Technology
Arts Sciences
TH Köln

ALGORITHMEN UND PROGRAMMIERUNG I
WS 2020/2021
Prof. Dr. Frank Victor

Praktikum 7 Abgabe bis 31.01.2021

Name, Vorname: Özkurt, Cihat
Mat.-Nr.: 11148632
GMID: inf2323
Datum: 31.01.2021
Betreuer im Praktikum: Gross, Julien

Aufgabe 1: Programmieren in Java

Schreiben Sie ein Java-Programm, das x^y berechnet, wenn die Zahlen `x` und `y` eingegeben werden. Verwenden Sie hierzu die Methode `pow` der Java Klasse `Math`.

Hinweise:

- Sie müssen sich in der Java API die Methode `pow` ansehen, um zu sehen, wie sie aufgerufen wird. Wie findet man die Klasse `Math` in der Java 13 API?
- Wenn Sie ohne `import` arbeiten wollen, dann funktioniert das mit `Math.pow(x,y)`.
- Wenn Sie `import` verwenden möchten, damit der Aufruf nur noch `pow(x,y)` heißt, versuchen Sie es mit `import static java.lang.Math.*;`

Die Ausgabe des Programms sollte in etwa so aussehen:

```
Programm zur Berechnung der Potenz
Bitte geben Sie x ein: ___________
Bitte geben Sie y ein: ___________
Die Potenz von x hoch y ist: ______ .
```

Lösung:

<Kopieren Sie bitte Ihr Programm hier hin. >

```
import java.util.Scanner;
import static java.lang.Math.*;

public class Main {

   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      int x,y;
      System.out.print("x=");
      x = scanner.nextInt();
      System.out.print("y=");
      y = scanner.nextInt();
      System.out.println("Die Potenz von x hoch y ist: " + pow(x,y));
   }

}
```

Aufgabe 2: Programmieren in Java

Definieren Sie eine Klasse **`Queue`**. Eine **`Queue`** (deutsch: Schlange) ist eine Datenstruktur, bei der man das erste Element entnehmen kann (**`dequeue`**). Ein Datenelement kann nur an das Ende der Schlange angefügt werden (**`enqueue`**). Zusätzlich sind in der Klasse die Initialisierung (über den Konstruktor) und die Abfragen, ob die Schlange leer (**`is_empty()`**) bzw. voll (**`is_full()`**), definiert. Die Schlange soll **Integerzahlen** aufnehmen können und hat die folgenden Instanzvariablen:

```
private int nextFree;
private int[] arr;
```

Dabei gibt **`nextFree`** den Index der ersten freien Stelle in der Schlange an und das Array enthält die Elemente der Schlange. Freie Plätze werden enthalten eine 0.

Beispiel:

Schlange[5]: 10 20 30 0 0 **`nextFree = 3`**

(1) s.dequeue() lässt 10 die Schlange verlassen:
20 30 0 0 0 **`nextFree = 2`**

(2) s.enqueue(100) fügt 100 an das Ende der Schlange ein:
20 30 100 0 0 **`nextFree = 3`**

(3) s.dequeue()
s.dequeue()
s.dequeue()
s.is_empty() liefert **`true`**
0 0 0 0 0 **`nextFree = 0`**

Neben der Implementierung der Klasse **`Queue`** besteht Ihre Aufgabe darin, eine Anwendung zu schreiben, die eine Warteschlange für Integerzahlen simuliert. Formulieren Sie in der Anwendung 3 Szenarien so wie im Beispiel dargestellt, die zeigen, dass die Schlange ordnungsgemäß funktioniert und Fehlermeldungen wie „Schlange voll“ und „Schlange leer“ ausgibt.

