PIB-PRG2 Klausur SS2020

<u>Wichtige Hinweise</u>:

1. Diese ist nicht die originale Datei und enthält die Fragen der PRG2-Klausur SS2020 nicht wörtlich. Hier bekommt ihr also nicht alle Fragen wie in der echten Klausur dargestellt, wenn ihr aber mit den hier geschriebenen Fragen euch komfortable fühlt, dann seid ihr für ein gutes Teil (mind. 50%) der Klausur vorbereitet.
2. In der vorherigen Klausuren kamen ganze Aufgaben über Maps, Collections, Streams und Generics. Dort sollte man z.B. eine generische Klasse mithilfe von Mappers (Maps), Streams und Collections bauen. => **lernt Collections, Maps und Streams auch wenn sie hier nicht befördert waren.**

Punkteverteilung:

Aufgabe 1	Aufgabe 2	Aufgabe 3	Aufgabe 4	Aufgabe 5	Aufgabe 6	Summe
12	8	9	20	18	21	88

Aufgabe 1

Es gab hier 12 Multiple-Choice-Fragen bei jeder Frage kann/können eine/mehrere Antworte richtig sein. Eure Lösung ist nur dann richtig, wenn ihr alle richtige Antworte und keine falsche Antwort gekreuzt habt. Die Multiple-Choice-Fragen findet ihre in irgendeine Klausur vor dieser. Unsortiert kam z.B.

1. Kann eine Rekursive Funktion auch Iterativ berechnet werden -> Ja immer
2. Address-Operator in einer Variable C -> &
3. Mittels was kann die Größe einer Variable in C ermittelt werden -> sizeof
4. Welche davon sind syntaktisch gültige Lambda Ausdrücke?
5. Welche Instanziierungen aus der folgenden generischen Klasse sind richtig (kompilieren)?
6. Welche Vererbungs-Beziehungen zwischen den folgenden generischen Klassen/Interfaces sind existieren?
7. Wo werden die rekursive Aufrufe einer rekursiven Funktion gespeichert -> Auf dem Stack
8. Welche der Folgenden inneren Klassen sind nur innerhalb eines Anweisungsblocks gültig -> local classes
9. Welche Eingenschaften besitzt C -> manuelle Speicherverwaltung+ prozedurale Programmierung
10. Was sind die Vorteile von Rekursiven Funktionen -> Geeignet für Graphen und Bäume + Kürzere Formulierung

Guckt euch die Klausur von SS2019 für bessere Ansicht nach.

Aufgabe 2

1) Instanziiere Objekten von folgenden Klassen: A, B und C

```
Public class A{
    public class B{}
    public static class C{}
    Public static void main(String[] args){
        A a = new A();
        A.B ab = a.new B();
        A.C ac = new A.C();
    }}
```

2) Definiere Generics, nenne Vorteile Von Generics und wie sind generische Klassen/Methoden in Java gebaut?

Vorteil : Verbesserte Typsicherheit ,
Bei Generics gibt man keinen bestimmten Datentyp an, sondern kann Klassen/Funktionen vom Typ <T t> initialisieren, wobei t einen beliebigen typen annehmen kann. Mittels Wildcards kann man sich auf bestimmte Oberklassen von Datentypen beschränken. (Numbers / Integer/ Double ..)

```
Public <T extends Number> void someMethod(){...}
Public class GenericBox <T>{
        private T item;
        public T getGenericBox{return item}
        public T setGenericBox(T item){this.item = item;}
}
```

3) Vervollständigen Sie folgende Codes so dass Die Werte aller x auf dem Bildschirm ausgegeben werden.

```
public class outerClass{
    public int x = 0;
    class innerClass{
        public int x = 1;
        void print(int x){
            System.out.println(x);              //2
            System.out.println(this.x);         //1
            System.out.printn(outterClass.x);   //0
        }
    }
    OutterClass.innerClass ic = new OutterClass().new innerClass();
    ic.print(2);
}
```

Aufgabe 3

1) Implementieren Sie folgende Funktion Rekursiv:

```
public Double iterate(Double x0, int n, UnaryOperator<Double> f) {

}
```

Die Methode Iterate bekommt eine reele Zahl x0, natürliche Zahl n und eine Funktion f: R -> R.
Iterate muss n-mal rekursive aufgerufen werden und in jedem Schritt gilt: $X_{i+1} = f(X)$

Lösung:

```
public Double iterate(Double x0, int n, UnaryOperator<Double> f) {
    if(n > 0) {
        x0 = f.apply(x0);
        return iterate(x0, n-1, f);
    }
    return x0;
}
```

2) Rufen Sie die Methode iterate() mit folgenden Funktionen:
   a. f(x) = 2x
   b. $f(x) = x^2$

Hinweis: verwenden Sie Lambda Ausdrücke und wählen Sie x0 und n beliebig.

Lösung:

```
iterate(1, 10, x -> 2*x);
iterate(1, 10, x -> x*x);
```

Aufgabe 4

Hier werden einige Methoden implementiert/vervollständigt aus einer Klasse namens **Sorter**.

