Assignment Java Core 1

IMPORTANT NOTE: This assignment is marked by oral test
Contact me @ https://www.facebook.com/quynhtran.ly.94

Question 1: (Students are given a full code to this question)

Student pls kindly read the code, understand code and then try to complete this question by themselves. NOT ALLOW: COPY AND PASTE ^^(0 mark, and cant take the final test) Thank you

Create a Java console program to manage students.

Background Context

Write a program to manage information of student. The program implements terminology of Object Oriented Programming (OOP) paradigm. OOP is one of the best choosing ways to design software program.

In this assignment, we will use **ArrayList** to store list of student. In fact, ArrayList is popular used to manipulate with data. ArrayList provides some useful methods, such as: add(), remove(), sort() ., etc.

Program Specifications

A student information consists of ID, Student Name, Semester, Course Name (There are only three courses: Java, .Net, C/C++). The program allows use to create list of student, update/delete student information. On the other hand, use can search student(s) and sort result by student name

1. **Main Screen as below:**

WELCOME TO STUDENT MANAGEMENT

1. **Create**
2. **Find and Sort**
3. **Update/Delete**
4. **Report**
5. **Exit**

(Please choose 1 to Create, 2 to Find and Sort, 3 to Update/Delete, 4 to Report and 5 to Exit program).

