PANIKRUNDE 2017

24.02.2017

Panikrunde 2017 - Überblick

10:00 – 11:30 – Kontrollstrukturen, Funktionen und Rekursion

11:30 – 12:30 – Objektorientierung

12:30 – 13:00 – Pause

13:00 – 14:30 – Arrays, Collections und Generics

14:30 – 16:00 - Streams und Exceptions

1. Kontrollstrukturen, Funktionen und Rekursion

10:00 – 11:30

Übung 1

```
public static void main(String[] args) {
    int zahl = Integer.parseInt(args[0]);
    int quersumme1 = berechneQuersummeIterativ(zahl);
    System.out.println("Die iterativ-berechnete Quersumme ist: " + quersumme1);
    int quersumme2 = berechneQuersummeRekursiv(zahl);
    System.out.println("Die rekursiv-berechnete Quersumme ist: " + quersumme2);
}
```

1) Schreibe eine Methode, die die Quersumme zu einer gegebenen Zahl berechnet und die zu folgendem Programm passt:
   a) Iterativ (mit Schleifen)
   b) Rekursiv (ohne Schleifen)

```
public static void main(String[] args) {
    java.util.Scanner sc=new java.util.Scanner(System.in);
    String text=sc.nextLine();
    System.out.println(mirrorText(text));
}
```

2) Schreibe eine Methode, die eine Zeichenkette spiegelt (z.B. Test123 -> 321tseT) und die zu folgendem Programm passt:
   a) Iterativ (mit Schleifen)
   b) Rekursiv (ohne Schleifen)

```
public static void main(String[] args) {
    java.util.Scanner sc=new java.util.Scanner(System.in);
    int zahl=sc.nextInt();
    System.out.println(ungeraderTeiler(zahl));
}
```

3) Schreibe eine Methode, die den größten ungeraden Teiler einer Zahl findet und zu folgendem Programm passt:

Finde die Fehler

Finde alle sieben Fehler, die die Kompilierung verhindern.

In jeder Zeile ist maximal ein Fehler.

```
1   java.util.Scanner;
2
3   class Fehlersuche {
4
5       public static void main(String args[]) {
6           Scanner scan=new Scanner();
7
8           System.out.println("Please enter a number:");
9           int a=scan.nextInt();
10          boolean a=isPrime(a);
11          if(a)System.out.println(a++ " is prime!");
12          else{
13              System.err.println(a + " is not prime!");
14
15      }
16
17      static public bool isPrime(int z){
18          for(int i=2;i<=Math.sqrt(z);i++){
19              if(0==z%i){
20                  return false;
21              }
22          }
23
24      }
25  }
```

2. Objektorientierung

11:30 – 12:30

Übung 2

Verbessere die folgende Klassenstruktur mithilfe der Möglichkeiten, das die du in der Vorlesung gelernt hast. Befolge dabei folgende Aufgaben:

a) Entwickle eine geeignete Klassenhierarchie unter Nutzung aller Kniffe (Vererbung, Super-Konstruktoren, Polymorphismus, Abstrakte Klassen…)
b) Erstelle ein UML-Diagramm basierend auf den Ergebnissen aus a).
c) Übersetze das UML-Diagramm 1:1 in Quellcode und überprüfe, ob deine Verbesserungen das Klassen-Modell wie gewünscht verbessert haben.
d) Erweitere das Modell geeignet um die Aspekte:
    a) *Schulleiter*
    b) *Klasse/Stufe*
    c) *Unterrichtsfach*
    d) *Schule*

Übung 2

```
public class Hausmeister{
    protected int alter;
    public String name;

    public void arbeiten(){
        System.out.println(name + " repariert alles.");
    }
}
public class Lehrer{
    protected int alter;
    public String name;
    private String fach;
    private boolean istNett;

    public void arbeiten(){
        System.out.println(fach+"unterricht!");
    }
    public boolean gibtHausaufgaben(){
        return !istNett;
    }
    public void setFach(String fach){
        this.fach = fach;
    }
    public String getFach(){
        return fach;
    }
}
```

```
public class Schueler{
    protected int alter;
    public String name;
    protected String[] lieblingsfach;

    public void lernen(String fach){
        for(String s : lieblingsfach)
            if(s.equals(fach))
                System.out.println(name + " lernt " + fach);
    }
    public boolean magLehrer(Lehrer l){
        if(l.gibtHausaufgaben() == false)
            return true;
        else{
            for(String s : lieblingsfach)
                if(l.getFach().equals(s))
                    return true;
            return false;
        }
    }
}
```

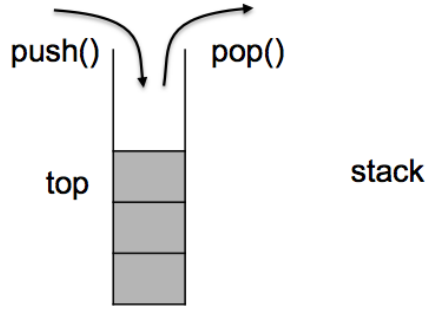
http://doodle.com/poll/vxeq29p4tdpkbtux

PAUSE BIS 13:00

3. Arrays, Collections und Generics

13:00 – 14:30

Übung 3



Schreibe eine eigene Stack-Klasse MyStack, die auf Arrays basiert und dynamisches Hinzufügen und Löschen von Objekten unterstützt.

Ein Stack ist eine Datenstruktur, die als „Stapel“ abspeichert und immer nur ein Objekt oben auf den Stapel legen oder vom Stapel runternehmen kann. Zudem hat unser Stack eine maximale Größe, die bei Initialisierung festgelegt wird.

Folgende Funktionen sollte die Klasse unterstützen:

MyStack(int size)	: Kontruktor, der die maximale Größe des Stacks festlegt
push(Object o)	: Legt das Objekt *o* oben auf den Stack, oder wirft eine Fehlermeldung wenn kein Platz mehr ist
pop(): Object	: Entfernt das oberste Element und gibt es zurück
size()	: Gibt die aktuelle Anzahl an Elementen zurück.

Weiterführende Aufgaben:

a) Nutze Generics, um nur passende Elemente zu speichern.

b) Nutze eine ArrayList oder DeepCopy, um einen Stack ohne Limit zu erstellen.

4. Exceptions und Streams

14:30 – 16:00

Übung 4

Schreibe ein Programm, dass Texte in Konsonantenschrift umwandelt, also alle Vokale (auch Umlaute) entfernt. Bsp: "Kannst du dies lesen?" -> "Knnst d ds lsn?". Befolge dafür folgende Aufgaben:

a) Lies eine Text-Datei *name.txt* ein. (Achte dabei auf alle möglichen auftretenden Fehler)

b) Wandle den Text in Konsonantenschrift um.

c) Speichere den Text in eine neue Datei, die dann *name_processed.txt* heißen sollte. Du musst also eine neue Datei erstellen.

d) Erweitere nun die Lösung, dass eine *„ProcessingForbiddenException“* auftritt, sofern im Text irgendwo die Zeichenkette „NoProcessing!“ vorkommt. Die Ausnahme soll außerdem Auskunft über den Ort der Zeichenkette geben und die Ausgabe wie folgt aussehen:

```
Exception in thread "main" ProcessingForbiddenException: Processing was forbidden at index 15!
```

VIEL ERFOLG FÜR DIE KLAUSUR!