Workshop #6: **Collection framework**

Learning Outcomes:

Upon successful completion of this workshop, you will have demonstrated the abilities to:

- Practice the ArrayList class.
- Practice some basic functions : add, update, remove, display, search,… in the list
- Describe to your instructor what you have learned in completing this workshop.

Total: [10 points]

In this demonstration, ArrayList is used to store a list of students. Student details include: Code, name, mark. A menu is supported for user choosing one operation at a time:

(1) Add new student

(2) Search a student based on his/her code

(3) Update name and mark of a student based on his/her code

(4) Remove a student based on his/her code

(5) List all students.

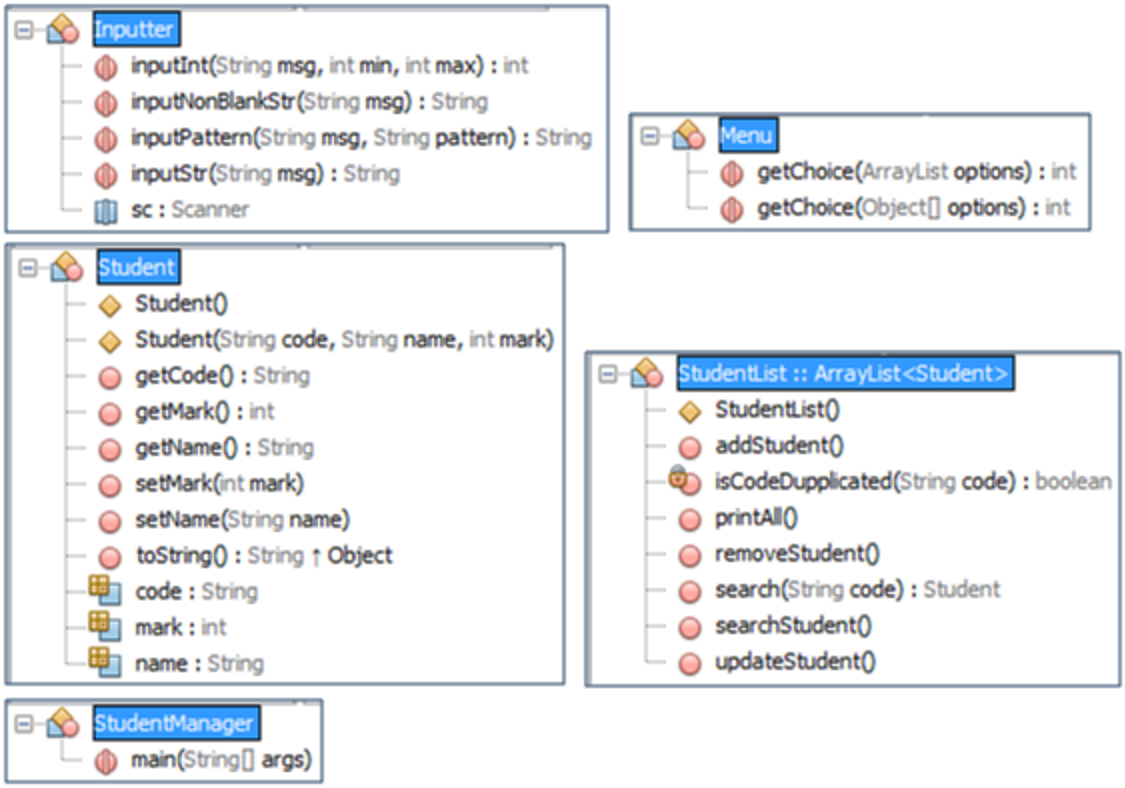
Constraints:

Student code must be in format <S000>, 0 is a digit.

Student name can not be blank

Mark: 0..10

Design:


Inputter
inputInt(String msg, int min, int max) : int
inputNonBlankStr(String msg) : String
inputPattern(String msg, String pattern) : String
inputStr(String msg) : String
sc : Scanner
Menu
getChoice(ArrayList options) : int
getChoice(Object[] options) : int
Student
Student()
Student(String code, String name, int mark)
getCode() : String
getMark() : int
getName() : String
setMark(int mark)
setName(String name)
toString() : String ↑ Object
code : String
mark : int
name : String
StudentList :: ArrayList<Student>
StudentList()
addStudent()
isCodeDupplicated(String code) : boolean
printAll()
removeStudent()
search(String code) : Student
searchStudent()
updateStudent()
StudentManager
main(String[] args)

