NAMEN DER DOZENTEN: Himmelspach, Trancón Widemann

NORDAKADEMIE
HOCHSCHULE DER WIRTSCHAFT

Portfolioprüfung (Final): Einführung in die OO-Programmierung (I166)

A20a/b

QUARTAL: II/2021

Dauer: 90 Minuten Anzahl Seiten **ohne** Deckblatt: 6 **Datum:** 16.06.2021

Hilfsmittel: keine.

Bemerkungen:

- *Bitte prüfen Sie zunächst die Klausur (alle Teile) auf Vollständigkeit.*
- *Bitte vermerken Sie auf Ihren Antwortbögen folgende Angaben:*
  - *Name*
  - *Matrikelnummer*
  - *Zenturie*
  - *Seitenzahl*
  - *Aufgabennummer*
  - *ModulNr. der Lehrveranstaltung*

Es sind 85 Punkte erreichbar.

Aufgabe	Erreichbare Punkte
1	3
2	8
3	8
4	5
5	6
6	7
7	5
8	10
9	10
10	9
11	11
12	3
Summe	85

1. (3 Punkte) Schreiben Sie eine Methode `max`, die von zwei übergebenen ganzzahligen Parameterwerten das Maximum ermittelt und dieses als Ergebnis zurückgibt.

2. (8 Punkte) Schreiben Sie eine Methode `fill`, die ein neu erzeugtes Ergebnis-Array, mit einer übergebenen Größe $N > 0$, an jeder Position i im Array mit der jeweiligen Fibonacci-Zahl $fib(i)$ befüllt und dieses zurückgibt. Die Fibonacci-Zahl berechnet sich wie folgt:

$$fib(i) = \begin{cases} 0 & \text{für } i = 0 \\ 1 & \text{für } i = 1 \\ fib(i-2) + fib(i-1) & \text{sonst} \end{cases}$$

   Beispiel: a[0] ist 0; a[1] ist 1; a[2] ist 1; a[3] ist 2; ...

3. (8 Punkte) Schreiben Sie eine Methode `checkListSorting`, die eine übergebene Liste mit ganzzahligen Werten (Datentyp `Long`), auf aufsteigende Sortierung überprüft. Ist die Liste aufsteigend sortiert, so soll die Methode wahr zurückliefern, ansonsten falsch.

4. (5 Punkte) In der unten angegebenen Klasse sehen Sie eine Methode `add`. Verwenden Sie diese Methode (als externen Methodenaufruf) zur Realisierung einer Methode `mult` in einer Klasse `MultFun`. Die Methode `mult` soll zwei positive ganzzahlige Parameter verwenden und das Ergebnis der Multiplikation beider Parameter zurückgeben.

   Achtung: Verwenden Sie zur Lösung keinesfalls den * Operator von Java!

```
public class AddFun {
  public int add(int a, int b){
    return a + b;
  }
}
```

5. (6 Punkte) Was sind die Ausgaben der folgenden Algorithmen?

   1.

```
public int doSomething (int a, int b) {
        if (a−b >= 0) {
                return 1 + doSomething(a−b, b);
        }
        return 0;
}

/* Aufruf: */ System.out.println(doSomething(23, 5));
```

   => 4

   23 - 5 = 18
   18 - 5 = 13
   13 - 5 = 8
   8 - 5 = 3
   3 - 5

```
public int maxVal ( int value1 , int value2 ) {
    if ( value1 >= value2 ) {
        return value1 ;
    } else {
        return value2 ;
    }
}

public int[] fill ( int N ) {
    int [] resArr = new int [N];
    resArr [0] = 1 ;
    resArr [1] = 1 ;
    for (int i = 2 ; i < N ; i++) {
        resArr [i] = resArr [i-1] + resArr [i - 2];
    }
    return resArr ;
}

public boolean checkListSorting ( List <Long> list ) {
    int i = 0 ;
    while ( i < list.size() - 1 ) {
        if ( list.get(i) <= list.get(i+1))
            i++;
        } else {
            return false ;
        }
    }
    return true ;
}


public class MultFun {
    public int mult ( int number1 , int number2 ) {
        AddFun adder = new AddFun ();
        int prevRes = 0 ;
        for ( int i = 0 ; i < number1 ; i++ ) {
            prevRes = adder . add ( prevRes , number2 );
        }
        return prevRes;
    }
}
```