1) Implementieren Sie die Methode sortBy() mittels der Selection-Sort Algorithmus wobei **input** Liste der Werte ist und f die Funktion, die als Sortierkriterium gilt (d.h. anstatt für x,y aus **input** Sortiere nach **f.apply(x)** und **f.apply(y)**). **Hinweis**: hier bekamen wir der Pseudocode von Selection-Sort.

```
public void sortBy(ArrayList<Integer> input, Function<Integer, Integer> f) {

}
```

Lösung:

```
public void sortBy(ArrayList<Integer> input, Function<Integer, Integer> f) {
    int n = input.size() -1;
    int left = 0;
    do {
        int min = left;
        for(int i=left+1; i<n; i++){
            if(f.apply(input.get(i)) > f.apply(input.get(min))) {
                min = i;
            }
        }
        swap(input, min, left); // wird in der nächsten Teilaufgabe implementiert
    } while(left < n);
}
```

2) Implementieren Sie die swap() Methode mit folgenden Deklaration:

```
public void swap(ArrayList<Integer> input, int i, int j) {

}
```

Dieser muss die ArrayList-Elemente an der Stellen i und j vertauschen.

Lösung:

```
public void swap(ArrayList<Integer> input, int i, int j) {
    Integer temp = input.get(i);
    input.set(i, input.get(j));
    input.set(j, temp);
}
```

3) Sei **list** eine gültige ArrayList<Integer>. Rufen Sie für die folgenden Funktionen (als ein Sortierkriterium) jeweils die Methode sortBy() mit jeweils ein Lambda Ausdrück und eine Methodenreferenz:
   a. f(x) = digitSum(x) // QuerSumme
   b. f(x) = digitCount(x) // AnzahlZiffern
   c. f(x) = x

   Hinweis: digitSum() und digitCount() sind hier gegeben und ein Objekt namens **s** ist auch bereits instanziiert worden und kann zum Aufruf von sortBy() verwendet werden.

   Lösung:

```
s.sortBy(list, x -> s.digitSum(x));
s.sortBy(list, Sorter::digitSum);

s.sortBy(list, x -> s.digitCount(x));
s.sortBy(list, Sorter::digitCount);

s.sortBy(list, x -> x);
s.sortBy(list, Integer::intValue);
```

Aufgabe 5

1) Erläutern Sie wie aus einem C-Programmcode eine ausführbare Datei entsteht (Nennen + Erklären) und Beschreiben Sie wie man in einer Datei Funktionen aus anderen Dateien aufrufen kann.
   a. 3 Schritte: siehe Folien von C
      i. Preprocessing
      ii. Compiling
      iii. Linking
   b. Durch #include <name.h> für std Header Dateien und #include „name.h“ für eigene Header Dateien.

2) Gehen Sie davon aus, dass die Startadresse des Arrays 0x1075bb018 ist.

```c
#include <stdio.h>
int arr[] = {10, 11, 12};
int val;
int *pi;

void p(){
    printf("arr:");
    for (int i = 0; i < sizeof(arr) / sizeof(arr[0]); ++i)
        printf(" %d", arr[i]);
    printf(" val: %d pi: %p *pi: %d\n", val, pi, *pi);
}
int main() {
    pi = arr;

    val = *pi;      p();
    val = *pi++;    p();
    val = *++pi;    p();
    val = --*pi;    p();
    val = (*pi)++;  p();

    return 0;
}
```

Lösung (Ausgabe):

```
arr: 10 11 12 val: 10 pi: 0x601040 *pi: 10
arr: 10 11 12 val: 10 pi: 0x601044 *pi: 11
arr: 10 11 12 val: 12 pi: 0x601048 *pi: 12
arr: 10 11 11 val: 11 pi: 0x601048 *pi: 11
arr: 10 11 12 val: 11 pi: 0x601048 *pi: 12
```

3) Geben Sie die Ausgabe folgender Code:

```c
int a = 11, b = 22;

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    swap(&a, &b);
    printf("a=%d b=%d", a, b);
    return 0;
}
```

Lösung (Ausgabe):

```
a=22 b=11
```

Aufgabe 6

1) Implementieren Sie die Methode findMax, diese soll der Maximum in dem gegebenen Array finden und durch einen Pointer zurückgeben.

```
void findMax(                    ) {

}

int main() {
    int numbers[] = {2, 15, 3, 69, 25, 30};
    int count = sizeof(numbers)/sizeof(numbers[0]);
    int *pMax;
    findMax(numbers, count, &pMax);
    printf("*pMax: %d", *pMax);
}
```

Lösung:

```
void findMax(int nums[], int n, int **pMax) {
    *pMax = nums;
    int i;
    for(i=0; i<n; i++) {
        if(nums[i] > **pMax) {
            *pMax = &nums[i];
        }
    }
}
```

2) Implementieren Sie die Methode swap(), diese vertauscht zwei gegebene Int-Pointers.

```
void swap(int *a, int *b) {

}
```

Lösung:

```
void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

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

Lösung:

```
typedef struct {
    char name[50];
    struct {
        int d, m, y;
    } birthdate;
    char sex;
    char city[50];

} Person;
```