

Programmierung 2

Klausur SS 2019

Klausur Programmierung 2

Aufgabe 1

(15P)

Multiple Choice Fragen (auch mehrere Antworten können richtig sein):

- a) Kann eine rekursive Funktion auch iterativ berechnet werden? (1P)
- ☐ Ja, immer
 - ☐ Nein, nie
 - ☐ Manchmal
- b) Welche Typen von inneren Klassen gibt es in Java? (1P)
- ☐ Static Classes
 - ☐ Inner Classes
 - ☐ Static Nested Classes
 - ☐ Static Anonymous Classes
 - ☐ Local Anonymous Classes
 - ☐ Static Inner Classes
 - ☐ Local Classes
 - ☐ Static Local Classes
 - ☐ Nested Classes
 - ☐ Anonymous Classes
 - ☐ Nested Inner Classes
 - ☐ Inner Anonymous Classes
- c) Worauf kann man aus einer inneren Klasse heraus zugreifen? (1P)
- ☐ Auf public Member der umgebenen Klasse
 - ☐ Auf private Member der umgebenen Klasse
 - ☐ Auf protected Member der umgebenen Klasse
- d) Was sind generelle Vorteile der Objektorientierten Programmierung? (1P)
- ☐ Wiederverwendbarkeit
 - ☐ geringer Overhead
 - ☐ Erweiterbarkeit
 - ☐ Performance
 - ☐ Wartbarkeit
- e) Welche der folgenden Lambda Ausdrücke sind syntaktisch korrekt? (1P)
- ☐ (a, b) -> a+b
 - ☐ (a, b) -> return a+b
 - ☐ (a, b) -> return a+b;
 - ☐ (a, b) -> (return a+b;)
 - ☐ a, b -> a+b
 - ☐ int a, int b -> a+b
 - ☐ a, b -> a+b;
 - ☐ (int a, int b) -> a+b

f) Gegeben sei folgender Code: (1P)

```
class Container<T extends Number> {  
    private T content;  
    public Container(T arg) {  
        content = arg;  
    }  
    public Container(Container<T> other){  
        this.content = other.getContent();  
    }  
    public T getContent(){ return this.content; }  
}
```

Welche der folgenden Instantiierungen ist korrekt?

- ☐ Container<Integer> e1 = new Container<Integer>(new Integer(42));
- ☐ Container<Number> e2 = new Container<Number>(c1);
- ☐ Container<Number> e3 = new Container<Integer>(c1);
- ☐ Container<Integer> e4 = new Container<Integer>(c1);

g) Welche der folgenden Eigenschaften besitzt C? (1P)

- ☐ hohe Typsicherheit
- ☐ manuelle Speicherverwaltung
- ☐ plattformunabhängigkeit
- ☐ prozedurale Programmierung
- ☐ funktionale Programmierung

h) Mithilfe welchen Operators kann man in C die Größe einer Variablen abfragen? (1P)

- ☐ #
- ☐ &
- ☐ *
- ☐ getsize
- ☐ getSize
- ☐ sizeof
- ☐ getSizeOf
- ☐ varsize

i) Durch welchen Operator erhält man in C die Adresse einer Variablen? (1P)

- ☐ #
- ☐ +
- ☐ @
- ☐ §
- ☐ &
- ☐ *
- ☐ **
- ☐ getaddr
- ☐ stdio
- ☐ ->
- ☐ >>
- ☐ addr

- j) Welche der folgenden Datentypen gibt es in C? (1P)
- ☐ string
 - ☐ float
 - ☐ byte
 - ☐ short int
 - ☐ int *
 - ☐ boolean
 - ☐ long long
- k) Gehen Sie davon aus, in einem C-Programm gibt es einen Integer int i. Welche Verwendung von scanf zum einlesen eines integers ist korrekt? (1P)
- ☐ scanf("%d", *i);
 - ☐ scanf("%d", &i);
 - ☐ scanf("%d", i);
 - ☐ scanf("%d", #i);
 - ☐ scanf(i);
 - ☐ scanf("i");
 - ☐ scanf("%i");
- l) Gehen Sie davon aus, in einem C-Programm gibt es einen Integer int i. Welche Verwendung von printf zum Ausgeben des integers ist korrekt? (1P)
- ☐ printf("%d", *i);
 - ☐ printf("%d", &i);
 - ☐ printf("%d", i);
 - ☐ printf("%d", #i);
 - ☐ printf(i);
 - ☐ printf("i");
 - ☐ printf("%i");
- m) Gegeben sei folgender Ausschnitt eines C-Programms (1P)
- ```
long l = 140732807899964;
long* p1 = &l;

printf("%p\n", &p1);
printf("%p\n", &p1);
printf("%p\n", *p1);
```
- Angenommen der Code produziert folgende Ausgabe:
- ```
0x7ffee9ca2b38
0x7ffee9ca2b30
0x7ffee905eb3c
```
- Welche ist die Adresse der Variablen l?
- ☐ 0x7ffee9ca2b38
 - ☐ 0x7ffee9ca2b30
 - ☐ 0x7ffee905eb3c