UML diagram


```
6 package ws07question3;
 7
 8 /**
 9  *
10  * @author Quynh Tran Ly
11  */
12 public class Manager {
13     //Show menu
14     public static void menu() {
15         System.out.println(" 1. Create");
16         System.out.println(" 2. Find and Sort");
17         System.out.println(" 3. Update/Delete");
18         System.out.println(" 4. Report");
19         System.out.println(" 5. Exit");
20         System.out.print(" Enter your choice: ");
21     }
22 }
23
```

```
6 package ws07question3;
 7
 8 import java.util.ArrayList;
```

```java
import java.util.Scanner;

/**
 *
 * @author Quynh Tran Ly
 */
public class Validation {

    private final static Scanner in = new Scanner(System.in);

    //check user input number limit
    public static int checkInputIntLimit(int min, int max) {
        //loop until user input correct
        while (true) {
            try {
                int result = Integer.parseInt(in.nextLine().trim());
                if (result < min || result > max) {
                    throw new NumberFormatException();

                }
                return result;
            } catch (NumberFormatException e) {
                System.err.println("Please input number in rage [" + min + ", " + max + "]");
                System.out.print("Enter again: ");
            }
        }
    }

    //check user input string
    public static String checkInputString() {
        //loop until user input correct
```

```
40      while (true) {
41          String result = in.nextLine().trim();
42          if (result.isEmpty()) {
43              System.err.println("Not empty");
44              System.out.print("Enter again: ");
45          } else {
46              return result;
47          }
48      }
49  }
50
51  //check user input yes/ no
52  public static boolean checkInputYN() {
53      //loop until user input correct
54      while (true) {
55          String result = checkInputString();
56          //return true if user input y/Y
57          if (result.equalsIgnoreCase("Y")) {
58              return true;
59          }
60          //return false if user input n/N
61          if (result.equalsIgnoreCase("N")) {
62              return false;
63          }
64          System.err.println("Please input y/Y or n/N.");
65          System.out.print("Enter again: ");
66      }
67  }
68
69  //check user input u / d
70  public static char checkInputUD() {
```

```
71      //loop until user input correct
72      while (true) {
73          String result = checkInputString();
74          //return true if user input u/U
75          if (result.equalsIgnoreCase("U")) {
76              return 'U';
77          }
78          //return false if user input d/D
79          if (result.equalsIgnoreCase("D")) {
80              return 'D';
81          }
82          System.err.println("Please input u/U or d/D.");
83          System.out.print("Enter again: ");
84      }
85  }
86
87  //check user input course
88  public static String checkInputCourse() {
89      //loop until user input correct
90      while (true) {
91          String result = checkInputString();
92          //check input course in java/ .net/ c/c++
93          if (result.equalsIgnoreCase("java")
94                  || result.equalsIgnoreCase(".net")
95                  || result.equalsIgnoreCase("c/c++")) {
96              return result;
97          }
98          System.err.println("There are only three courses: Java, .Net, C/C++");
99          System.out.print("Enter again: ");
100     }
101 }
```

```
102
103     //check id and exist
104     public static int checkIdExist(ArrayList<Student> ls, String id) {
105         for (int i = 0; i < ls.size(); i++) {
106             if (ls.get(i).getId().equalsIgnoreCase(id)) {
107                 return i;
108             }
109         }
110         return -1;
111     }
112
113 }
114
1 package ws07question3;
2
3
4 public class Student implements Comparable<Student> {
5
6     private String id;
7     private String studentName;
8     private String semester;
9     private String courseName;
10
11     public Student() {
12     }
13
14     public Student(String id, String studentName, String semester, String courseName) {
15         this.id = id;
16         this.studentName = studentName;
17         this.semester = semester;
18         this.courseName = courseName;
```

```
19  }
20
21  public String getId() {
22      return id;
23  }
24
25  public void setId(String id) {
26      this.id = id;
27  }
28
29  public String getStudentName() {
30      return studentName;
31  }
32
33  public void setStudentName(String studentName) {
34      this.studentName = studentName;
35  }
36
37  public String getSemester() {
38      return semester;
39  }
40
41  public void setSemester(String semester) {
42      this.semester = semester;
43  }
44
45  public String getCourseName() {
46      return courseName;
47  }
48
49  public void setCourseName(String courseName) {
```

```
50      this.courseName = courseName;
51  }
52
53  @Override
54  public int compareTo(Student t) {
55      return t.studentName.compareTo(this.studentName);
56  }
57
58  public void print() {
59      System.out.printf("%-15s%-15s%-15s\n", studentName, semester, courseName);
60  }
61
62  @Override
63  public String toString() {
64      return "Student{" + "id=" + id + ", studentName=" + studentName + ", semester=" + semester + ",
