Faculty of Civil Engineering
Chair of Intelligent Technical Design

Weimar, 7th January 2022

Online exam WiSe 2021 SOLUTION

Object-oriented modelling and programming in engineering

Introduction

1. Please explain **three** (3) requirements of a good program and how does object-oriented programming help fulfil these requirements.

[1 point]

3 of these:

Software has to be expandable. By using self-contained classes, new functionality can be easily added by new classes

Software has to be changeable. A functionality is encapsulated in a class and needs to be changed only once.

Software has to be portable. By using interfaces, software can support multiple hardware, e.g. different keyboards.

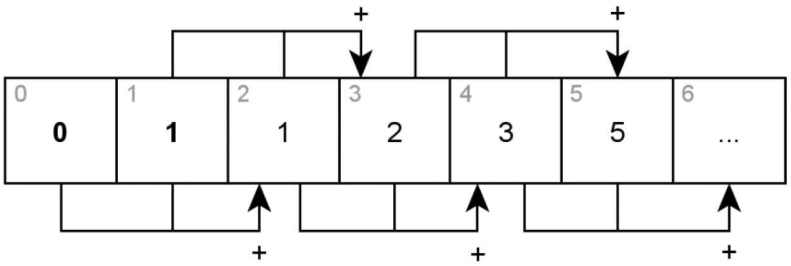
Code has to be readable. Classes structure code into objects. You have to understand the meaning of a class only once and then you understand it in all uses.

Software has to be reusable. Classes can be used in different libraries and programs.

Control structures

2. A method shall calculate the Fibonacci numbers and store them in an array. The steps below (I to VI) describe the necessary steps. The figure below shows an example for this calculation. Draw a Nassi-Shneiderman diagram to visualize the steps for this algorithm, using the steps listed below.

   I) The method is called 'fibonacci' and returns an array of integer values.
   II) An integer named 'sizeOfArray' is the input parameter. This defines the number of Fibonacci numbers to calculate. The minimum size is 2.
   III) Initialize a new array object named 'result' with the given size 'sizeOfArray'
   IV) Initialize this array's elements: the element with the index 0 with the value zero and the element with the index 1 with the value 1.
   V) Iterate over the array 'result' starting from index 2 up to (sizeOfArray – 1)
   VI) Calculate each array's element as the sum of values of the two prior elements as shown in the Figure below. The numbers depicted in the upper left corner of the boxes in the figure are the indices and the numbers in the center area are the integer values of the array's elements.



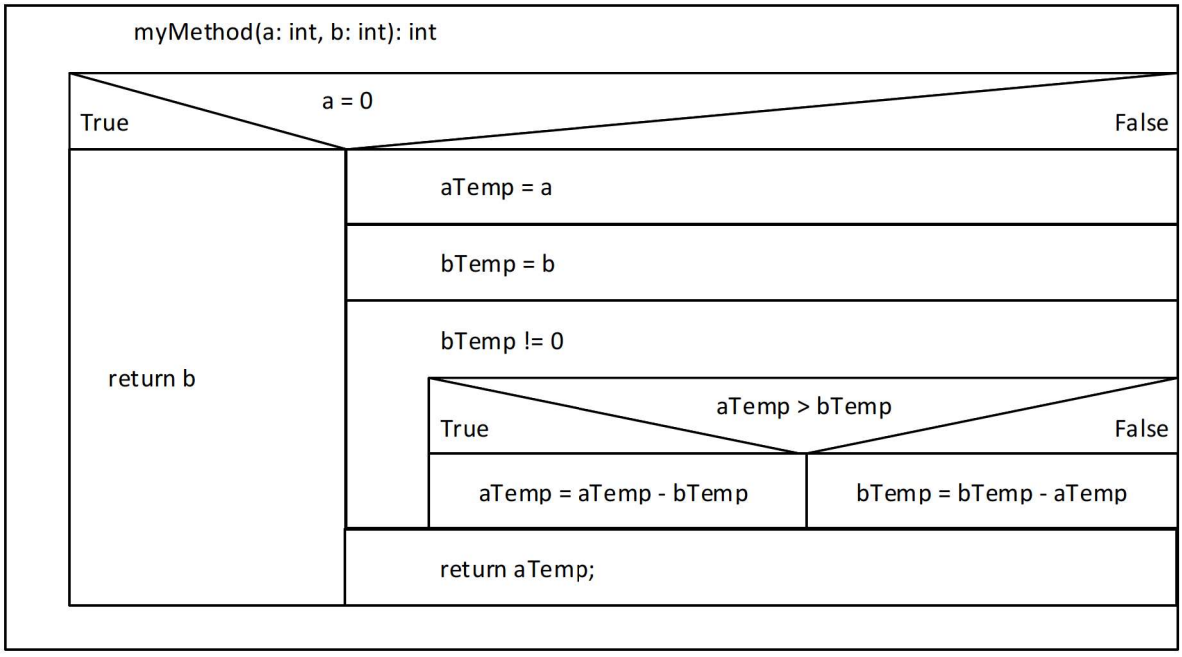
[1 point]

```
fibonacci (sizeOfArray : int) : int[ ]

    int[] result = new int [sizeOfArray]
    result [0] = 0
    result [1] = 1
    for (int i = 2; i < sizeOfArray; i++)
        result[i] = result [i - 2] + result [i - 1]
    return result
```

