Programming III
Laboratory 5

Objectives

- classes, interfaces, inheritance
- collections

Exercises

1. Create the following class hierarchy:

   a) A class *Airplane* that has the following attributes: producer, code, number of flights, fuel capacity

   b) From *Airplane* class derive classes *FightAriplain* that has the following characteristics: can or cannot camouflage and weapon capacity; *LineArplain* that has the following characteristic: maximum number of passengers and *EntertaimentAirplain* hat has the following characteristics: current owner and a list of previous owners.

   d) *Line* and *entertainment* airplanes implement also the interface *LuxuryOptions* that contain

   information about the fact if a plain has: noise-cancelling headphones, personal touch screen TV for each paseger

   Requirements:

   1. Create the class hierarchy described

   2. Create a list (*LinkedList* or *ArrayList*) of airplanes and display it.

   3. Create and display a map that contains the type of airplane and for each type count how many

   airplanes are in the list. Display the map in the following format: planeType number of plays displayed like and arrays of strars

   a) FightAirplane **

   b) LineArpline *****

   c) EntertainmentAirplane ***

   4. Display from the list of airplanes the ones that have noise canceling headphones

   5. Find and display the EntertaimentAirplain that had the most owners

   6. Display the average fuel capacity of all air-plains from the list, display the average fuel capacity of fight airplanes

2. Write a generic method to exchange the positions of two different elements in an array.

3. Write a generic method to count the number of elements in a collection that have a specific property (for example, odd integers, prime numbers, palindromes).

4. Design a class that acts as a library for the following kinds of media: book, video, and newspaper. Provide one version of the class that uses generics and one that does not. Feel free to use any additional APIs for storing and retrieving the media.

5. Joe Mocha is defining an interface Appendable that includes an append method. He then defines two classes, MyString and MyList, which both implement Appendable. He wants Java's type system to allow a MyString to be appended to a MyString, and a MyList to be appended to a MyList, but not MyString to a MyList, or a MyList to a MyString. Here is his definition of Appendable:

```
interface Appendable {
```

```
        Appendable append(Appendable a);
    }
```

What is wrong with this definition? What is a correct one? Also write a definition for a classes MyString and MyList and uses the revised definition of Appendable.

Homework deadline 2 weeks

1. Create the following class hierarchy

a) (4p) (2p)Create a class *Club* that has a name, year of founding and an address and from it derive the classes *FootballClub* that has like properties the team members list and an trainer; and the class

NightClub that has like properties a timetable for functioning. (1p)Each football club *member* has the following attributes: name, salary and contract expire date (define expire date using java Date class). (1p)For storing the club timetable propose an suitable abstract data structure.

b) (1p) Create a list of clubs and display them

c) (1p) For all years in which a club was founded create a map that stores the year

and the number of clubs founded in that year. Display the map like a bar diagram (something similar with the diagrams from class exercise)

d) (1p) Find the football club that has the lower number members with the contract expire date near the current date

e) (1p) Find the night club that has the longest opening time (has the longest time during

the week when is opened)

f) (1p) Find the football club that has the minimum average salary

g)(1p) Create a function that allows to display only collection of objects of type Club

h)(1p-bonus) Sort the initial list of clubs base on clubs name if 2 clubs have the same name sort them by founding year.

OBS: Implement each requirement in it's own function