Aufgabenstellung

Erstellen Sie eine Klasse Person mit den Attributen Name (String) und Alter (int). Die Methode toString() soll überschrieben werden.

(a) Erstellen Sie eine Container-Klasse PersonContainer, die neben Ihrem Namen (mit beliebigem Alter) auch die folgenden Person-Objekte als Liste zurückliefert: [Meier,40] [Müller, 55] [Beyer, 37] [Albrecht, 42] [Hansen, 56]. Wird Ihr Name nicht zurückgegeben, dann 0 Punkte! Sehen Sie sich dazu die Schnittstelle List in der Java-API an. (10 Punkte)

(b) Erstellen Sie eine MainUI_b-Klasse, in der die folgende Funktionalität *imperativ* realisiert ist:

- Filtern: Alle Personen ausgeben, deren Name auf »er« endet
- Filtern: Alle Personen ausgeben, deren Name mit »M« beginnt
- Sortieren: Nach Name aufsteigend
- Sortieren: Nach Alter aufsteigend
- Reduzieren: Älteste Person
- Reduzieren: Person mit längstem NamenVerwenden Sie dafür *keine* funktionale Programmierung. Nutzen Sie dafür das Grundgerüst MainUI1. (30 Punkte)

c) Realisieren Sie in der Klasse MainUI_c die Funktionalität aus b) mit funktionaler Programmierung (nicht imperativ!). Nutzen Sie dafür ebenfalls das Grundgerüst MainUI1. (30 Punkte)

(d) Fassen Sie in der Klasse MainUI_d die zusammenhängenden Methoden zur Filterung, Sortierung und Reduktion in jeweils eine Methode zusammen. Nutzen Sie dafür das Grundgerüst MainUI2. (30 Punkte)

```
public class MainUI1
{
    private static List<Person> liste = PersonContainer.getListe();


    public static void main(String[] args){
        //Filtern: Alle Personen ausgeben, deren Name auf "er" endet
        for(Person p : filtern1()) System.out.println(p);
        System.out.println("");


        //Filtern: Alle Personen ausgeben, deren Name mit "M" beginnt
        for(Person p : filtern2()) System.out.println(p);
        System.out.println("");


        //Sortieren: Nach Name aufsteigend
        for(Person p : sortieren1()) System.out.println(p);
        System.out.println("");
```

```
    //Sortieren: Nach Alter aufsteigend
    for(Person p : sortieren2()) System.out.println(p);
    System.out.println("");

    //Reduzieren: Älteste Person
    System.out.println(reduzieren1());
    System.out.println("");

    //Reduzieren: Person mit längstem Namen
    System.out.println(reduzieren2());
    System.out.println("");
}

private static List<Person> filtern1(){
    //Implementieren
    return null;
}

private static List<Person> filtern2(){
    //Implementieren
    return null;
}

private static List<Person> sortieren1(){
    //Implementieren
    return null;
}

private static List<Person> sortieren2(){
    //Implementieren
    return null;
}

private static Person reduzieren1(){
    //Implementieren
    return null;
}
```

```
    private static Person reduzieren2(){
        //Implementieren
        return null;
    }
}

public class MainUI2
{
    private static List<Person> liste = PersonContainer.getListe();

    public static void main(String[] args){
        //Filtern: Alle Personen ausgeben, deren Name auf "er" endet
        //Implementieren: Aufruf filterMethod und Iteration über Ergebnis
        System.out.println("");

        //Filtern: Alle Personen ausgeben, deren Name mit "M" beginnt
        //Implementieren: Aufruf filterMethod und Iteration über Ergebnis
        System.out.println("");

        //Sortieren: Nach Name aufsteigend
        //Implementieren: Aufruf sortedMethod und Iteration über Ergebnis
        System.out.println("");

        //Sortieren: Nach Alter aufsteigend
        //Implementieren: Aufruf sortedMethod und Iteration über Ergebnis
        System.out.println("");

        //Reduzieren: Älteste Person
        //Implementieren: Aufruf reduceMethod und Ausgabe des Ergebnisses
        System.out.println("");

        //Reduzieren: Person mit längstem Namen
        //Implementieren: Aufruf reduceMethod und Ausgabe des Ergebnisses
        System.out.println("");
    }

    private static List<Person> filterMethod(Predicate<Person> predicate){
        //Implementieren
        return null;
```

```
    }

    private static List<Person> sortedMethod(Comparator<Person> comparator){
        //Implementieren
        return null;
    }

    private static Person reduceMethod(BinaryOperator<Person> op){
        //Implementieren
        return null;
    }
}
```