keine negativen Werte eingelesen und dem `Bank`-Objekt übergeben werden, damit es nicht zu solch einer `RuntimeException` kommt. Für alle anderen Fälle werfen Sie Objekte der konkreten Ableitungen von `AccountingException`.

Passen Sie am Ende die Klasse `Banking` an, welche die Klasse `Bank` und deren Methoden verwendet. Fangen Sie die entsprechenden Exceptions beim Ausführen einer Operation im Fehlerfall ab, geben Sie die entsprechende Fehlermeldung auf der Konsole aus, und fordern Sie den Benutzer erneut dazu auf, eine Eingabe durchzuführen.

<u>Beispiel:</u>

```
********** Bankverwaltung ********
Kunde+Konto anlegen ..... a
Einzahlen ..... e
Beheben ..... b
Überweisen ..... t
Übersicht drucken ..... d
Beenden ..... q
Welche Menuoption? [a|e|b|t|d|q]: e
*** Einzahlen ***
Kontonummer: 5
Einzahlungsbetrag: 100
Fehler beim Ausführen der Operation
Grund: Account 5 existiert nicht
Wiederholen Sie es nocheinmal!

********** Bankverwaltung ********
Kunde+Konto anlegen ..... a
Einzahlen ..... e
Beheben ..... b
Überweisen ..... t
Übersicht drucken ..... d
Beenden ..... q
Welche Menuoption? [a|e|b|t|d|q]: q
```

<u>Abzugeben sind:</u>

- Java-Programm
- Testausgaben mit entsprechenden Fehlermeldungen