CQ14.1 - You are supplied a project for managing clocks (file Exam1.rar).
This project includes:

(1) I_List.java: Interface for managing a list
(2) I_Menu.java: Interface declared methods for a menu in console program.
(3) Inputter.java: Class that supports data input from users
(4) Item.java: Class represents a item
(5) Menu.java: the Menu class
(6) Clock.java: Class represents a clock
(7) ClockList.java: Class for a list of clocks
(8) ClockListUse.java: Main program.

Describe:

Each **clock** includes the following information: **code** (String), **make** (String), **size** (int), **price** (int). The file **data-in.txt** that stores the clock list initially has the following format:

```
C001,Citizen,40,295
C002,Longines,40,3300
C003,Seiko,39,297
C004,Rolex,41,20000
C005,Omega,42,5000
```

This list is uploaded, processed and then saved to **data-out.txt** in the following format:

```
C001- CITIZEN- 40- 295
C002- LONGINES- 40- 3300
C004- ROLEX- 41- 20000
C005- OMEGA- 42- 5000
```

You works:

Implement the class **Clock** extending from **Item** (i.e. Item is a superclass and Clock is a subclass) with the following information:

Clock
-size:int -price:int
+Clock() +Clock(code:String, make:String, size:int, price:int) +getSize():int +getPrice():int +setSize(size:int):void +setPrice(price:int):void +input():void +toString():String

Where:

- getSize():int - return size.
- getPrice():int - return price.
- setSize(size:int):void - update size.
- setPrice(price:int):void - update price.
- input():void - input a **clock** from user
- toString():String - return the string of format: **code**, **make**, **size**, **price**

Write the class **ClockList** that extends **Clock** and implements the interface **I_List**. This class uses **ArrayList** to manage the list of clocks and implements methods in **I_List** as below:

- void addFromFile(String fName); load the list from the file data-in.txt.
- void saveToFile(String fName); save list to the file data-out.txt.
- void addClock(); add a new clock, with input.
- void findClockByCode(String code); find and print out all clocks from the list by code. If code does not exist, print out "Not found".
- void findClockByMake(String make); find and print out all clocks from the list by make. If make does not exist, print out "Not found".
- Clock findClockWithMaxPrice(); return the clock with the highest price (suppose in the list only one clock with the highest price).
- Clock findClockWithMinPrice(); return the clock with the lowest price (suppose in the list only one clock with the lowest price).
- void countClock(); count and print out the total number of clocks from the list (called from SortAndPrint() and run in SortAndPrint()).
- void sortAndPrint(); sort clocks by **ascending price** and print out the list of clocks with the format: **code**, **make**, **size**, **price**.

CQ14.2 - You are supplied a project for managing cars (file Exam2.rar).
This project includes:

(1) I_List.java: Interface for managing a list
(2) I_Menu.java: Interface declared methods for a menu in console program.
(3) Inputter.java: Class that supports data input from users
(4) Item.java: Class represents a item
(5) Menu.java: the Menu class
(6) Car.java: Class represents a car
(7) CarList.java: Class for a list of cars
(8) CarListUse.java: Main program.

Describe:

Each **car** includes the following information: **code** (String), **make** (String), **owner** (String), **price** (double), **color** (String). The list of cars is stored in the file **data.dat** (a binary file). The data saved and read back from the file **data.dat** has the following format:

```
CA001, BMW, AN DONG, 63.25, BLACK
CA002, MERCEDES, AN YEN, 112.56, RED
CA003, HONDA CIVIC 2022, AN LAC, 36.99, GRAY
```

You works:

Implement the class **Car** extending from **Item** (i.e. Item is a superclass and Car is a subclass) with the following information:

Car
-owner:String -price:double -color:String
+Car() +Car(code:String, make:String, owner:String, price:double, color: String) +getOwner():String +getPrice():double +getColor():String +setOwner(owner:String):void +setPrice(price:double):void +setColor(color:String):void +input():void +toString():String

Where:

- getOwner():String - return owner.
- getPrice():double - return price.
- getColor():String - return color
- setOwner(owner:String):void - update owner.
- setPrice(price:double):void - update price.
- setColor(color: String): void - update color.
- input():void - input a **car** from user.
- toString():String - return the string of format: **code**, **make**, **owner**, **price**, **color**.