courseName=" + courseName + '}';
65  }
66
67 }
68
```

```
6 package ws07question3;
 7
 8 import java.util.ArrayList;
 9 import java.util.Collections;
10
11 /**
12  *
13  * @author Quynh Tran Ly
14  */
15 public class StudentList {
```

```
16
17  private ArrayList<Student> ls = new ArrayList<Student>();
18
19  public StudentList() {
20      ls.add(new Student("1", "Tran Ly", "3", "Java"));
21       ls.add(new Student("2", "Huynh Van", "2", "C/C++"));
22  }
23
24  //Allow user create new student
25  public void createStudent() {
26      //loop until user input not duplicate
27      while (true) {
28          System.out.print("Enter id: ");
29          String id = Validation.checkInputString();
30          System.out.print("Enter name student: ");
31          String name = Validation.checkInputString();
32          //Check id exit
33          if (Validation.checkIdExist(ls, id) != -1) {
34              System.err.println("Id has exist student. Pleas re-input.");
35              if (!Validation.checkInputYN()) {
36                  return;
37              }
38          }
39          System.out.print("Enter semester: ");
40          String semester = Validation.checkInputString();
41          System.out.print("Enter name course: ");
42          String course = Validation.checkInputCourse();
43          //check student exist or not
44          ls.add(new Student(id, name, semester, course));
45          System.out.println("Add student success.");
46          System.out.print("Do you want to continue : y/n: ");
```

```
47          if (!Validation.checkInputYN()) {
48              return;
49          }
50      }
51  }
52
53  ////Allow user create find and sort
54  public void findAndSort() {
55      //check list empty
56      if (ls.isEmpty()) {
57          System.err.println("List empty.");
58          return;
59      }
60      ArrayList<Student> listStudentFindByName = new ArrayList<>();
61      System.out.print("Enter name to search: ");
62      String name = Validation.checkInputString();
63      //this code is used to find the student has name that contains ""
64      for (Student student : ls) {
65          //check student have name contains input
66          if (student.getStudentName().contains(name)) {//
67              listStudentFindByName.add(student);
68          }
69      }
70      if (listStudentFindByName.isEmpty()) {
71          System.err.println("Not exist.");
72      } else {
73          Collections.sort(listStudentFindByName);
74          System.out.printf("%-15s%-15s%-15s\n", "Student name", "semester", "Course Name");
75          //loop from first to last element of list student
76          for (Student student : listStudentFindByName) {
77              student.print();
```

```
78          }
79      }
80  }
81  //Update and Delete
82  public void updateOrDelete() {
83      if (ls.isEmpty()) {
84          System.err.println("List empty.");
85          return;
86      }
87      System.out.print("Enter id to update/delete: ");
88      String id = Validation.checkInputString();
89      int index = Validation.checkIdExist(ls, id);
90      if (index == -1) {
91          System.err.println("Not found student.");
92          return;
93      } else {
94
95          System.out.print("Do you want to update (U) or delete (D) student: ");
96          //check user want to update or delete
97          if (Validation.checkInputUD() == 'U') {
98              System.out.print("Enter name student: ");
99              String name = Validation.checkInputString();
100             System.out.print("Enter semester: ");
101             String semester = Validation.checkInputString();
102             System.out.print("Enter name course: ");
103             String course = Validation.checkInputCourse();
104             ls.get(index).setStudentName(name);
105             ls.get(index).setSemester(semester);
106             ls.get(index).setCourseName(course);
107             System.err.println("Update success.");
108             return;
```

```
109         } else {
110             ls.remove(index);
111             System.err.println("Delete success.");
112             return;
113         }
114     }
115 }
116 //Report
117 public void report()
118 {   Collections.sort(ls);
119     for (Student l : ls) {
120         System.out.println(l);
121     }
122 }
123 public ArrayList<Student> getLs() {
124     return ls;
125 }
126 
127 public void setLs(ArrayList<Student> ls) {
128     this.ls = ls;
129 }
130 
131 }
132 
```

```
6 package ws07question3;
7 
8 /**
9  *
10  * @author Quynh Tran Ly
11  */
```

```java
public class WS07Question3 {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        StudentList sl= new StudentList();
         while (true) {
            //Show menu option
            Manager.menu();
            int choice = Validation.checkInputIntLimit(1, 5);
            switch (choice) {
                case 1:
                    sl.createStudent();
                    break;
                case 2:
                    sl.findAndSort();
                    break;
                case 3:
                    sl.updateOrDelete();
                    break;
                case 4:
                    sl.report();
                    break;
                case 5:
                    return;
            }

        }
    }
```

```
43

44 }

45
```