3. Draw a Nassi-Shneiderman diagram for the method plotted below, and name what this method is doing.

```
public int myMethod(int a, int b)
 {
    if (a == 0)
    {
        return b;
    }
    else
    {
        int aTemp = a;
        int bTemp = b;
        while (bTemp != 0)
        {
            if (aTemp > bTemp)
            {
                aTemp = aTemp - bTemp;
            }
            else
            {
                bTemp = bTemp - aTemp;
            }
        }

        return aTemp;
    }
 }
```

[1 point]



Greatest common divisor

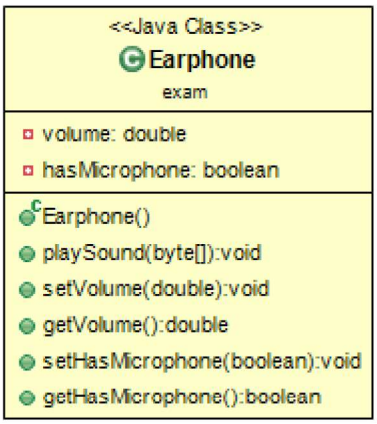
UML

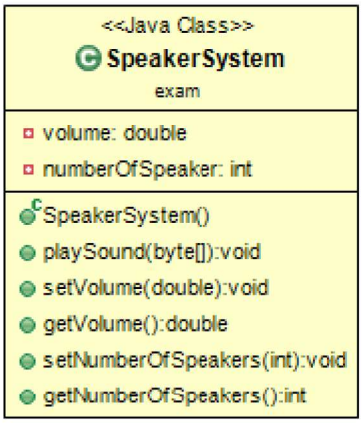
4. Have a look at the source code below. Draw the UML class diagram for the two classes. Describe how inheritance can be used to improve the depicted implementation. Draw the new UML class diagram, with appropriate symbols for inheritances.

[1 point]

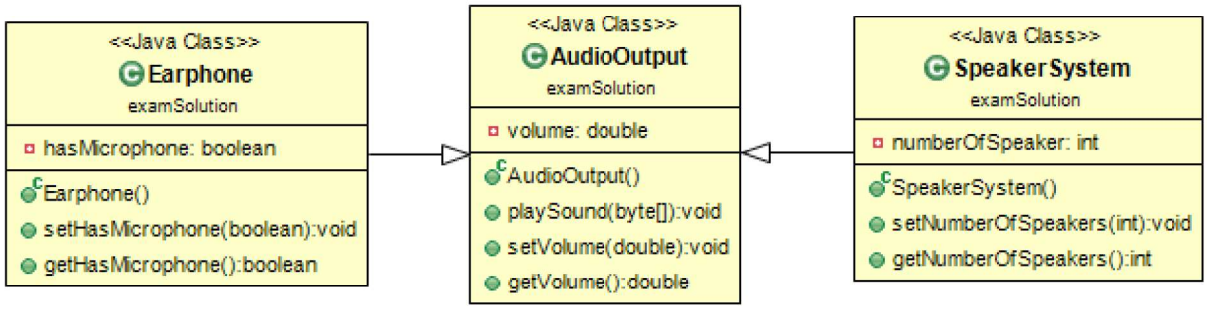
```java
public class SpeakerSystem {

    private double volume;

    private int numberOfSpeaker;

    public SpeakerSystem()
    {
        this.volume = 100;
        this.numberOfSpeaker = 2;
    }

    public void playSound(byte[] soundData)
    {
        //...
    }

    public void setVolume (double newVolume)
    {
        this.volume = newVolume;
    }

    public double getVolume ()
    {
        return this.volume;
    }

    public void setNumberOfSpeakers (int speakers)
    {
        this.numberOfSpeaker = speakers;
    }

    public int getNumberOfSpeakers ()
    {
        return this.numberOfSpeaker;
    }
}
```

```java
public class Earphone {

    private double volume;
    private boolean hasMicrophone;

    public Earphone()
    {
        this.volume = 100;
        this.hasMicrophone = false;
    }

    public void playSound(byte[] soundData)
    {
        //...
    }

    public void setVolume (double newVolume)
    {
        this.volume = newVolume;
    }

    public double getVolume ()
    {
        return this.volume;
    }

    public void setHasMicrophone (boolean hasMic)
    {
        this.hasMicrophone = hasMic;
    }

    public boolean getHasMicrophone ()
    {
        return this.hasMicrophone;
    }
}
```





- Use inheritance of a parent class



5. Draw a UML class diagram with associations and cardinalities for the class structure described below.

I) The name of the class is 'Cell'.

II) The class has 8 related 'neighbors' of type 'Cell'.

III) The class 'Cell' has an 'id' of type integer.