2.

```
int a = 3;
if (a % 2 != 0) {
  a = a − 5;        -> -2
  if (a < 0) {
    a *= −1;        -> 2
  }
  if (a % 2 != 1) {
    a /= 2;         -> 1
  } else {
    a *= 2;
  }
}
System.out.println(a);      => 1
```

3.

```
boolean a = true;
boolean b = true;
System.out.println( (a & !b) | (!a & b) );   => false
```

4.

```
int sum = 0;                          1   0 + 0 · 1 = 0
for (int i = 1; i < 3; i++) {         2   0 + 0 · 2 = 0
        sum += (sum*i);
}
System.out.println(sum);    => 0
```

6. (7 Punkte) Schreiben Sie eine Methode `isPrime`, die für eine übergebene ganzzahlige Zahl (ganzzahliger primitiver Datentyp) ermittelt, ob diese eine Primzahl ist. Das Ergebnis (wahr / falsch) soll an den Aufrufer zurückgegeben werden.

7. (5 Punkte) Sie stehen vor der Aufgabe alle Einträge einer Liste, die den Zustand „dead“ erreicht haben, aus derselbigen zu entfernen. Schreiben Sie dafür eine Methode mit dem Namen `removeDead`, die dies auf einer übergebenen `ArrayList<Element>` ausführt.

   Verwenden Sie zur Lösung dieser Aufgabe keine lambda-Ausdrücke, verwenden Sie eine geeignete Schleife.

```
public class Element {
  private boolean dead;
  public boolean isAlive() { return !dead; };
}
```

```
public boolean isPrime(int number) {
    if (number <= 1) {
        return true
    }
    int i = 2;
    while (i < number) {
        if (number % i == 0) {
            return false;
        }
        i++;
    }
    return true;
}

public void removeDead(ArrayList<Element> list) {
    Iterator<Element> it = list.iterator();
    while (it.hasNext()) {
        Element element = it.next();
        if (!element.isAlive()) {
            it.remove();
        }
    }
}
```

8. (10 Punkte) Finden Sie die im folgenden Quelltext mindestens zehn versteckte Logik-, Konventions-, Semantik- und Syntaxfehler. Notieren Sie die jeweilige Zeilennummer und geben Sie pro Fehler eine kurze Benennung/Fehlerbeschreibung an.

```
1  public double ComputeMean[ArrayList<int> Values] (
2
3      private Long sum := 0;
4
5      for (int i == 0; i <= Values.length(); I+=1) {
6        {
7          if (Values[i] != null) {
8            sum =+ Values[i];
9          }
10       }
11     }
12
13     return sum / values.length();
14 }
```

1 Namenskonvention camelCase für Methoden verletzt

1 Eckige statt Runden Klammern für Parameter

1 ArrayList speichert nur Objekt, keine primitiven Datentypen

3 Zuweisungsoperator ist „=" nicht „:="

3 primitiver Datentyp „Long" wird klein geschrieben

5 Zuweisungsoperator ist „=" nicht „=="

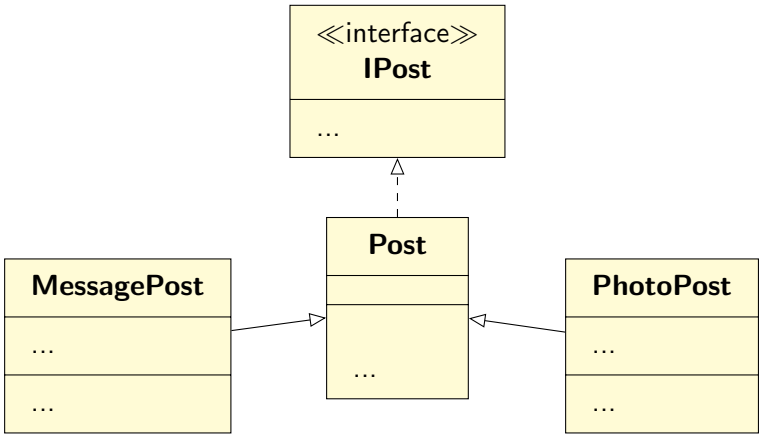
5 Mit „<=" entsteht eine Out Of Bounds Exception