Question 2: Students pls do this question based on the new structure(UML Diagram) from Question 1.

Prescription Course Management System

This case study is relatively straightforward, and hence can be tackled by most beginning modelers fairly effortlessly.

Background

A School wishes for us to design and develop an automated Course Management System (CM). The requirements are as follows:

The system is to keep track of the following information for each course:

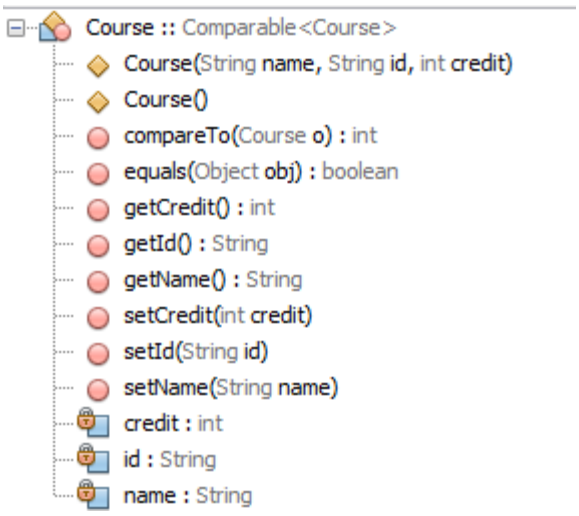
- Course id as a string
- Course name as a string
- Number of credit as a whole number

The system is required to support the following queries:

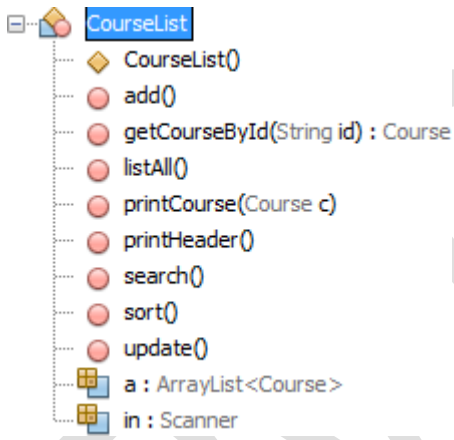
1. A list of all available courses in the system
2. Search and display information of a course by course id
3. Record/Add information of course
4. Sort all courses by number of credit as ascending
5. Update information of a specific course (by course id)

You should des:

1. Course class: This will hold the information of course and add some needed operation.



2. CourseList class: This will hold and manage the list of courses, the list of courses can be used as collection.



3. And finally, make your own main function and test all the above operations

The program output might look something like:

```
COURSE MANAGEMENT SYSTEM
1. A list of all available courses in the system
2. Search and display information of  a course by course id
3. Record/Add information of course
4. Sort all courses by number of credit as ascending
5. Update information of a specific course (by course id)
0. Exit
Select your choice: 3
Enter course id: PRO192
Enter course name: OOP with Java
Enter course credit: 3
Information of course has been added

COURSE MANAGEMENT SYSTEM
1. A list of all available courses in the system
2. Search and display information of  a course by course id
3. Record/Add information of course
4. Sort all courses by number of credit as ascending
5. Update information of a specific course (by course id)
0. Exit
Select your choice: 1
Course id Course name                   Course credit
PRO192    OOP with Java                 3


COURSE MANAGEMENT SYSTEM
1. A list of all available courses in the system
2. Search and display information of  a course by course id
3. Record/Add information of course
4. Sort all courses by number of credit as ascending
5. Update information of a specific course (by course id)
0. Exit
Select your choice: 5
Enter course id: PRO192
Information of course
PRO192    OOP with Java                 3
Enter new course credit: 4
Information of course has been updated

COURSE MANAGEMENT SYSTEM
1. A list of all available courses in the system
2. Search and display information of  a course by course id
3. Record/Add information of course
4. Sort all courses by number of credit as ascending
5. Update information of a specific course (by course id)
0. Exit
Select your choice: 1
Course id Course name                   Course credit
PRO192    OOP with Java                 4
```

```
COURSE MANAGEMENT SYSTEM
1. A list of all available courses in the system
2. Search and display information of  a course by course id
3. Record/Add information of course
4. Sort all courses by number of credit as ascending
5. Update information of a specific course (by course id)
0. Exit
Select your choice: 2
Enter course id: pro192
Information of course
PRO192    OOP with Java                 3
```

Question 3:

Students kindly draw UML Diagram and take the short screen of output and then submit to the lecturer. Thank you :-D

Background:

An internet shopping site sells software and books and any products. Customers fill in a virtual "shopping basket" with their selection of items from a catalog of both categories. They can add or remove individual shopping items to/from the shopping basket until they are happy with their selection; they then submit their basket to the site for total price plus postage-and-packing (P&P) calculation.

Each item has an individual price that varies from item to item and is shown in the on-line catalogue. Because software titles are considered weightless (cdroms are very light), postage and packing is a single price ($3.25) for each software item up to 3 items, $1.50 per software item thereafter.
Postage and packing for books is calculated from the sum of the weights of the books as follows:

- $5.00 under 500 grams (total weight),
- $9.50 for book weight totals of 500grams or above but under one kilogram,
- $7.00 are added for every additional kilogram thereafter

e.g. P&P = $9.50 + $7.00 for weights 1kg or above but less than 2kg, $9.50 + $14.00 for 2kg or above, but less than 3, etc...
The weight of each book is shown in the on-line catalogue.

Total postage and packing cost for a single order is calculated as the sum of the P&P cost of books and the P&P cost of software.

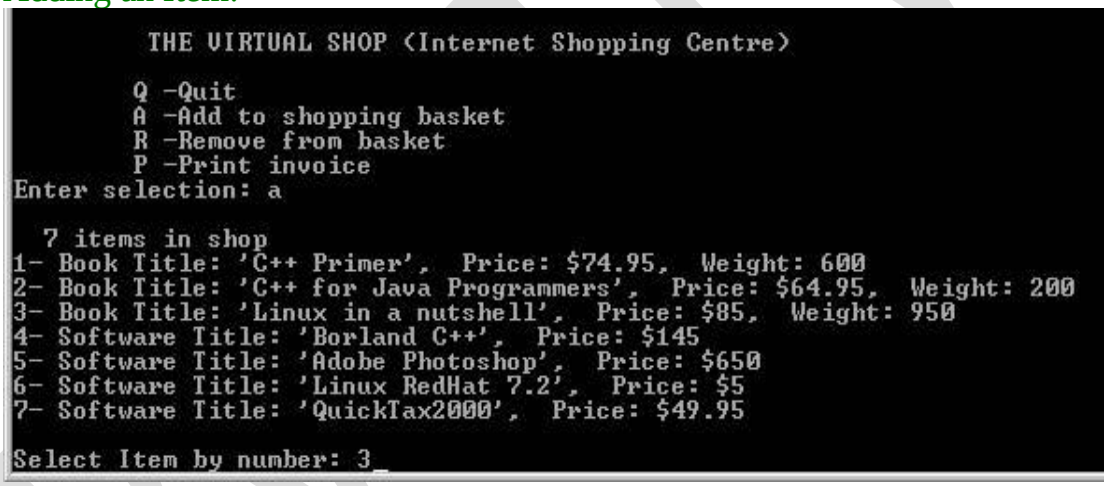
Your Tasks:

1. Draw a basic class diagram for a OOP design that will enable the software to take orders, then calculate the total price of a virtual shopping basket, including postage and packing.
   The classes involved are Menu, Basket, Shop, ShopApp, ShopItem, Book, and Software.
   ShopItem is a base class for all kinds of sale items.
   Your design will use dynamic binding, which means that a Basket object will contain a

collection of ShopItems rather than a collection of books and another collection of software titles

2. Implement the classes : ShopItem data should be read from a text file.
3. Screens should display the catalogue and a menu. Items may be selected as numbers in a list like the following

Adding an Item:



Removing an Item:

Printing an Invoice

Technical Requirements

- The project should be applied as much as possible of OOP characteristics : encapsulation, inheritance and polymorphism and other topics of Java programming: exception handling, I/O

- Classes should be carefully designed to protect object data integrity by thoughtful use of public, private and protected sections as necessary.

- ShopItem and derived classes should not be directly responsible for console input . That role must be left to programs that use the class.

- Named constants should be used instead (constants may be global) of “magic number”.

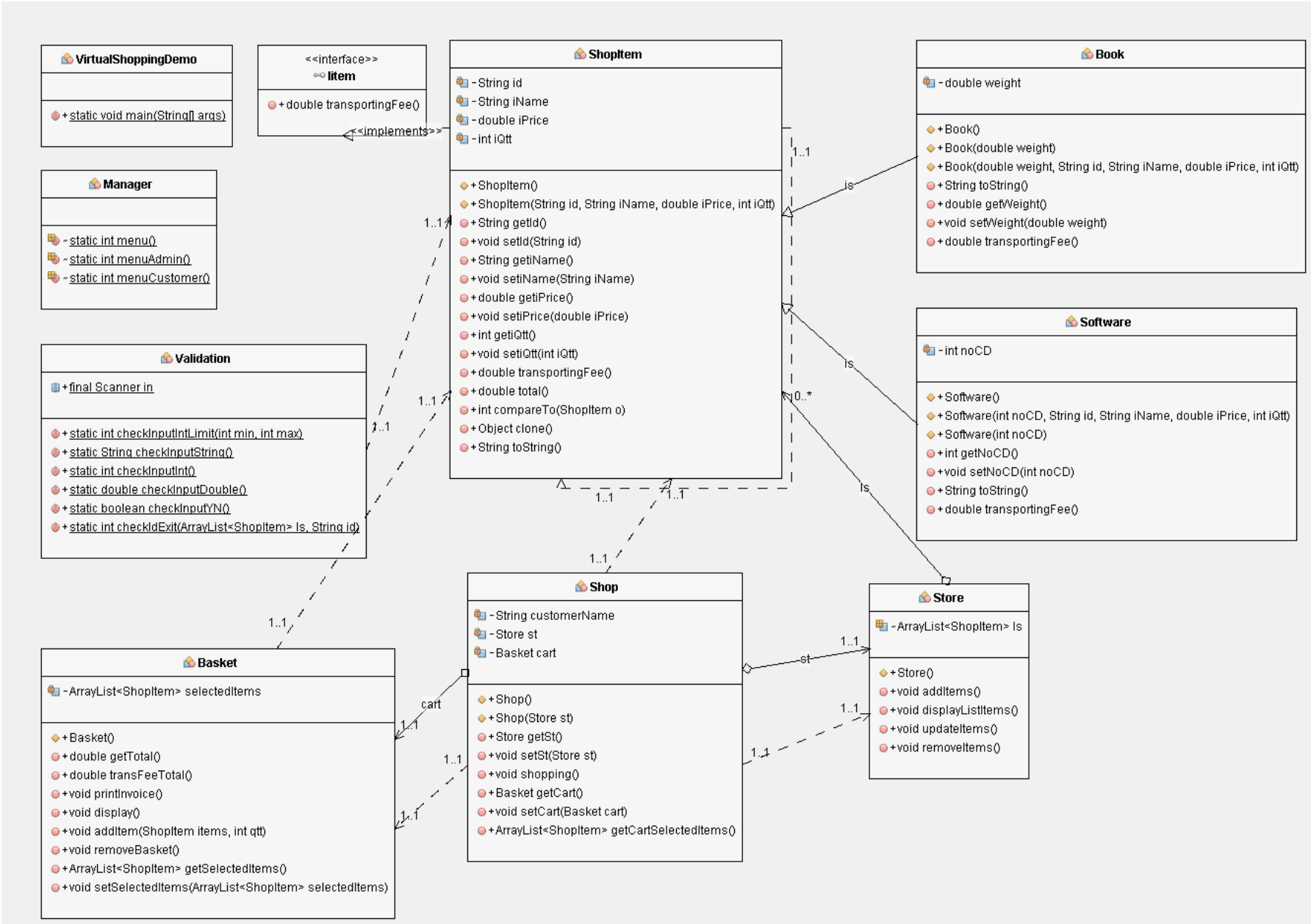
- Functions should be kept small.

