Aufgabe 1 (Java-Grundlagen) [14 Punkte]

a) [3 Punkte] Bitte markieren Sie im Folgenden die korrekten Aussagen über native Arrays (keine `java.util.ArrayList`):

- ○ Die Elemente eines Arrays werden mit `null` initialisiert.
- ○ Die Elemente eines Arrays werden mit dem Default-Wert des spezifizierten Datentyps initialisiert.
- ○ Ein Array wird in Java stets mit Call-by-Reference an Methoden übergeben.
- ○ Ein Array wird in Java stets mit Call-by-Value an Methoden übergeben.
- ○ Ein Array ist eine generische Datenstruktur.
- ○ Ein Array ist ein primitiver Datentyp.

b) [5 Punkte] Gegeben sei eine `java.util.List`, die Elemente des Typs `Product` beinhaltet. Bitte implementieren Sie die Methode `printProducts(List<Product> productList)`, die die Inhalte dieser Liste *mithilfe eines Iterators* ausgibt. Verwenden Sie keine `foreach`-Schleife. Sie können davon ausgehen, dass die Klasse `Product` eine valide `toString()`-Methode enthält. Die Methode ist wie folgt eingebettet bzw. wird wie folgt aufgerufen:

```
1  public static void main(String[] args) {
2      List<Person> productList = new ArrayList<Person>();
3      // ... code for filling the list...
4      printProducts(productList);
5  }
```

c) [3 Punkte] Welches Ausführungsschema verwendet Java? Wie funktioniert dieses und welche Vorteile bringt es mit sich?

d) [3 Punkte] Erklären Sie kurz die Begriffe *Call-by-Value* und *Call-by-Reference*. Was wird in Java eingesetzt?

Aufgabe 2 (Objektorientierung, Polymorphie) [22 Punkte]

a) [2 Punkte] Was versteht man unter dem Begriff des "Autoboxing"? Geben Sie bitte auch ein Beispiel an.

b) [2 Punkte] Wozu werden die Methoden `hashcode` und `equals` bei `java.util.HashMaps` verwendet?

c) [2 Punkte] Wieso sollten Sie stets auch die `hashcode`-Methode auch überschreiben, wenn Sie die `equals`-Methode überschreiben? Was kann passieren, falls Sie dies nicht machen?

d) [3 Punkte] Was versteht man unter dem Prinzip der Ersetzbarkeit? Wie hängt dieses mit der Vererbung in Java zusammen?

e) [3 Punkte] Gegeben sei folgende Klasse `Track`. Erweitern Sie die Klasse dahingehend, dass sie das `Comparable`-Interface implementiert. Zwei Tracks sollten anhand der Attribute `trackTitle`, `artistName` und `albumName` verglichen werden. Bei Gleichheit des `trackTitle` soll der `artistName` zum Vergleich verwendet werden und bei erneuter Gleichheit soll der `albumName` verwendet werden soll.

```
1
2 public class Track {
3
4    private String trackTitle;
5    private String artistName;
6    private String albumName;
7
8    /* getter and setter */
9 }
```

f) [1 Punkt] Wie können Sie basierend auf der vorigen Aufgabe eine `java.util.List trackList` von Tracks aufsteigend sortieren? Bitte geben Sie den entsprechenden Aufruf an.

g) [7 Punkte] Betrachten Sie folgenden Code und beantworten Sie die darunter angeführten Fragen.

```
1 public class Application {
2    public static void main(String[] args) {
3       ClassA a = new ClassA();
4       ClassA b = new ClassB();
5       ClassB c = new ClassB();
6
7       a.methodA();
8       b.methodA();
9       c.methodZ("foo");
10      c.methodA();
11   }
12 }
13
14 public class ClassA {
15    public String x = "classA";
16    private String y = "test";
17    protected String z = "z";
18
19    public void methodA() {
20       System.out.println(x + ", " + y);
21    }
22 }
23
24 public class ClassB extends ClassA {
25   private String x = "classB";
26   private String y = "y";
27
28   public void methodZ(String x) {
29      System.out.println(x + ", " + y + z);
30   }
31
32   public void methodA() {
33      System.out.println(x);
34   }
35 }
```

1) Welche Methoden können Sie in der `main`-Methode auf dem Objekt `a` aufrufen? (1 Punkt)

2) Welche Methoden können Sie in der `main`-Methode auf dem Objekt `b` aufrufen? (1 Punkt)

3) Welche Methoden können Sie in der `main`-Methode auf dem Objekt `c` aufrufen? (1 Punkt)

4) In `ClassB` kann auf `z` zugegriffen werden, wieso? (1 Punkt)

5) Was wird beim Aufruf der `main`-Methode ausgegeben? (3 Punkte)

h) *2 Punkte* Wann werden in Java statische Methoden bzw. Variablen initialisiert? Wann werden nicht-statische Methoden bzw. Variablen initialisiert?