5 Die Variable „i" wird kleingeschrieben

7 ArrayList werden nicht mit „[]" indiziert

8 Kombinierter Zuweisungsoperator ist „+=" nicht „=+"

9. (10 Punkte) Gegeben ist das folgende, in der Vorlesung diskutierte Klassendiagramm



Nehmen Sie an, dass die folgenden Variablendeklarationen vereinbart wurden:

```
IPost ip;
Post p = new Post("Autor");
MessagePost m = new MessagePost("author","text");
PhotoPost ph = new PhotoPost("author","filename","caption");
```

Welche der folgenden Anweisungsfolgen hat welchen Effekt? Betrachten Sie jeden der folgenden in der Tabelle angegebenen Code Schnipsel nur im Kontext der Deklarationen. Wählen Sie eine Antwort aus folgender Liste und notieren Sie die Tupel (a, b) mit $a \in \{A, \ldots, J\} \wedge b \in \{1, \ldots, 4\}$. Beachten Sie dabei: jedes Anweisungskürzel a darf nur einmal verwendet werden, nur für gegebene und richtige Antworten gibt es Punkte.

	Zuweisung	Effekt
A	p=m;	(a, 1)
B	p=ph;	(b, 1)
C	p=m; m=p;	(c, 3)
D	ph=m;	(d, 3)
E	ip=p;	(e, 1)
F	ip=m;	(f, 1)

	Zuweisung	Effekt
G	ip=null; p=ip;	(g, 3)
H	p=ph; ph=(PhotoPost)p;	(h, 1)
I	p=m; ph=(PhotoPost)p;	(i, 2)
J	p=(PhotoPost)m;	(j, 3)

Mögliche Effekte:

1. Zuweisung ohne Fehler ausführbar.
2. Zuweisung erzeugt zur Laufzeit einen Fehler.
3. Zuweisung erzeugt zur Compilezeit einen Fehler.
4. Es kann ein Laufzeitfehler auftreten, muss aber nicht.

10. (9 Punkte) Welche der folgenden Aussagen über Interfaces, Klassen und abstrakten Klassen sind richtig ?
Lösungen angeben als $(a, b)\ mit\ a \in \{A, \ldots, I\} \wedge b \in \{R, F\}$ **(R: Richtig, F: Falsch)**
Bewertung: Für jede richtige Lösung (entweder (a, F) *oder* (a, R)*) einen Punkt, für jede Falsche (oder widersprüchliche oder fehlende) 0 Punkte*

(a, R) A. Eine Subklasse ist eine Klasse, die eine andere Klasse erweitert bzw. von dieser Klasse erbt. Sie erbt alle Datenfelder und Methoden von ihrer Superklasse.

(b, F) B. Eine Subklasse ist eine Klasse, die in einer anderen Klasse als Typ einer Exemplarvariablen verwendet wird. Sie ermöglicht das Entwurfsmuster „Delegation“.

(c, F) C. Eine Superklasse ist eine Klasse, die dringend „Refactored“ werden muss. Superklassen werden als „super“ bezeichnet, da sie für viele unterschiedliche Facetten einer Software zuständig sind. Weitere Spezialisierungen dieser Klasse werden in der Regel nicht mehr vorgenommen.

(d, F) D. Konstruktoren einer Spezialisierung können zu einem beliebigen Zeitpunkt der Abarbeitung der Anweisungen ihres Konstruktors den Konstruktor der Generalisierung aufrufen.

(e, R) E. Eine Variable kann ein Objekt halten, dessen Typ entweder gleich dem deklarierten Typ der Variablen oder ein beliebiger Subtyp des deklarierten Typs ist.