- Consistent style standards will be kept:

  - Indentation should be used..
    * Identifiers will be meaningful (not too abbreviated).
    * Every file will have a banner comment.

Happy Virtual Shopping

Happy doing Assigment :-D


VirtualShoppingDemo
+static void main(String[] args)
<<interface>>
Item
+double transportingFee()
<<implements>>
ShopItem
-String id
-String iName
-double iPrice
-int iQtt
+ShopItem()
+ShopItem(String id, String iName, double iPrice, int iQtt)
+String getId()
+void setId(String id)
+String getiName()
+void setiName(String iName)
+double getiPrice()
+void setiPrice(double iPrice)
+int getiQtt()
+void setiQtt(int iQtt)
+double transportingFee()
+double total()
+int compareTo(ShopItem o)
+Object clone()
+String toString()
Book
-double weight
+Book()
+Book(double weight)
+Book(double weight, String id, String iName, double iPrice, int iQtt)
+String toString()
+double getWeight()
+void setWeight(double weight)
+double transportingFee()
Manager
~static int menu()
~static int menuAdmin()
~static int menuCustomer()
Validation
+final Scanner in
+static int checkInputIntLimit(int min, int max)
+static String checkInputString()
+static int checkInputInt()
+static double checkInputDouble()
+static boolean checkInputYN()
+static int checkIdExit(ArrayList<ShopItem> ls, String id)
Software
-int noCD
+Software()
+Software(int noCD, String id, String iName, double iPrice, int iQtt)
+Software(int noCD)
+int getNoCD()
+void setNoCD(int noCD)
+String toString()
+double transportingFee()
Shop
-String customerName
-Store st
-Basket cart
+Shop()
+Shop(Store st)
+Store getSt()
+void setSt(Store st)
+void shopping()
+Basket getCart()
+void setCart(Basket cart)
+ArrayList<ShopItem> getCartSelectedItems()
Store
~ArrayList<ShopItem> ls
+Store()
+void addItems()
+void displayListItems()
+void updateItems()
+void removeItems()
Basket
-ArrayList<ShopItem> selectedItems
+Basket()
+double getTotal()
+double transFeeTotal()
+void printInvoice()
+void display()
+void addItem(ShopItem items, int qtt)
+void removeBasket()
+ArrayList<ShopItem> getSelectedItems()
+void setSelectedItems(ArrayList<ShopItem> selectedItems)
is
ls
st
cart
1..1
0..*

```java
package virtualshoppingdemo;

import java.util.ArrayList;

/**
 *
 * @author Quynh Tran Ly
 */
public class VirtualShoppingDemo {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
        Store st = new Store();
        Shop sh = new Shop();
        sh=new Shop(st);
        st.ls.add(new Book(1, "111", "Conan", 10.3, 10));
        st.ls.add(new Book(2, "222", "Doreamon", 1.2, 10));
        st.ls.add(new Software(7, "333", "My Tam", 7.8, 10));
        st.ls.add(new Book(3, "444", "Tom", 5.5, 10));
        st.ls.add(new Software(5, "555", "Snow", 6, 10));

//loop until user want to exit
        outer:
        while (true) {

            int choice = Manager.menu();
            //Menu for admin
            if (choice == 1) {
                while (true) {
                    int choiceA = Manager.menuAdmin();
                    switch (choiceA) {
                        case 1:
                            System.out.println("- Add Items");
                            st.addItems();
                            break;
                        case 2:
                            System.out.println("- Update Items");
                            st.updateItems();
                            break;
                        case 3:
                            System.out.println("- View Items");
                            st.displayListItems();
                            break;
                        case 4:
                            System.out.println("- Remove Items");
                            st.removeItems();
                            break;
                        case 5:
                            continue outer;
                    }
                }
            }
            //Menu for Customer
            if (choice == 2) {
                while (true) {
                    int choiceB = Manager.menuCustomer();
```

```
65                switch (choiceB) {
66                  case 1:
67                    System.out.println("1. Add to Shopping Basket");
68
69                    sh.shopping();
70                    break;
71                  case 2:
72                    System.out.println("Display the Shopping Basket");
73                    sh.getCart().display();
74                    break;
75                  case 3:
76                    System.out.println("Remove Shopping Basket");
77                    sh.getCart().removeBasket();
78                  case 4:
79                    System.out.println("Print Invoice(Orders)");
80                    sh.getCart().printInvoice();
81                    break;
82                  case 5:
83                    System.out.println("5. Exit");
84                    continue outer;
85              }
86            }
87
88          }
89          // Exit
90          if (choice == 3) {
91            return;
92          }
93
94      }
95    }
96 }
97
```

```
6 package virtualshoppingdemo;
 7
 8 import java.util.ArrayList;
 9 import java.util.Scanner;
10
11 /**
12  *
13  * @author Quynh Tran Ly
14  */
15 public class Validation {
16    public  static final Scanner in = new Scanner(System.in);
17
18    //check user input number limit
19    public static int checkInputIntLimit(int min, int max) {
20        //Students write here appropriate statements to complete this function
21      }
22    }
23
24    //check user input string
25    public static String checkInputString() {
26      //Students write here appropriate statements to complete this function
27      }
28    }
29
```

```
30    //check user input int
31    public static int checkInputInt() {
32        //Students write here appropriate statements to complete this function
33    }
34
35    //check user input double
36    public static double checkInputDouble() {
37        //Students write here appropriate statements to complete this function
38    }
39
40    //check user input yes/ no
41    public static boolean checkInputYN() {
42        //Students write here appropriate statements to complete this function
43       }
44    }
45
46    public static int checkIdExit(ArrayList<ShopItem> ls, String id)
47    { //Students write here appropriate statements to complete this function
48
49    }
50 }
```

```
6  package virtualshoppingdemo;
7
8  import java.util.ArrayList;
9
10 /**
11  *
12  * @author Quynh Tran Ly
13  */
14 public class Manager {
15
16    //display menu
17    static int menu() {
18       //loop until user want to exit
19       System.out.println("1. Admin Roles");
20       System.out.println("2. Customer Roles");
21       System.out.println("3. Exits");
22       System.out.print("Enter your choice: ");
23       int choice = Validation.checkInputIntLimit(1, 3);
24       return choice;
25    }
26
27    //display menu
28    static int menuAdmin() {
29       //loop until user want to exit