Lösung:

<Kopieren Sie bitte Ihr Programm hier hin. >

```
public class Queue {
   int nextFree;
   private int[] arr;

   public Queue(){
      arr = new int[5];
      int [] arr = {0,0,0,0,0};
   }
    public void enqueue(int n){
      if(is_full(arr) == false){
        for(int i = 0; i < 5; i++)
            if(arr[i] == 0){
               arr[i]=n;
               nextFree++;
               for(int j = 0; j < 5; j++)
                  System.out.print(arr[j] + " ");
               System.out.print(" -> nextFree = " + nextFree + "\n");
               return;
            }
      }
      else
          System.out.println("Schlange ist voll");
   }

    public void dequeue(){
```

```
        if(is_empty(arr) == false){
          for(int i = 0; i < nextFree-1; i++){
              arr[i] = arr[i+1];


          }
          arr[nextFree-1] = 0;
          nextFree--;
          for(int j = 0; j < 5; j++)
                  System.out.print(arr[j] + " ");
              System.out.print(" -> nextFree = " + nextFree + "\n");
        }
        else
          System.out.println("Schlange ist leer");

    }

    public boolean is_full(int a[]){
        for(int i = 0; i < 5; i++){
          if(arr[i] == 0)
              return false;
        }
        return true;
    }
    public boolean is_empty(int a[]){
        for(int i = 0; i < 5; i++)
          if(arr[i] != 0)
              return false;
        return true;
    }

}



import java.util.ArrayList;

public class Main {


   public static void main(String[] args) {
      Queue q = new Queue();
      q.enqueue(10);
      q.enqueue(20);
      q.enqueue(30);
      q.enqueue(40);
      q.enqueue(50);
      q.enqueue(60);

      q.dequeue();
      q.dequeue();
      q.dequeue();
      q.dequeue();
      q.dequeue();
      q.dequeue();

      q.enqueue(10);
      q.enqueue(20);
      q.dequeue();

   }

}
```

Aufgabe 3: Programmieren in Java

Verwenden Sie die **`Queue`** und die Anwendung aus Aufgabe 2 in dieser letzten Aufgabe. Sie können die Schlange und die Anwendung kopieren und nach Ihren Bedürfnissen anpassen.
Die Aufgabe besteht darin, einen Taxistand zu implementieren und zu simulieren.
Es gibt insgesamt **8 Taxis** in der Stadt und **1 Taxistand**. Der Taxistand funktioniert als FIFO Struktur (Schlange, Queue, s.o.).

Der folgende Code kann für Sie hilfreich sein:

Die Klasse `Taxi` ist vorgegeben:

```
/**
 * Ein Taxi ist ein Objekt, das aus dem Fahrernamen, dem Kennzeichen und
 * einer eindeutigen Nummer besteht.
 */
public class Taxi {
    private String namefahrer;
    private String kennzeichen;
    private int nummer;

    /**
     * Der Konstruktor legt ein Taxi Objekt an und speichert darin den
     * Fahrernamen, das Kennzeichen und die Taxinummer.
     */
    public Taxi(String namefahrer, String kennzeichen, int nummer) {
        this.namefahrer = namefahrer;
        this.kennzeichen = kennzeichen;
        this.nummer = nummer;
    }

    /**
     * Diese Methoden verwenden wir zur Ausgabe der Meldungen in der Klasse
     * Schlange.
     */
    public String getnamefahrer() {
        return namefahrer;
    }

    public String getkennzeichen() {
        return kennzeichen;
    }

    public int getnummer() {
        return nummer;
    }
}
```

Die Klasse `Schlange` hat die folgenden Instanzvariablen und den folgenden Konstruktor:

```
public class Schlange {
    private int nextFree;
    private Taxi[] arr;

    public Schlange(int nextFree) {
      this.nextFree = nextFree;
      arr = new Taxi[5];
      …
    }
```

…

Die Klasse `App` hat folgenden Aufbau:

```
public class App {
    public static void main(String args[]) {
        Taxi a = new Taxi(…);
        Taxi b = new Taxi(…);
        …
        Schlange taxistand = new Schlange(0);       // Nächster freier Platz
                                                    // ist im Index 0

        System.out.println("Ausgangssituation");
        taxistand.clear();      // leert den Taxistand
        taxistand.ausgeben();   // zeigt den Taxistand (siehe Ausgabe)

        System.out.println("\1. Situation");
        taxistand.clear();      // leert den Taxistand
        taxistand.enqueue(a);
        taxistand.enqueue(b);
        taxistand.enqueue(c);
        taxistand.enqueue(d);
        taxistand.enqueue(e);
        taxistand.enqueue(f);
        taxistand.ausgeben();
        …
```