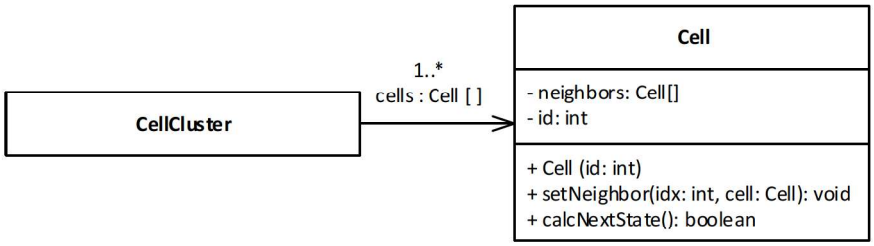
IV) For initialization (construction) at least the 'id' is needed.

V) A method with an integer as parameter for setting each individual neighbor is needed.

VI) The method 'calcNextState', which doesn't take parameters, returns a Boolean value.

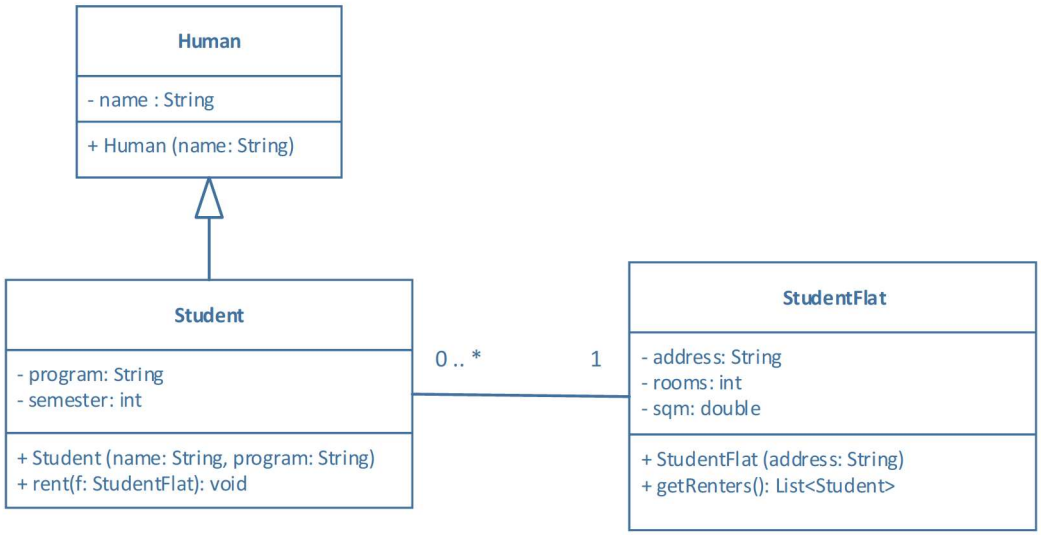
VII) Another class with the name 'CellCluster' has a private attribute 'cells' which is an array of elements of type Cell. The attribute 'cells' contain at least a single 'Cell'.

[1 point]



Writing Source Code

6. Write the source code in Java based on the UML diagram depicted below. Note, that the implementation code for the methods' bodies is not needed.



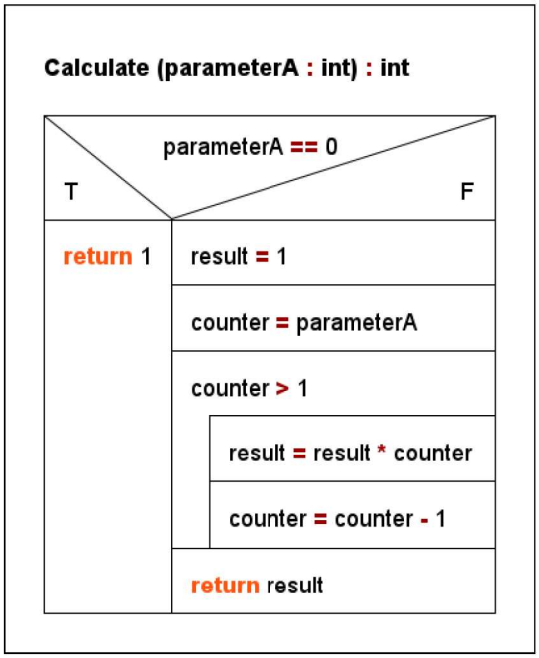
[1 point]

```java
public class Human {
    private String name;

    public Human(String name)
    {

    }

}
```

```java
public class Student extends Human {

    private String program;
    private int semester;
    private StudentFlat flat;

    public Student(String name, String program) {


    }

    public void rent (StudentFlat f)
    {

    }

}
```

```java
public class StudentFlat {

    private String address;
    private int rooms;
    private double sqm;
    private ArrayList<Student> students;

    public StudentFlat(String address)
    {

    }

    public       List<Student> getRenters()
    {

    }

}
```

7. Write a method in Java that is described in the Nassi-Shneiderman diagram below, and name what this method calculates.



[1 point]

```
public int Calculate          (int parameterA)
{
    if (parameterA == 0)
    {
        return 1;
    }
    else
    {
        int result = 1;
        int counter = parameterA;
        while (counter > 1)
        {
            result = result * counter;
            counter = counter - 1;
        }
        return result;
    }
}
```

- Factorial