EF Informatik: Lernziele 2. Java-Prüfung (6.1.2023)

Die Prüfung hat zwei Teile. Der erste, theoretische Teil findet auf Papier ohne Computerunterstützung statt, der zweite ist ein Programmierteil, der unter den gleichen Bedingungen stattfindet wie die letzte Prüfung:

Die Programmierprüfung wird ähnlich wie die Übungen gestaltet sein. Das Internet wird aber nur am Anfang für das Klonen des Repositories und am Schluss für die Abgabe verfügbar sein. Anderweitige Benutzung des Internets ist untersagt. Aufgabenstellung und Abgabe erfolgen über Github. Es werden nur die bei den Übungen verwendeten Module einzusetzen sein (`Java.util.Random`, `Java.util.Arrays`, `Java.lang.Math`, `Java.util.Scanner`). Andere Module sind nicht zugelassen und führen zu Punkteabzug. Das Repository `java-infos-and-snippets` sollte vor Beginn der Prüfung geklont (mit `git clone`) bzw. aktualisiert (mit `git pull`) werden. Es ist erlaubt, in diesem Repository nachzuschauen, wie gewisse Befehle heissen oder wie man sie einsetzt.

1. Die Lernziele aus der letzten Prüfung bilden die Basis für die Themen dieser Prüfung und müssen grundsätzlich beherrscht werden. Ausgenommen sind die Punkte 16, 20, 22 und 39.
2. Es sollen auch Theoriefragen beantwortet werden können, insbesondere zu den folgenden Punkten:

   6) Die primitiven Datentypen (`boolean`, `int`, `double` etc.) kennen:
      i. Erklären, wie sie im Speicher abgebildet werden.
      ii. Händisches Rechnen mit Zahlen im Zweierkomplement (inkl. integer overflow).
      iii. Erklären, was Wrapper-Klassen sind und warum und wie sie benutzt werden.

   8) Zuweisung von Arrays/ Objekten:
      i. Erklären und aufzeichnen, was im Speicher bei der Deklaration und bei der Initialisierung von Arrays/ Objekten passiert
      ii. Erklären, was bei Zuweisungen per = zwischen Arrays/ Objekten vom gleichen Typ passiert.
      iii. Wissen und erklären, was Pointer und Garbage Collection sind und wie sie in Java funktionieren.
      iv. Wissen, dass und wie sich primitive Datentypen diesbezüglich anders verhalten.

   25) Aus einfachen Codebeispielen herauslesen, ob eine Variable eine Instanz- oder eine Klassenvariable ist und welchen Wert sie zu einem bestimmten Zeitpunkt hat.

   28) Codebeispiele lesen können und verstehen, ob eine statische oder eine dynamische Methode einer Klasse aufgerufen wird.

   37) Ein Beispiel für implizites Casting in Java nennen oder eines verstehen.

   38) Explizites Casting einer Instanz einer Basisklasse zur passenden abgeleiteten Klasse: Erklären, wie der Methodenaufruf von überschriebenen Methoden funktioniert.

3. Die Prinzipien von Objektorientierter Programmierung kennen und Beispiele nennen, wie diese Prinzipien in Java umgesetzt werden.
4. Die `@Override`-Annotation kennen und wissen, was sie macht.
5. Die Unterschiede und Gemeinsamkeiten zwischen normalen `parent`-Klassen, abstrakten Klassen und Interfaces erklären und anwenden.
6. Die Unterschiede zwischen abstrakten und nicht abstrakten Methoden kennen und sie implementieren.
7. Generische Methoden und Klassen kennen und schreiben können.
8. Interfaces zur Einschränkung generischer Typ-Parameter einsetzen.
9. Das Interface `Comparable` kennen und generische Methoden zu `Comparable`-Objekten schreiben.
10. Klassendeklarationen mit Generics genau lesen und die Notation korrekt anwenden (bezieht sich `extends` auf die Klasse oder den generischen Typ?)

11. Wissen, dass in Java type erasure stattfindet und welche Konsequenzen dies für die Verwendung von Arrays von Generics und für Arrays von Objekten generischen Typs hat.
12. Die abstrakten Datentypen (Modelle) List, Stack, Queues und die Zusammenhänge zwischen ihnen kennen.
13. Wissen, wie List, Stack und Queue als Array-Liste und als dynamische Array-Liste implementiert werden können.

Übung zur Sichtbarkeit

In diesen Code haben sich vier Fehler eingeschlichen. Finde sie und erkläre, was falsch ist.

```
class Visibility {
  static int a = 0;
  int b = 1;

  void f(Visibility v) {
    int c = 0;
    System.out.println(a);
    System.out.println(b);
    for (int i = 0; i < 10; i++) {
      System.out.println(i);
    }
    System.out.println(i);
    System.out.println(c);
  }

  public static void main(String[] args) {
    System.out.println(a);
    System.out.println(b);
    System.out.println(c);
    Visibility v = new Visibility();
    System.out.println(v.b);
    System.out.println(f(v));
  }
}
```