Write the class **CarList** that extends **Car** and implements the interface **I_List**. This class uses **ArrayList** to manage the list (cars) and implements methods in **I_List** as below (you can add other functions in the class CarList):

- void addFromFile(String fName): load the list from the file data.dat.
- void saveToFile(String fName): save list to the file.
- int find(String code): find the code of a car in the list.
- void addCar(): add a new car, with input.
- void findCarByMake(String make); find and print out all cars from the list by make. If make does not exist, print out "Not found".
- void findCarByPartOfOwner(String owner); find and print out all cars from the list by part of owner. If the information does not exist, print out "Not found".
- void remove(): delete a car by code, with input.
- void update(): update owner of a car by code, with input. Notes: Information is not allowed to blank.
- void sortAndPrint(); sort cars by **ascending code** and print out the list of cars with the format: **code**, **make**, **owner**, **price**, **color**.

CQ14.3 - You are supplied a project for managing tShirts (file Exam3.rar).
This project includes:

(1) I_List.java: Interface for managing a list
(2) I_Menu.java: Interface declared methods for a menu in console program.
(3) Inputter.java: Class that supports data input from users
(4) Item.java: Class represents a item
(5) Menu.java: the Menu class
(6) TShirt.java: Class represents a tShirt
(7) TShirtList.java: Class for a list of tShirts
(8) TShirtListUse.java: Main program.

Describe:

Each **tShirt** includes the following information: **code** (String), **make** (String), **style** (String), **size** (int), **color** (int). The file **data-in.txt** that stores the tShirt list initially has the following format:

```
TS001;A;m-011;3;425
TS002;D;f-023;2;286
TS003;B;m-024;5;109
TS004;D;f-052;4;500
TS005;E;m-036;6;802
```

This list is uploaded, processed and then saved to **data-out.txt** in the following format:

```
TS001,A,M-011,3,425
TS002,D,F-023,2,286
TS004,D,F-052,4,500
TS005,E,M-036,6,802
```

You works:

Implement the class **TShirt** extending from **Item** (i.e. Item is a superclass and TShirt is a subclass) with the following information:

TShirt
-style:String -size:int -color:int
+TShirt() +TShirt(code:String, make:String, style:String, size:int, color:int) +getStyle():String +getSize():int +getColor():int +setStyle(style:String):void +setSize(size:int):void +setColor(color:int):void +input():void +toString():String

Where:

- getStyle():String - return style
- getSize():int - return size
- getColor():int - return color
- setStyle(style:String):void - update style
- setSize(size:int):void - update size
- setColor(color:int):void - update color
- input():void - input a **tShirt** from user
- toString():String - return the string of format: **code**, **make**, **style**, **size**, **color**

Write the class **TShirtList** that extends **TShirt** and implements the interface **I_List**. This class uses **ArrayList** to manage the list of tShirts and implements methods in **I_List** as below:

- void addFromFile(String fName); load the list from the file data-in.txt.
- void saveToFile(String fName); save list to the file data-out.txt.
- int find(String code); find the code of a tShirt in the list.
- void addTShirt(); add a new tShirt, with input.
- void findTShirtByStyle(String style); find and print out all tShirts from the list by style. If style does not exist, print out "Not found".
- void findTShirtByPartOfStyle(String style); find and print out all tShirts from the list by part of style. If the information does not exist, print out "Not found".
- void remove(); delete a tShirt by code, with input.
- void update(); update size and/or color of a tShirt by code, with input (i.e. if information is blank, the old information will not be changed).
- void sortAndPrint(); sort tShirts by ascending make and ascending size, and print out the list of tShirts with the format: **code**, **make**, **style**, **size**, **color**.

CQ14.4 - You are supplied a project for managing softDrinks (file Exam4.rar).
This project includes:

(1) I_List.java: Interface for managing a list
(2) I_Menu.java: Interface declared methods for a menu in console program.
(3) Inputter.java: Class that supports data input from users
(4) Item.java: Class represents a item
(5) Menu.java: the Menu class
(6) SoftDrink.java: Class represents a softDrink
(7) SoftDrinkList.java: Class for a list of softDrinks
(8) SoftDrinkListUse.java: Main program.

Describe:

Each softDrink includes the following information: **code** (String), **make** (String), **volume** (int), **price** (double). The generated softDrinks list is stored in the **data.dat** file (binary file) and read back as follows:

```
List created:
2C017, Coca-Cola, 235, 182.0
MD020, Mirinda, 330, 165.0
MD033, Mirinda, 330, 125.0
PS021, Pepsi, 320, 199.0
SP005, Schweppes, 320, 156.0
Total SoftDrink: 5
```

You works:

Implement the class **SoftDrink** extending from **Item** (i.e. Item is a superclass and SoftDrink is a subclass) with the following information:

SoftDrink
-volume:int -price:double
+SoftDrink() +SoftDrink(code:String, make:String, volume:int, price:double) +getVolume():int +getPrice():double +setVolume(volume:int):void +setPrice(price:float):void +input():void +toString():String

Where:

- getVolume():int - return volume.
- getPrice():double - return price.
- setVolume(volume:int):void - update volume.
- setPrice(price:float):void - update price.
- input():void - input a **softDrink** from user.
- toString():String - return the string of format: **code**, **make**, **volume**, **price**.