Aufgabe 3 (Exceptions) [10 Punkte]

a) [3 Punkte] Was versteht man unter der *lokalen Behandlung* von Exceptions? Geben Sie bitte Beispielcode an, der eine derartige lokale Behandlung demonstriert.

b) [3 Punkte] Was versteht man unter dem *Weiterreichen* von Exceptions? Geben Sie bitte Beispielcode an, der eine Weiterreichen demonstriert.

c) [4 Punkte] Betrachten Sie folgenden Code und erweitern Sie diesen wie folgt um Exception Handling. Sie können dabei davon ausgehen, dass die Methoden `bar` und `com` korrekt implementiert wurden.

- Die Methode `bar` könnte durch eine mögliche Division durch 0 eine `ArithmeticException` werfen. Reichen Sie diese bitte weiter.
- Die Methode `com` kann eine `IllegalArgumentException` werfen, diese soll weitergereicht werden.
- Die Methode `com` kann auch eine (checked) `MyCustomException` werfen, diese soll lokal behandelt werden, indem Sie vereinfach "MyCustomException occurred" ausgeben.

```
1 public void foo(int i) {
2   if (i < 3 ) {
3     bar(i/3);
4   }
5   else {
6     com(i+1);
7   }
8 }
```

Aufgabe 4 (Testen) [12 Punkte]

a) [4 Punkte] Betrachten Sie die statische Methode `double computeAverage(List<Integer> input)` der Klasse `TestingExercise`, die eine `java.util.List` von Ganzzahlen entgegennimmt und den Durchschnitt der enthaltenen Zahlen berechnet. Implementieren Sie diese Methode im Folgenden aus.

b) [6 Punkte] Verfassen Sie für die obige Methode `double computeAverage(List<Integer> input)` zwei JUnit-Testmethoden.

c) [2 Punkte] Wofür kann die Annotation `@BeforeEach` verwendet werden?

Aufgabe 5 (Generics) [15 Punkte]

a) [3 Punkte] Wieso kann in Java in generischen Klassen kein Objekt mit einem Konstruktor der generischen Klasse erstellt werden?

b) [3 Punkte] Wie definiert man in Java den "Raw Type" einer Klasse? Wie entsteht dieser?

c) [3 Punkte] Wozu führt der Compiler Brückenmethoden ein? Wieso werden diese benötigt?

d) [2 Punkte] Betrachten Sie folgende Methodensignatur:

```
1 public <T extends Comparable<? super T>> T foo ()
```

Welche Klassen sind hier als konkretes Typargument für die Typvariable T zulässig bzw. welche Bedingungen müssen diese erfüllen?

e) [4 Punkte] Implementieren Sie eine *generische* Klasse `DataStore`, die folgende Anforderungen erfüllt: es sollen Zahlen gespeichert und dann entsprechend ihrer Position (Index) wieder abgefragt werden können. Achten Sie bei der Implementierung unbedingt darauf, dass nur Zahlen als generischer Typ verwendet werden können (z.B. Integer oder Double). Eine beispielhafte Verwendung der Klasse finden Sie im Folgenden:

```
1 public class Main {
2
3   public static void main(String[] args) {
4     DataStore<Integer> store = new DataStore<Integer>();
5     store.setNumber(13);
6     store.setNumber(45);
7     System.out.println(store.getNumber(1));
8   }
9 }
```

Aufgabe 6 (Funktionale/moderne Programmierung) [8 Punkte]

a) [2 Punkte] Gegeben sei folgende `java.util.HashMap`:

```
1 Map<Integer, String> products = new HashMap<Integer, String>();
2 products.put(1, "Product 1");
3 products.put(2, "Product 2");
4 products.put(3, "Product 3");
```

Geben Sie die Inhalte dieser Map mittels eines Befehls mittels Java's funktionaler Sprachelemente aus. Die Ausgabe soll wie folgt aussehen:

```
1 1: Product 1
2 2: Product 2
3 3: Product 3
```

b) [2 Punkte] Welches Functional Interface wird für `forEach` benötigt und was gibt dieses Interface vor?

c) [2 Punkte] Was sind "Default Methoden"? Wo können diese implementiert werden?

d) [2 Punkte] Gegeben sei `List<Integer> list = new LinkedList<Integer>();`.

Filtern Sie diese Liste mithilfe von funktionalen Sprachelementen dahingehend, dass nur mehr durch 3 teilbare Integer in der resultierenden Liste enthalten sind.

Aufgabe 7 (JVM) [9 Punkte]

a) [2 Punkte] Was ist der Inhalt einer `.class`-Datei? Wie wird diese zur Ausführung von Programmcode genützt?

b) [3 Punkte] Welche Aufgabe hat die sogenannte Dispatch-Tabelle einer Klasse? Wann wird diese benötigt?

c) [4 Punkte] Was versteht man unter dem Konzept von *Generational Collectors*? Welche Eigenschaften werden hier ausgenützt und wie funktioniert es?

Aufgabe 8 (Clean Code und Refactoring) [10 Punkte]

a) [2 Punkte] Welche Rolle spielen beim Refactoring vorhandene Unit-Tests? Wozu werden diese eingesetzt?

b) [4 Punkte] Beschreiben Sie kurz das "Replace Temp with Query"-Refactoring und geben Sie ein kurzes Beispiel an.

c) [4 Punkte] Beschreiben Sie kurz das "Pull Up Constructor Body"-Refactoring und geben Sie ein kurzes Beispiel an.