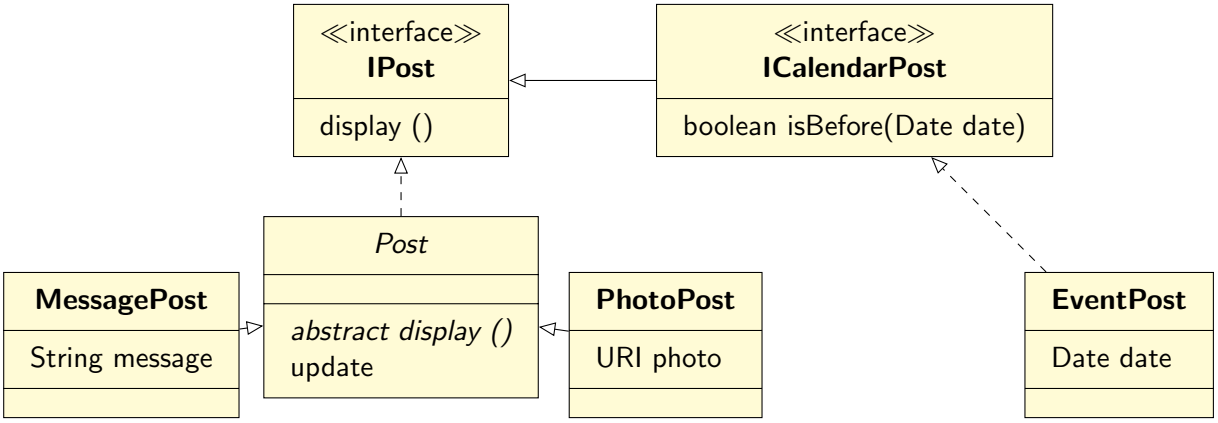
(f, R) F. Alle Klassen, die ein Interface implementieren, zählen zu dessen Subtypen.

(g, R) G. Eine Subklasse kann die Implementierung einer Methode überschreiben. Dazu deklariert die Subklasse eine Methode mit der gleichen Signatur wie in der Superklasse, implementiert diese jedoch mit einem anderen Rumpf. Die überschreibende Methode wird dann bei Aufrufen der Unterklasse vorgezogen.

(h, F) H. Methodenaufrufe in Java sind niemals polymorph. Derselbe Methodenaufruf kann immer nur eine bestimmte Methode aufrufen. Dies wird durch den deklarierten Typ der Variablen fest vorgegeben.

(i, F) I. Eine Klasse darf keine Instanzvariable definieren, die den gleichen Namen wie eine geerbte trägt.

11. (11 Punkte) Sie bekommen folgendes Klassendiagramm und sollen dafür die Klassen erzeugen. Geben Sie die Typdefinitionen der Klassen und Schnittstellen inklusive der Datenfelder und Methoden (nur Signaturen) an, die diese laut Klassendiagramm deklarieren bzw. implementieren müssen.



12. (3 Punkte) Sie stehen vor der Aufgabe alle Einträge einer Liste, die den Zustand „dead“ erreicht haben, bei der Berechnung des Gesamt-IQs zu ignorieren. Schreiben Sie dafür eine Methode mit dem Namen `computeTotalIQ`, die dies auf einem übergebenen `Stream<Element>` ausführt und den summierten „total iq“ als Ergebnis zurückgibt.

```
public class Element {
  private boolean dead;
  private int iq;
  public boolean isDead() { return dead; };
  public int getIQ() { return iq; };
}
```

```
public interface IPost {
    public void display();
}

public abstract class Post implements IPost {
    public abstract void display();
    public void update();
}

public class PhotoPost extends Post {
    private URI photo;
    public void display();
}
```

```
public class MessagePost extends Post {
    private String message;
    public void display();
}

public interface ICalendarPost extends IPost {
    public boolean isBefore();
}

public class EventPost implements ICalendarPost {
    private Date date;
    public boolean isBefore();
}
```