Prüfung Softwareentwicklung II (IB)

Datum	:	11.07.2013, 08:30 Uhr
Bearbeitungszeit	:	90 Minuten
Prüfer	:	Prof. Dr. Oliver Braun
Hilfmittel	:	Keine
Erreichbare Punkte	:	90

Name: _____

Vorname: _____

Matrikelnummer: _____ Studiengruppe: _____

Hörsaal: _____ Platz Nr.: _____

Unterschrift: _____

Bitte kontrollieren Sie, ob Sie eine vollständige Angabe mit 6 Aufgaben auf 8 Seiten erhalten haben.

Aufgabe	1	2	3	4	5	6	Summe
max. Punkte	14	23	12	18	12	11	90

Anmerkungen:

- Sie müssen als Antworten **keine** kompletten Programme schreiben, sondern nur den explizit verlangten Teil eines Programms.
- Schreiben Sie die Lösungen in die dafür vorgesehenen Kästchen. Sollte Ihnen der Platz dabei nicht reichen, benutzen Sie die Rückseite **und vermerken Sie das im dazugehörigen Kästchen!**

Aufgabe 1 (14 Punkte)

Gegeben sei folgende `equals`-Methode. Beantworten Sie die folgenden Teilfragen so exakt und knapp wie möglich.

(a) Was bedeutet `@Override`? (2)

```
@Override
public boolean equals(Object obj) {
```

(b) Was vergleicht der erste Vergleich? (2)

```
    if (this == obj)
        return true;
```

(c) Warum muss nur das andere Objekt auf `null` geprüft werden? (2)

```
    if (obj == null)
        return false;
```

(d) Was überprüfen die nächsten beiden Zeilen? (2)

```
    if (getClass() != obj.getClass())
        return false;
```

(e) Was macht die nächste Zeile und warum funktioniert es? (2)

```
    Person other = (Person) obj;
```

(f) Wann sind also die zwei Objekte gleich? (2)

```
    if (!name.equals(other.name))
        return false;
    return true;
}
```

(g) Wie heisst die Datei in der diese `equals`-Methode steht? (2)

Aufgabe 2 (23 Punkte)

Gegeben sei das folgende Interface für natürliche Zahlen:

```
interface NaturalNumber {
    NaturalNumber zero();
    NaturalNumber succ();
    NaturalNumber add(final NaturalNumber other);
    int value();
}
```

Implementieren Sie im Folgenden schrittweise eine Klasse `NatNum` die das Interface `NaturalNumber` implementiert. Natürliche Zahlen sollen intern als `int` repräsentiert werden.

(a) Geben Sie den Kopf der Klasse `NatNum` an: (2)

(b) Geben Sie alle benötigten Objekt- und Klassenvariablen an: (4)

(c) Definieren Sie einen Custom-Konstruktor, der einen `int` als Parameter hat und ein `NatNum`-Objekt erzeugt, das den entsprechenden Wert repräsentiert. Sollte der übergebene Wert negativ sein, soll das Objekt den Wert 0 repräsentieren. (4)

(d) Geben Sie eine Implementierung für die Methode `zero` an, die die natürliche Zahl 0 zurück gibt. (3)

(e) Geben Sie eine Implementierung für die Methode `succ` an, die die nächste natürliche Zahl nach der Zahl selbst zurück gibt. (3)

(f) Geben Sie eine Implementierung für die Methode `add` an, die eine natürliche Zahl zurück gibt, die die Summe dieser und der übergebenen natürlichen Zahl repräsentiert. (4)

(g) Geben Sie eine Implementierung für die Methode `value` an, die den repräsentierenden `int`-Wert zurück gibt. (3)

Aufgabe 3 (12 Punkte)

Implementieren Sie eine Enum `FSK` mit den folgenden Eigenschaften:

- Die Enum `FSK` soll folgende Werte enthalten: `Ab0`, `Ab6`, `Ab12`, `Ab16` und `Ab18`.
- Es gibt eine Objektvariable `age` in der das zur Freigabe notwendige Mindestalter gespeichert ist. Anmerkung: Sie brauchen dann einen Konstruktor um es zu setzen.
- Außerdem gibt es die Methode

  ```
  boolean allowedAtAge(final int age)
  ```

  die zu einem übergebenen Alter `true` genau dann zurück gibt, wenn das übergebene Alter für die FSK-Freigabe ausreichend ist.

Aufgabe 4 (18 Punkte)

(a) Definieren Sie eine lokale Variable und weisen Sie ihr ein `int`-Array mit Platz für 1000 `int`s zu. (2)

(b) Befüllen Sie Ihr Array so, dass es die Zahlen 1 bis 1000 **in umgekehrter Reihenfolge** enthält. (6)

(c) Definieren Sie ein zweites `int`-Array, das halb so groß ist und kopieren sie jedes zweite Element vom ersten Array in das zweite, beginnend mit dem ersten. (10)

Aufgabe 5 (12 Punkte)

Der folgende Code ist fehlerhaft und wird nicht kompiliert:

```
1  interface A {
2      private int mu(int a);
3      A ma(Ama);
4  }
5  class B extends A {
6      private final int a;
7      private B(final int a) {
8          a = 7;
9      }
10     Override
11     public int mu(int a) {
12         return a;
13     }
14     public A ma(A ma) {
15         return this;
16     }
17     A mkA() {
18         new B(5);
19     }
20 }
```

Durch Änderung von 6 Zeilen in obigem Code, wird er kompilierbar. Geben Sie jeweils die Zeilennummer an und schreiben Sie dahinter wie die Zeile **korrekt** aussehen muss.

Aufgabe 6 (11 Punkte)

Gegeben seien folgende Interfaces und Klassen:

```
interface X {
    int getX();
}
interface Y {
    int getY();
}
abstract class Z {
    abstract int getZ();
    int getX() {
        return getZ() + 5;
    }
}
class A extends Z implements Y {
    int getZ() {
        return 12;
    }
    public int getY() {
        return getX();
    }
}
class B extends A implements X {
    public int getX() {
        return 100;
    }
    int getZ() {
        return -7;
    }
}
class App {
    public static void main(String[] args) {
        A a = new A();
        System.out.println(a.getX()); //
        System.out.println(a.getZ()); //
        Z z = a;
        System.out.println(z.getX()); //
        z = new B();
        System.out.println(z.getX()); //
        System.out.println(a.getY()); //
    }
}
```

Was wird beim Ausführen der Klasse `App` ausgegeben? Schreiben Sie Ihre Antworten direkt hinter die Zeile in der die Ausgabe angestossen wird (hinter die Kommentarzeichen).