Prüfung Softwareentwicklung II (IB)

Datum	:	11.07.2016, 16:30 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 4 Aufgaben auf 9 Seiten erhalten haben.

Aufgabe	1	2	3	4	Summe
max. Punkte	11	23	33	23	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 (11 Punkte)

Gegeben seien folgende Klassen:

```
abstract class X {
    int getX() {
        return 42;
    }
}
abstract class Z extends X {
    abstract int getZ();
    public int getY() {
        return getZ() + 5;
    }
}
class A extends Z {
    private int x = 12;
    @Override public int getX() {
        return super.getX() + x;
    }
    public int getZ() {
        return 13;
    }
}
class B extends A {
    private int x = 13;
    @Override public int getX() {
        return x;
    }
    public int getZ() {
        return super.getZ() - 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.getY()); //
        z = new B();
        System.out.println(z.getX()); //
        System.out.println(z.getZ()); //
    }
}
```

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).

Aufgabe 2 (23 Punkte)

Gegeben sei die folgende abstrakte Klasse für Fahrzeuge:

```
abstract class Vehicle {
    private int speed = 0;
    public boolean hasEngine() {
        return false;
    }
    abstract public int noOfWheels();
    public final int getSpeed() {
        return speed;
    }
    public void speedUp() {
        speed++;
    }
    public void slowDown() {
        speed--;
    }
}
```

Implementieren Sie im Folgenden schrittweise eine Klasse `Bike` die als konkrete Klasse die Klasse `Vehicle` erweitert.

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

(b) Definieren Sie zwei verkettete Custom-Konstruktoren. Einer der beiden bekommt als Parameter die maximale Geschwindigkeit, der andere die maximale Geschwindigkeit und die aktuelle Geschwindigkeit. (10)

Definieren Sie auch evtl. **notwendige** Objektvariablen. Beachten Sie dabei, dass die aktuelle Geschwindigkeit bereits in `Vehicle` verwaltet wird, die Objektvariable `speed` aber nicht vererbt wird! Es ist verboten eine zusätzliche Objektvariable für die Geschwindigkeit in `Bike` zu definieren.

(c) Redefinieren Sie die beiden Methoden `speedUp` und `slowDown` so, dass die Geschwindigkeit nie unter 0 fallen und nie über die maximale Geschwindigkeit steigen kann. (8)

(d) Wenn Sie die Klasse jetzt übersetzen wollen, gibt der Compiler folgende Fehlermeldung (3)

```
Bike is not abstract and does not override...
```

Implementieren Sie alles noch notwendige sinnvoll, so dass die Klasse compiliert werden kann.

Aufgabe 3 (33 Punkte)

Gegeben sei das folgende Interface für eine Website:

```
interface Website {
    String baseUrl();
    String getPage(final String name);
    void addPage(final String name, final String content);
}
```

Implementieren Sie im Folgenden schrittweise eine Klasse `SimpleWebsite` die das Interface `Website` implementiert.

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

(b) Ein einfaches Website-Objekt speichert den Servernamen und das Protokoll als Zeichenketten in zwei Objektvariablen `servername` und `protocol`. Außerdem wird in zwei getrennten Arrays `pagenames` und `pagecontents` der Name bzw. der Inhalt einer Webpage gespeichert. Dabei findet sich der Name und der Inhalt einer Page unter dem gleichen Index. Beispiel: Der Inhalt der Seite deren Name *Impressum* den Index 5 im Array `pagenames` hat, findet sich im Array `pagecontents` beim Index 5. Ohne Einschränkung können Sie davon ausgehen, dass die Webpages paarweise verschiedene Namen haben. (8)

Darüber hinaus wird in einer Objektvariablen `nextpage` gespeichert, welcher Index als nächster befüllt werden kann, bzw. wieviele Seiten es bereits gibt.

Definieren Sie alle benötigten Objektvariablen.

(c) Definieren Sie einen Custom-Konstruktor, mit drei Parametern: `servername`, `protocol` und `maxpages`, so dass beispielsweise mit (5)

```
new SimpleWebsite("ob.cs.hm.edu", "http", 100)
```

ein Website-Objekt das maximal 100 Pages haben kann, erzeugt wird.

(d) Geben Sie eine Implementierung für die Methode `baseUrl` an, die die Basis-URL der einfachen Website zurück gibt, z.B. `http://ob.cs.hm.edu/`. (3)

(e) Geben Sie eine Implementierung für die Methode `getPage` an, die zu einem Seiten-Namen die URL gefolgt vom Inhalt zurück gibt, z.B. soll `getPage("about")` zurück geben: (9)

```
http://ob.cs.hm.edu/about
Blablabla...
```

Wobei „Blablabla...” der Inhalt sein soll.

Wird die Seite mit dem Namen nicht gefunden, soll statt dem **Inhalt** die Zeichenkette `"404 - Page not found"` zurück gegeben werden.

(f) Geben Sie eine Implementierung für die Methode `addPage` an, die eine neue Seite in die beiden Arrays einfügt, aber nur dann, wenn noch Platz ist. (6)

Aufgabe 4 (23 Punkte)

Implementieren Sie ein Schachbrett als zwei-dimensionales Array. Ein Schachbrett ist quadratisch und besteht aus 8 mal 8 Feldern. Jedes Feld ist entweder schwarz oder weiß. Links und rechts neben sowie unter und über einem schwarzen Feld sind nur weiße Felder. Diagonal bleibt die Farbe gleich.

(a) Definieren Sie eine lokale Variable und weisen Sie ihr ein zwei-dimensionales `boolean`-Array für ein Schachbrett zu. (2)

(b) Befüllen Sie Ihr Array so, dass es wie ein Schachbrett schwarze und weiße Felder enthält. Dabei steht `true` für ein schwarzes und `false` für ein weißes Feld. Beginnen Sie mit einem schwarzen Feld. (10)

(c) Definieren Sie ein `char`-Array für ein Schachbrett. In den beiden bereits vorgegebenen Variablen `rowRook` und `columnRook` ist die Position eines Turmes als Index im zweidimensionalen Array gespeichert (Sie müssen nichts umrechnen). Ein Turm kann sich von der aktuellen Position nur senkrecht und waagerecht bewegen. Befüllen Sie das Array so, dass an der Position des Turms ein `T` steht. Alle Felder die er waagerecht erreichen kann, sollen mit einem `-`, alle Felder die er senkrecht erreichen kann mit einem `|` belegt werden. In allen sonstigen Feldern steht ein Punkt. (11)

Im Beispiel für die Position 1, 5 würde die zeilenweise Ausgabe des zweidimensionalen Arrays dann so aussehen:

```
.....|..
-----T--
.....|..
.....|..
.....|..
.....|..
.....|..
.....|..
```

Achtung: Das ist nur ein Beispiel. In der Lösung müssen Sie sich auf die Variablen `rowRook` und `columnRook` beziehen.