n) Gegeben sei folgender Code:

```
01 public class Sorter {
02     public Integer countDigits(Integer i){
03         return String.valueOf(i).length();
04     }
05     public int findMax(List<Integer> input, Function<Integer, Integer> f){
06         int max = 0;
07         for(int i = 0; i<input.size(); i++)
08             if(f.apply(input.get(i)) > f.apply(input.get(max)))
09                 max = i;
10         return max;
11     }
12     public static void main(String[] args){
13         Sorter s = new Sorter();
14         a.findMax(new ArrayList<Integer>(), a::countDigits);
15         a.findMax(new ArrayList<Integer>(), Integer::highestOneBit);
16     }
17 }
```

- i. Welche Form einer Methoden-Referenz ist in Zeile 14 gegeben? (1P)
- ☐ Referenzen zu statischen Methoden
 - ☐ Referenzen zu Instanz-Methoden eines gegebenen Typs
 - ☐ Referenzen zu Instanz-Methoden eines Objekts
 - ☐ Referenzen zu Konstruktoren
- ii. Welche Form einer Methoden-Referenz ist in Zeile 15 gegeben? (1P)
- ☐ Referenzen zu statischen Methoden
 - ☐ Referenzen zu Instanz-Methoden eines gegebenen Typs
 - ☐ Referenzen zu Instanz-Methoden eines Objekts
 - ☐ Referenzen zu Konstruktoren

Aufgabe 2

(12P)

Implementieren Sie die beiden Funktionen `findRecursive` und `findIterative` der Klasse `SubstringFinder`. Beide Funktionen sollen prüfen, ob ein gegebener Teilstring `needle` in dem String `haystack` enthalten ist. Ist dies der Fall soll `true` zurück gegeben werden, andererseits `false`. Sie dürfen keine Methoden der `String`-Klasse verwenden mit Ausnahme von `length`, `equals`, `substring` und `charAt`.

```
public class SubstringFinder {  
    public boolean findRecursive(String haystack, String needle) {
```

```
}
```

```
public boolean findIterative(String haystack, String needle) {
```

```
    }  
}
```

Aufgabe 3

(20P)

Gegeben seien die unten aufgeführten Klassen `Car` und `CarPortal` zur Verwaltung der in einem Gebrauchtwagen-Portal angebotenen Wagen. Implementieren Sie die Methoden der `CarPortal`-Klasse mit folgender Funktionalität:

- a) Konstruktor: Initialisiert die Datenstruktur `cars`. Diese speichert `Car`-Objekte unter einer ID. Wählen Sie die Datenstruktur so, dass die `Car`-Objekte in der Reihenfolge, in der Sie hinzugefügt wurden, abgerufen werden können. (2P)
- b) `public void add(Integer id, Car car)`: Fügt `cars` unter der gegebenen ID einen neuen Wagen hinzu. (1P)
- c) `public void remove(Integer id)`: Entfernt den Wagen mit der gegebenen ID aus `cars`. (1P)
- d) `public List<Car> getCarsSorted(Comparator<Car> c)`: Gibt eine mit dem gegebenen `Comparator` sortierte Liste der Wagen zurück. Implementieren Sie zum Sortieren den Bubblesort-Algorithmus:

```
procedure bubbleSort(A : list of sortable items )  
    n = length(A)  
    for i = 1 to n-1 inclusive do  
        for j = n-1 downto 1 inclusive do  
            if A[j-1] > A[j] then  
                swap( A[j-1], A[j] )  
            end if  
        end for  
    end for  
end procedure
```

