Exceptions

Definieren einer einfachen Exception:

```
public class ValidationException extends Exception {

	public ValidationException(String message) {
		super(message);
	}

}
```

Werfen der Exception:

```
public class Pensionist {

	private int alter;

	public void setAlter(int alter) throws ValidationException {
		if (alter >= 65 && alter <= 150) {
			this.alter = alter;
		} else {
			throw new ValidationException("Alter muss "+
				"zwischen 65 und 150 sein.");
		}
	}

}
```

Abfangen der Exception:

```
public class Seniorenheim {

	private void seniorAnlegen() {
		Pensionist pensionist = new Pensionist();
		boolean ok = false;
		while (!ok) {
			System.out.print("Alter: ");
			int alter = In.readInt();
			try {
				pensionist.setAlter(alter);
				ok = true;
			} catch (ValidationException validationException) {
				System.out.println("Alter inkorrekt - "+
					validationException.getMessage());
			}
		}
		...
	}

}
```

Bsp 1: Exceptions

a) Wozu werden Exceptions eingesetzt; wobei nicht?
b) Wozu dient die RuntimeException?

Bsp 2: Datum zu Wochentag

Lesen Sie von Tastatur ein Datum im Format TT.MM.YYYY. Konvertieren Sie das gelesene Datum mit Hilfe der Klasse `java.text.SimpleDateFormat` in ein `java.util.Date`. Ist das Datum nicht konvertierbar, so soll eine entsprechende Meldung ausgegeben werden und der Benutzer soll erneut nach einem Datum gefragt werden. Geben Sie, wieder mit Hilfe der Klasse `java.text.SimpleDateFormat`, den Wochentag zum eingegebenen Datum aus.

Bsp 3: Integer parsen

Lesen Sie mit `In.readWord()` ein Wort von der Tastatur und konvertieren Sie dieses mit `Integer.parseInt()` in einen int-Wert. Melden Sie dem Benutzer zurück, ob seine Eingabe ein Integerwert war oder nicht.

Bsp 4: Mehrere Exceptionhandler

Erklären Sie die Funktionsweise folgender Methode:

```
public void dateiAusgeben(String dateiname) {
      try {
            FileInputStream in = new FileInputStream(dateiname);
            byte[] buffer = new byte[1000];
            int bytesRead = in.read(buffer);
            while (bytesRead > 0) {
                  System.out.write(buffer, 0, bytesRead);
                  bytesRead = in.read(buffer);
            }
      } catch (FileNotFoundException ex) {
            System.out.println("Die Datei "+dateiname+
                  " wurde nicht gefunden.");
      } catch (IOException ex) {
            System.out.println("Fehler beim Lesen "+
                  "der Datei. Grund: "+ex.getMessage());
      }
}
```

Wie geht Java mit mehreren Exceptionhandlern um?

Bsp 5: Element einfügen

Schreiben Sie eine Klasse namens `ObjectArray`, welche ein Array von Werten des Typ `Object` enthält. Die Größe des Arrays soll beim Erzeugen einer `ObjectArray`-Instanz angegeben werden können.
Schreiben Sie im `ObjectArray` eine Methode `void insert(int index, Object element)`, die ein Objekt an der Stelle `index` in das Array einfügt. Elemente können vor, zwischen oder unmittelbar nach einem bereits eingefügten Element platziert werden. Wird diese Regel verletzt, so soll eine `InvalidIndexException` geworfen werden, die einen Mitteilungstext und den ungültigen Index enthält. Ist das Array voll, so soll eine `OverflowException` geworfen werden, die lediglich eine Mitteilung enthält. Beide Exceptions müssen von Ihnen geschrieben werden.

Schreiben Sie weiters einige Testaufrufe, in denen Sie die Korrektheit Ihrer Methode überprüfen und beide Exceptions verwenden.

Bsp 6: MyIn

Schreiben Sie eine Klasse `MyIn`, in der Sie folgende Methoden implementieren:
-) `void open(String filename) throws FileNotFoundException`
-) `void close() throws IOException`
-) `String readLine() throws IOException`
-) `int readInt() throws IOException`
Verwenden Sie zur Umsetzung die entsprechenden Methoden der Mössenböck In-Klasse, erweitern Sie die Funktionsweise Ihrer Methoden um Exceptions. Die Exceptions existieren bereits im `java.io`-Package.

Lesen Sie mit Hilfe Ihrer Klasse `MyIn` eine Datei mit den Daten einer Transportroute und Summieren Sie die Distanzen auf. Geben Sie die Summe sowie den Zielort (der letzte Ort in der Datei) aus. Testen Sie Ihr Programm zumindest mit den beiden folgenden Routendateien:

Route 1

```
0     Wien
66    St. Pölten
124   Linz
137   Salzburg
327   Bregenz
```

Route 2

```
0     Wien
66    St. Pölten
abc   Linz
137   Salzburg
327   Bregenz
```

Bsp 7: Stacktrace

Wie wird ein Stacktrace ausgegeben? Welche Informationen enthält ein Stacktrace?

Beispiel eines Stacktrace:

```
ValidationException: Alter muss zwischen 65 und 150 sein.
      at Pensionist.setAlter(Pensionist.java:9)
      at Seniorenheim.seniorAnlegen(Seniorenheim.java:10)
      at Seniorenheim.main(Seniorenheim.java:21)
```