30       System.out.println("1. Add Items");
31       System.out.println("2. Update Items");
32       System.out.println("3. View Items");
33       System.out.println("4. Remove Items");
34       System.out.println("5. Exit");
35       System.out.print("Enter your choice: ");
36       int choice = Validation.checkInputIntLimit(1, 5);
37       return choice;
38    }
39
40    //display menu
41    static int menuCustomer() {
```

```
42      //loop until user want to exit
43      System.out.println("1. Add to Shopping Basket");
44      System.out.println("2. Display the Shopping Basket");
45      System.out.println("3. Remove from Shopping Basket");
46      System.out.println("4. Print Invoice(Orders)");
47      System.out.println("5. Exit");
48      System.out.print("Enter your choice: ");
49      int choice = Validation.checkInputIntLimit(1, 5);
50      return choice;
51    }
52
53
54
55
56 }
57
```

```
6 package virtualshoppingdemo;
7
8 /**
9  *
10  * @author Quynh Tran Ly
11  */
12 public interface Iitem {
13
14
15 }
16
```

```
6 package virtualshoppingdemo;
7
8 /**
9  *
10  * @author Quynh Tran Ly
11  */
12 public class ShopItem implements Cloneable, Iitem, Comparable<ShopItem>{
13    private String id;
14    private String iName;
15    private double iPrice;
16    private int iQtt;
17
18    //Students write here appropriate statements to complete this function
19
20    @Override
21    public int compareTo(ShopItem o) {
22      return this.id.compareTo(o.id);
23    }
24
25    @Override
26    public Object clone() throws CloneNotSupportedException {
27      return super.clone(); //To change body of generated methods, choose Tools | Templates.
28    }
29
30
31    @Override
32    public String toString() {
33      return "ShopItem{" + "id=" + id + ", iName=" + iName + ", iPrice=" + iPrice + ", iQtt=" + iQtt + '}';
```

```
34    }
35
36
37
38 }
39
```

```
5  */
6  package virtualshoppingdemo;
7
8  /**
9   *
10  * @author Quynh Tran Ly
11  */
12 public class Book extends ShopItem{
13    private double weight;
14
15    //Students write here appropriate statements to complete this function
16
17 }
18
```

```
6  package virtualshoppingdemo;
7
8  /**
9   *
10  * @author Quynh Tran Ly
11  */
12 public class Software extends ShopItem{
13    private int noCD;
14
15    //Students write here appropriate statements to complete this function
16 }
17
```

```
6  package virtualshoppingdemo;
7
8  import java.util.ArrayList;
9  import java.util.Collection;
10 import java.util.Collections;
11 import java.util.List;
12
13 /**
14  *
15  * @author Quynh Tran Ly
16  */
17 public class Store{
18
19    ArrayList<ShopItem> ls = new ArrayList<ShopItem>();
20
21    public Store() {
22    }
23
24    public void addItems() {
25       String id, name;
26       double price;
```

```
27      int quantity;
28      while (true) {
29          System.out.println("1. Add Book");
30          System.out.println("2. Add Software");
31          System.out.println("3. Exit");
32          int choice = Validation.checkInputIntLimit(1, 3);
33          switch (choice) {
34              //Students write here appropriate statements to complete this function
35
36              case 3:
37                  return;
38          }
39      }
40  }
41
42  public void displayListItems() {
43      Collections.sort(ls);
44      //Students write here appropriate statements to complete this function
45
46
47  }
48
49  public void updateItems() {
50      System.out.print("Enter Id: ");
51      String id = Validation.checkInputString();
52      for (int i = 0; i < ls.size(); i++) {
53          if (ls.get(i).getId().equalsIgnoreCase(id)) {
54              System.out.println("Found the items. want to update? Y/N");
55
56              //Students write here appropriate statements to complete this function
57
58              } else {
59                  System.out.println("Cant find the id");
60                  System.out.println("Enter again:");
61              }
62
63          }
64      }
65
66  }
67
68  public void removeItems() {
69      if (ls.isEmpty()) {
70          System.out.println("List is empty");
71          return;
72      }
73
74      //Students write here appropriate statements to complete this function
75
76  }
77
78 }
79
```

```
6 package virtualshoppingdemo;
7
8 import java.util.ArrayList;
9
```

```java
/**
 *
 * @author Quynh Tran Ly
 */
public class Shop {

    private String customerName;
    private Store st = new Store();
    private Basket cart = new Basket();

    public Shop() {
    }

    public Shop(Store st) {
        this.st = st;
    }

    public Store getSt() {
        return st;
    }

    public void setSt(Store st) {
        this.st = st;
    }

    public void shopping() {
        if (st.ls.isEmpty()) {
            System.err.println("Do not have any items");
            return;
        }
        System.out.println("Start shopping. Enter name:");
        String name = Validation.checkInputString();
        //Students write here appropriate statements to complete this function




    }


    public Basket getCart() {
        return cart;
    }

    public void setCart(Basket cart) {
        this.cart = cart;
    }

    public ArrayList<ShopItem> getCartSelectedItems() {
        return cart.getSelectedItems();
    }

}
```

```java
package virtualshoppingdemo;

import java.util.ArrayList;
```

```
 9 import java.util.List;
10
11 /**
12  *
13  * @author Quynh Tran Ly
14  */
15 public class Basket {
16
17    private ArrayList<ShopItem> selectedItems = new ArrayList<ShopItem>();
18
19
20    public Basket() {
21    }
22
23
24
25     public double getTotal(){
26        double tot=0;
27        //Students write here appropriate statements to complete this function
28
29        return tot;
30     }
31
32    public double transFeeTotal(){
33        double tft=0;
34       //Students write here appropriate statements to complete this function
35
36        return tft;
37    }
38
39    public void printInvoice(){
40       //Students write here appropriate statements to complete this function
41
42    }
43    public void display()
44    {
45
46        for (ShopItem selectedItem : selectedItems) {
47            System.out.println(selectedItem);
48        }
49    }
50
51    public void addItem(ShopItem items, int qtt){
52          //Students write here appropriate statements to complete this function
53
54
55    }
56    public void removeBasket() {
57        if (selectedItems.isEmpty()) {
58            System.out.println("Basket is empty");
59            return;
60        }
61
62        System.out.println("Enter the id to remove");
63         //Students write here appropriate statements to complete this function
64
65
66            }
67
```

```
68
69    public ArrayList<ShopItem> getSelectedItems() {
70        return selectedItems;
71    }
72
73    public void setSelectedItems(ArrayList<ShopItem> selectedItems) {
74        this.selectedItems = selectedItems;
75    }
76
77
78 }
79
```