Implement:

```java
// Class for inputting data using constraints
import java.util.Scanner;
public class Inputter {
    public static Scanner sc = new Scanner(System.in);
    // Get an integer between min... max
    public static int inputInt(String msg, int min, int max){
        if (min>max) {
            int t = min; min=max; max=t;
        }
        int data;
        do{
            System.out.print(msg);
            data = Integer.parseInt(sc.nextLine());
        }
        while (data<min || data>max);
        return data;
    }
    // Get a string with no condition
    public static String inputStr (String msg){
        System.out.print(msg);
        String data = sc.nextLine().trim();
        return data;
    }
```

```
24      // get a non-blank string
25      public static String inputNonBlankStr (String msg){
26          String data;
27          do{
28              System.out.print(msg);
29              data = sc.nextLine().trim();
30          }
31          while (data.length()==0);
32          return data;
33      }
34      // Get a string following a pattern
35      public static String inputPattern (String msg, String pattern){
36          String data;
37          do{
38              System.out.print(msg);
39              data = sc.nextLine().trim();
40          }
41          while (!data.matches(pattern));
42          return data;
43      }
44  }// Inputter class
```

```
1   // Class for a menu from a pre-defined options
2   import java.util.ArrayList;
3   import java.util.Scanner;
4   public class Menu {
5       public static int getChoice(ArrayList options){
6           for (int i=0; i<options.size(); i++){
7               System.out.print((i+1) + "-" + options.get(i));
8           }
9           System.out.print("Choose 1.." + options.size() + ": ");
10          Scanner sc = new Scanner(System.in);
11          return Integer.parseInt(sc.nextLine());
12      }
13      public static int getChoice(Object[] options){
14          for (int i=0; i<options.length; i++){
15              System.out.println((i+1) + "-" + options[i]);
16          }
17          System.out.print("Choose 1.." + options.length + ": ");
18          Scanner sc = new Scanner(System.in);
19          return Integer.parseInt(sc.nextLine());
20      }
21  }// Menu class
```

```java
// Class for a student
public class Student {
    String code="";  String name= ""; int mark=0;
    // constructors
    public Student() {
    }
    public Student(String code, String name, int mark) {
        this.code=code.toUpperCase();
        this.name=name.toUpperCase();
        this.mark = (mark>=0 && mark<=10)? mark: 0;
    }
    // Get data as a string for printing
    @Override
    public String toString(){
        return code + ", " + name + ", " + mark;
    }
    public String getCode() {...3 lines }
    public String getName() {...3 lines }
    public void setName(String name) {
        name = name.trim().toUpperCase();
        if (name.length()>0)this.name = name;// check validity
    }
    public int getMark() {...3 lines }
    public void setMark(int mark) {
        if (mark>=0 && mark <=10) this.mark = mark; // check validity
    }
}// Student class
```

```java
/* Class for a student list */
import java.util.ArrayList;
public class StudentList extends ArrayList<Student>{
    public StudentList() { // Default Constructor
        super();
    }
    // Search a student based on student's code. Return the student found
    // This method supports preventing code duplications
    public Student search(String code){
        code= code.trim().toUpperCase();
        for (int i=0; i<this.size(); i++) // Linear search is used.
            if (this.get(i).getCode().equals(code)) return this.get(i);// found
        return null; // not found
    }
    // checking whether a code is duplicated or not?
    private boolean isCodeDupplicated (String code){
        code= code.trim().toUpperCase();
        return search(code)!=null;
    }
```

```java
    // Add new student
    public void addStudent(){
        // Input data of new student
        String newCode, newName;
        int newMark;
        boolean codeDuplicated= false;
        do {// pattern: s000 or S000 ==> Pattern: "[sS][\\d]{3}"
          newCode = Inputter.inputPattern("St. code S000: ", "[sS][\\d]{3}");
          newCode= newCode.trim().toUpperCase();
          codeDuplicated = isCodeDupplicated(newCode);//check code duplication
          if (codeDuplicated)System.out.println("Code is duplicated!");
        }
        while (codeDuplicated==true);
        newName = Inputter.inputNonBlankStr("Name of new student: ");
        newName= newName.toUpperCase();
        newMark = Inputter.inputInt("Mark: ", 0, 10); // 0<=mark<=10
        // Create new student
        Student st = new Student(newCode, newName, newMark);
        // Add new student to the list
        this.add(st);
        System.out.println("Student " + newCode + " has been added.");
    }
```

```
42      // Search student based on inputted code
43      public void searchStudent(){
44          if (this.isEmpty())
45              System.out.println("Empty list. No search can be performed!");
46          else{
47              String sCode =Inputter.inputStr("Input student code for search:");
48              Student st= this.search(sCode);// search student based on code
49              if (st==null)
50                  System.out.println("Student " + sCode + " doesn't existed!");
51              else System.out.println("Found: " + st);
52          }
53      }
```

```
54      // Update name and mark based on student's code
55      public void updateStudent(){
56          if (this.isEmpty())
57              System.out.println("Empty list. No update can be performed!");
58          else {
59              String uCode =Inputter.inputStr("Input code of updated student:");
60              Student st = this.search(uCode);// search student
61              if (st==null)
62                  System.out.println("Student " + uCode + " doesn't existed!");
63              else{
64                  // Update student's name
65                  String oldName = st.getName();
66                  String msg = "Old name: " + oldName + ", new name:";
67                  String newName = Inputter.inputNonBlankStr(msg);
68                  st.setName(newName);
69                  // Update student's mark
70                  int oldMark = st.getMark();
71                  msg = "Old mark: " + oldMark + ", new mark 0..10:";
72                  int newMark = Inputter.inputInt(msg, 0, 10);
73                  st.setMark(newMark);
74                  System.out.println("Student " + uCode + " has been updated.");
75              }
76          }
77      }
```

```
78      // Remove a student based on student's code
79      public void removeStudent(){
80          if (this.isEmpty())
81              System.out.println("Empty list. No remove can be performed!");
82          else {
83              String rCode =Inputter.inputStr("Input code of removed student:");
84              Student st = this.search(rCode);// search student
85              if (st==null)
86                  System.out.println("Student " + rCode + " doesn't existed!");
87              else{
88                  this.remove(st); // remove this student
89                  System.out.println("Student " + rCode + " has been removed.");
90              }
91          }
92      }
```

```
93          // List all students
94          public void printAll(){
95              if (this.isEmpty())System.out.println("Empty list!");
96              else{
97                  System.out.println("Student list:");
                    for (Student st: this)System.out.println(st);
99                  System.out.println("Total: " + this.size() + " student(s).");
100             }
101         }
102     }// StudentList
```

```
1   /* Program for managing student list */
2   public class StudentManager {
3       public static void main(String[] args){
4           // options in menu
5           String[] options= { "Add new student","Search a student",
6               "Update name and mark", "Remove a student","List all", "Quit"};
            int choice=0; // user choice
8           StudentList list= new StudentList();// Init empty list
9           do{
10              System.out.println("\nStudent managing Program");
11              choice = Menu.getChoice(options);
12              switch(choice){
13                  case 1: list.addStudent(); break;// Add new student
14                  case 2: list.searchStudent(); break; // Search student
15                  case 3: list.updateStudent(); break; // Update student
16                  case 4: list.removeStudent(); break; // Remove student
17                  case 5: list.printAll(); break; // print all students
18                  default: System.out.println("Bye!");
19              }
20          }
21          while (choice >0 && choice < 6);
22      }
23  }// StudentManager class
```