 (4P)
- e) `public List<Car> getCarsFilteredAndSorted(Comparator<Car> c, Predicate<Car> p)`: Gibt eine durch das Prädikat `p` gefilterte und durch den `Comparator c` sortierte Liste der Wagen zurück. Implementieren Sie diese Methode mit Hilfe eines **parallelen Streams**. (3P)
- f) `public Map<String, List<Car>> getCarsFilteredAndGrouped(Predicate<Car> p)`: Filtert die Bestandsliste mit Hilfe des Prädikats `p` und gibt eine nach Markennamen gruppierte Map der gefilterten Bestandsliste zurück. Implementieren Sie diese Methode mit Hilfe eines **parallelen Streams**. (3P)
- g) `public long countCars(Predicate<Car> p)`: Gibt die Anzahl der Wagen die das gegebene `Predicate` erfüllen zurück. Implementieren Sie diese Methode mit Hilfe eines **parallelen Streams**. (2P)

Implementieren Sie in der unten aufgeführten Klasse `CarPortalTest` folgendes und nutzen Sie dabei Lambda Ausdrücke:

- (h) Zählen Sie alle gelben *Lamborghini*s. (2P)
- (i) Fragen Sie eine nach `horsePower` sortierte Liste aller *Ferraris* ab. (2P)


```
public class Car {
    private String color, brand;
    private int horsepower;

    public Car(String color, String brand, int horsepower){
        this.color = color;
        this.brand = brand;
        this.horsePower = horsepower;
    }

    public String getColor(){ return this.color; }

    public String getBrand(){ return this.brand; }

    public String horsepower(){ return this.horsePower ; }
}
```

```
import java.util.*;
import java.util.function.*;
import java.util.stream.*;

public class CarPortal {
    private Map<Integer, Car> cars;

    public CarPortal() {

    }

    public void add(Integer id, Car car){

    }

}
```

```
public void remove(Integer id){
```

```
}
```

```
public List<Car> getCarsSorted(Comparator<Car> c){
```

```
}
```

```
public List<Car> getCarsFilteredAndSorted(Comparator<Car> c, Predicate<Car> p){

}

public Map<String, List<Car>> get CarsFilteredAndGrouped(Predicate<Car> p){

}

public long countCars(Predicate<Car> p){

}

}
```

```
public class CarPortalTest {  
    public static void main(String[] args) {  
        CarPortal portal = new CarPortal();  
        String[] colors = {"black", "red", "green", "blue", "yellow"};  
        String[] brands = {"Ferrari", "Lamborghini", "Bugatti", "Maserati",  
                           "Pagani"};  
        java.util.Random ran = new java.util.Random();  
        for(int i=0; i<100; i++)  
            portal.add(i, new Car(colors[ran.nextInt(5)],  
                                  brands[ran.nextInt(5)],  
                                  100*ran.nextInt(300)));  
    }  
}
```

Aufgabe 4

(24P)

Implementieren Sie eine generische Klasse `Dictionary`. Der Rumpf der Klasse ist unten angegeben. Vervollständigen Sie die Klasse wie im Folgenden beschrieben. Achten Sie insbesondere darauf die Methodensignaturen entsprechend zu vervollständigen.

Die `Dictionary`-Klasse soll ein generisches Wörterbuch darstellen, welches Paare aus Einträgen vom Typ `K` und Übersetzung vom Typ `V` speichert. Wählen Sie eine Datenstruktur, die sicherstellt, dass die Elemente im Wörterbuch nach den Einträgen sortiert vorliegen. Die `Dictionary`-Klasse soll folgende Methoden besitzen:

- a) `Konstruktor`: Initialisiert die Datenstruktur. (2P)
- b) `insert`: Diese Methode nimmt eine Liste von Einträgen und eine Übersetzung entgegen und fügt dieses Paar dem Wörterbuch hinzu. (2P)
- c) `insertAll`: Diese Methode nimmt eine Liste von Einträgen und eine Liste von Übersetzungen entgegen. Es wird davon ausgegangen, dass beide Listen so sortiert sind, dass der Eintrag am Index `i` zur Übersetzung am Index `i` passt. Es sind nur Listen mit Subtypen von `K` bzw. `V` zulässig. Die Methode gibt `true` zurück, wenn das Einfügen erfolgreich war, ansonsten `false`. (3P)
- d) `getTranslation`: Gibt die Übersetzung zu einem als Parameter übergebenen Eintrag zurück. (1P)
- e) `printAll`: Gibt alle Einträge mit den dazugehörigen Übersetzungen in sortierter Reihenfolge auf dem Bildschirm aus. **Wichtig**: Nutzen Sie dazu einen passiven Iterator. (3P)
- f) `removeIf`: Nimmt ein Prädikat entgegen und löscht alle Einträge, auf die das Prädikat zutrifft. (2P)

Erweitern Sie außerdem die unten angegebene Klasse `Translation` wie folgt:

- g) Stellen Sie sicher, dass Objekte der Klasse `Translation` nach dem String `language` sortiert werden können. (2P)

Implementieren Sie weiterhin in der `main`-Funktion der Klasse `DictionaryTest` folgendes:

- h) Initialisieren Sie ein `Dictionary`-Object, welches einem Wort vom Typ `String` eine Menge von Übersetzungen vom Typ `Translation` in verschiedene Sprachen zuordnet. D.h. `K` ist vom Typ `String` und `V` ist eine Menge von `Translation`-Objekten. Die Menge der Übersetzungen für jedes Wort soll nach der `language` sortiert gespeichert werden. Wählen Sie eine entsprechende Datenstruktur. (3P)
- i) Fügen Sie dem `Dictionary` eine Übersetzung für das Wort *Hallo* in die Sprachen *French*(*Salut*) und *English*(*Hello*) hinzu. (2P)
- j) Fragen Sie von dem `Dictionary` die Übersetzung für das Wort *Hallo* ab, filtern Sie die französische Übersetzung heraus und geben Sie diese in Großbuchstaben aus. **Implementieren Sie dies mithilfe eines Streams.** (2P)

- k) Löschen Sie alle Einträge die mit H beginnen aus dem Dictionary. (2P)
Nutzen Sie dazu die Methode `removeIf` der Dictionary-Klasse.

Wichtiger Hinweis: Immer wenn in dieser Aufgabe ein Funktionales Interface zu implementieren ist, lösen Sie dies mit einem Lambda-Ausdruck, nicht mit einer anonymen Klasse.

```
import java.util.*;  
public class Dictionary {  
    // Deklarieren Sie Ihre Datenstruktur hier:
```

```
    public Dictionary() {
```

```
    }
```

```
    public          insert(  
    {
```

```
    }
```

```
    public boolean insertAll(  
    {
```

```
    }
```

```
public          getTranslation(  
{
```

```
}
```

```
public void printAll(){
```

```
}
```

```
public void removeIf(Predicate<K> p){
```

```
}
```

```
}
```



```
import java.util.*;
public class DictionaryTest {
    public static void main(String[] args){
```

```
    }
}
```

Aufgabe 5

(15P)

- a) Vervollständigen Sie das unten aufgeführte Programm. Die Funktion findMax soll das Maximum in einem übergebenen Array finden und über einen Pointer zurück geben. (6P)

```
Void findMax(                ) {
```

```
}
```

```
int main() {  
    int numbers[] = { 23, 54, 17, 69, 42 };  
    int *pMax;  
    int count = sizeof(numbers)/sizeof(numbers[0]);  
    findMax(numbers, count, &pMax);  
    printf("max: %d\n", *pMax);  
    return 0;  
}
```

b) Welche Ausgabe produziert folgendes C-Programm?

(5P)

```
#include<stdio.h>
typedef struct {
    int i;
}someStruct;

void dosomething(int i, int arr[j], int k, int *p, someStruct s) {
    i++; arr[0]++; k++; (*p)++; s.i++;
}

int main(){
    int i = 0;
    int arr[3] = {0,0,0};
    someStruct s = { 0 };
    dosomething(i, arr, arr[1], &arr[2], s);
    printf("%d, %d, %d, %d, %d", i, arr[0], arr[1], arr[2], s.i);
    return 0;
}
```

c) Definieren Sie in C einen Datentyp Person mit folgenden Eigenschaften: Name
Geburtsdatum, Geschlecht, Wohnort. Wählen Sie möglichst effiziente Datentypen.

(4P)