Die **Ausgabe** sieht dann in etwa so aus:

```
Ausgangssituation
Alle 5 Plätze sind leer.

Taxistand
frei frei frei frei frei

1. Situation
Alle 5 Plätze sind leer.
Das Taxi: 1, Frank Victor, BN – FV 300 fährt auf Platz 1
Das Taxi: 2, Angela Merkel, B - DE 001 fährt auf Platz 2
Das Taxi: 3, James Bond, BN - JB 007 fährt auf Platz 3
Das Taxi: 4, Manuel Neuer, M - MN 001 fährt auf Platz 4
Das Taxi: 5, Angelique Kerber, BN - AK 111 fährt auf Platz 5
Fehler: Das Taxi: 6, Boris Becker, M - BB 4911 kann nicht einfahren! Der Taxistand ist picke packe voll!

Taxistand
1 2 3 4 5
```

Hinweise:

- Die Schlange des Taxistands und damit das Array enthält Objekte, und zwar Taxis. Wenn ein Platz nicht besetzt ist, verwenden Sie bitte die **`null`**-Referenz. Sie könnten also schreiben `arr[nextFree] = null;` wenn ein Platz frei wird.
- Die Methode `public void enqueue(Taxi x)` ist ganz leicht zu schreiben. Wenn der Stand nicht voll ist, wird `arr[nextFree] = x;` gesetzt. Sonst wird eine Meldung ausgegeben (siehe Ausgabe).
- Die Methode `public void dequeue()` ist etwas aufwändiger. Sie gibt aus, welches Taxi den Stand verlassen hat. Die anderen Taxis, die warten, müssen nachrücken.
- Verwenden Sie bitte `javadoc` Kommentare für die Klassen und Methoden!

Lösung:

<Kopieren Sie bitte Ihr Programm hier hin. >

```
public class Taxi {
    private String namefahrer;
    private String kennzeichen;
    private int nummer;
 /**
 * Der Konstruktor legt ein Taxi Objekt an und speichert darin den
 * Fahrernamen, das Kennzeichen und die Taxinummer.
 */
 public Taxi(String namefahrer, String kennzeichen, int nummer) {
    this.namefahrer = namefahrer;
    this.kennzeichen = kennzeichen;
    this.nummer = nummer;
 }

 /**
 * Diese Methoden verwenden wir zur Ausgabe der Meldungen in der Klasse
 * Schlange.
 */
 public String getnamefahrer() {

    return namefahrer;
 }

 public String getkennzeichen() {
    return kennzeichen;
 }

 public int getnummer() {
    return nummer;
 }
}
```

```
public class Schlange {
    private int nextFree,reserveFree;
    private Taxi[] arr;
    private Taxi[] reserve_array;
    int capacity = 5;
    int capacity2 = 3;


    public Schlange(int nextFree){
        this.nextFree = nextFree;
        arr = new Taxi[5];
        reserve_array = new Taxi[3];
    }
    public void enqueue(Taxi x){
        if(is_full(arr) == false){
            for(int i = 0; i < 5; i++)
                if(arr[i] == null){
                    arr[i] = x;
                    nextFree++;
                    return;
                }
        }if(nextFree>=5)
            for(int i = 0; i < 3; i++)
                if(reserve_array[i] == null){
                    reserve_array[i] = x;
                    nextFree++;
                    reserveFree++;
                    return;
                }
    }

    public void clear(){
        for(int i=0;i<5;i++){
            arr[i] = null;
            if(i<3)
                reserve_array[i]=null;
        }
        System.out.println("Alle 5 Plätze sind leer.");
    }
```

```
    public void dequeue(){

        if(is_empty(arr) == false){
            for(int i = 0; i < capacity - 1; i++){
                arr[i] = arr[i+1];
            }
            arr[capacity - 1] = reserve_array[0];

            //arr[nextFree-1] = reserve_array[0];
           for(int i = 0; i < capacity2 - 1; i++){
                reserve_array[i] = reserve_array[i+1];
            }
            reserve_array[capacity2-1] = null;
            reserveFree--;
            nextFree--;
        }
    }
    public boolean is_full(Taxi t[]){
        for(int i = 0; i < 5; i++){
            if(arr[i] == null)
                return false;
        }
        return true;
    }
    public boolean is_empty(Taxi t[]){
        for(int i = 0; i < 5; i++)
            if(arr[i] != null)
                return false;
        return true;
    }

    public void ausgeben(){

         for(int i = 0; i < capacity; i++)
            if(arr[i] != null)
            System.out.println("Das Taxi " + (i+1) + " "+ arr[i].getnamefahrer() + ", " + arr[i].getkennzeichen() + " fährt auf Platz " + (i+1));
         for(int i = 0; i < reserveFree ; i++)
            if(reserve_array[i] != null)
            System.out.println("Fehler: Das Taxi " + (capacity+i+1) + ", " +  reserve_array[i].getnamefahrer() + ", " + reserve_array[i].getkennzeichen() + " kann nicht einfahren! Der Taxistand ist picke packe voll!");

         System.out.println("Taxistand");
         for(int i = 0; i < 5; i++){
            if(arr[i] == null)
                System.out.print("frei ");
            else
                System.out.print((i+1 + " "));
         }
         System.out.println("");

    }

}









public class Main {
    public static void main(String[] args) {
        Taxi a = new Taxi("Frank Victor"," BN – FV 300 " , 1);
        Taxi b = new Taxi("Angela Merkel", "B - DE 001",2);
        Taxi c = new Taxi("James Bond", "BN - JB 007",3);
        Taxi d = new Taxi("Manuel Neuer", "M - MN 001",4);
        Taxi e = new Taxi("Angelique Kerber", "BN - AK 111", 5);
        Taxi f = new Taxi("Boris Becker", " M - BB 4911", 6);
        Taxi g = new Taxi("Al Pacino", " 34 - GS 1905", 6);
        Taxi h = new Taxi("Queen Elizabeth ", " Eng - Q 1900", 6);

        Schlange taxistand = new Schlange(0);

        System.out.println("Ausgangssituation");
        taxistand.clear(); // leert den Taxistand
        System.out.println("");
        taxistand.ausgeben(); // zeigt den Taxistand (siehe Ausgabe)
        System.out.println("\n1. Situation");
        taxistand.clear(); // leert den Taxistand

        taxistand.enqueue(a);
        taxistand.enqueue(b);
        taxistand.enqueue(c);
        taxistand.enqueue(d);
        taxistand.enqueue(e);
        taxistand.enqueue(f);

        taxistand.ausgeben();

        System.out.println("\n**************************************\n");
        System.out.println("2. Situation");

        taxistand.enqueue(g);
        taxistand.enqueue(h);
        taxistand.ausgeben();

        System.out.println("\n**************************************\n");
```

```
    System.out.println("3. Situation");

    taxistand.dequeue();
    taxistand.dequeue();
    taxistand.dequeue();
    taxistand.dequeue();
    taxistand.dequeue();
    taxistand.ausgeben();
  }

}
```

Aufgabe 4: Programmieren in Java

Zur Lösung dieser Aufgabe ist etwas Recherche-Arbeit von Ihnen zu leisten. Besorgen Sie sich Informationen über das Thema **Binäre Suche**. Schauen Sie dazu in die Literatur oder ins Internet. Beschreiben Sie bitte zunächst auf einer Seite, wie die Binäre Suche funktioniert und welchen Aufwand die Methode hat (O-Notation).

Entwickeln und testen Sie bitte eine Java-Methode **`binsearch`**, die eine binäre Suche auf einem **`int`**-Array realisiert. Die Methode gibt die Position eines gesuchten Wertes im **`int`**-Array zurück bzw. -1, wenn der Wert nicht vorkommt.

Lösung:

<Kopieren Sie bitte Ihr Programm hier hin. >

```
import java.util.Scanner;

public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      int[] a = {1,2,4,5,6,8,11,20,30,50};

      int first = 0;
      int last = a.length-1;
      int middle = (first+last)/2;
      System.out.print("n = ");
      int n = scanner.nextInt();

      while(first <= last){
         if(n > a[middle])
            first = middle + 1;
         else if(n == a[middle]){
            System.out.println("Suchende Zahl wurde gefunden " + n + " -> " + (middle+1)+  " index");

            break;
         }
         else
            last = middle-1;
         middle = (first+last)/2;
      }
      if(first > last)
         System.out.println("Suchende Zahl nich gefunden");
   }

}
```