Test cases

Run the program, the following menu allows user to choose one operation at a time:

Student managing Program
1-Add new student
2-Search a student
3-Update name and mark
4-Remove a student
5-List all
6-Quit
Choose 1..6:

Test case	Option	Input and Result
1	2	No input,

		Output: Message: Empty list. No search can be performed!
2	3	No input, Output: Message: Empty list. No update can be performed!
3	4	No input. Output: Message: Empty list. No remove can be performed!
4	5	No input. Output: Message: Empty list!
5	1	Input code : ABC/ A1234/ S12345/S001. Input name: blank/ James Input mark: -2/12/7 Message: Student S001 has been added.
6	2	Add: S008, Jack, 5
7	3	Add: s004,monica, 8
8	5	Output: Student list: S001, JAMES, 7 S008, JACK, 5 S004, MONICA, 8 Total: 3 student(s).
9	2	Input student code for search: Input S123 Output: Student S123 doesn't exist!
10	2	Input student code for search: Input s008 Output: Found: S008, JACK, 5
11	3	Input code of updated student: Input S123 Output: Student S123 doesn't exist!
12	3	Input code of updated student: Input s008 Old name: JACK, new name: input bill Old mark: 5, new mark 0..10: input 10 Output: Student s008 has been updated.
13	5	Output: Student list: S001, JAMES, 7 S008, BILL, 10 S004, MONICA, 8 Total: 3 student(s).
14	4	Input code of removed student: Input s123 Output: Student s123 doesn't exist!
15	4	Input code of removed student: Input s008 Output: Student s008 has been removed.
16	5	Output: Student list: S001, JAMES, 7 S004, MONICA, 8 Total: 2 student(s).