Write the class **SoftDrinkList** that extends **SoftDrink** and implements the interface **I_List**. This class uses **ArrayList** to manage the list of softDrinks and implements methods in **I_List** as below:

- void addFromFile(String fName): load the list from the file data.dat.
- void saveToFile(List<SoftDrink> list, String fName); save list to the file.
- void findSoftDrinkByCode(String code); find and print out all softDrinks from the list by code. If code does not exist, print out "Not found".
- void findSoftDrinkByMake(String make); find and print out all softDrinks from the list by make. If make does not exist, print out "Not found".
- void findSoftDrinkByVolume(int volume);); find and print out all softDrinks from the list by volume. If volume does not exist, print out "Not found".
- SoftDrink findSoftDrinkWithMaxPrice(); return the softDrink with the highest price (suppose in the list only one softDrink with the highest price).
- SoftDrink findSoftDrinkWithMinPrice(); return the softDrink with the lowest price (suppose in the list only one softDrink with the lowest price).
- void countClock(); count and print out the total number of softDrinks from the list (called from SortAndPrint() and run in SortAndPrint()).
- void sortAndPrint(); sort softDrinks by **ascending code** and print out the list of softDrinks with the format: **code**, **make**, **volume**, **price**.

CQ14.5 - You are supplied a project for managing books (file Exam5.rar).
This project includes:

(1) I_List.java: Interface for managing a list
(2) I_Menu.java: Interface declared methods for a menu in console program.
(3) Inputter.java: Class that supports data input from users
(4) Item.java: Class represents a item
(5) Menu.java: the Menu class
(6) Book.java: Class represents a book
(7) BookList.java: Class for a list of books
(8) BookListUse.java: Main program.

Describe:

Each **book** includes the following information: **code** (String), **make** (String), **name** (String), **year** (int).
The file **data-in.txt** that stores the book list initially has the following format:

```
B001-Prentice Hal-Computer Organization and Architecture-2012
B002-Prentice Hall-First Course in Database Systems-2008
B003-Wiley-Data Structures and Algorithms in Java-2014
B004-Pearson-Core Java 1: Fundamentals-2018
B005-Pearson-Core Java 2: Advanced features-2019
```

This list is uploaded, processed and then saved to **data-out.txt** in the following format:

```
B001; PRENTICE HAL; COMPUTER ORGANIZATION AND ARCHITECTURE; 2012
B003; WILEY; DATA STRUCTURES AND ALGORITHMS IN JAVA; 2014
B004; PEARSON; CORE JAVA 1: FUNDAMENTALS; 2018
B005; PEARSON; CORE JAVA 2: ADVANCED FEATURES; 2019
```

You works:

Implement the class **Book** extending from **Item** (i.e. Item is a superclass and Book is a subclass) with the following information:

Book
-name:String -year:int
+Book() +Book(code:String, make:String, name:String, year:int) +getName():String +getYear():int +setName(name:String):void +setYear(year:int):void +input():void +toString():String

Where:

- getName():String - return name.
- getYear():int - return year.
- setName(name: String):void - update name.
- setYear(year:int):void - update year.
- input():void - input a **book** from user
- toString():String - return the string of format: **code**, **make**, **name**, **year**

Write the class **BookList** that extends **Book** and implements the interface **I_List**. This class uses **ArrayList** to manage the list of books and implements methods in **I_List** as below:

- void addFromFile(String fName): load the list from the file data-in.txt.
- void saveToFile(String fName): save list to the file data-out.txt.
- int find(String code): find the code of a book in the list.
- void addBook(): add a new book, with input.
- void findBookByMake(String make); find and print out all books from the list by make. If make does not exist, print out "Not found".
- void findBookByPartOfName(String name); find and print out all books from the list by part of name. If the information does not exist, print out "Not found".
- void remove(): delete a book by code, with input.
- void update(): update a book by code, with input. Notes: If information is blank, the old information will not be changed.
- void sortAndPrint(); sort books by **descending year** and print out the list of books with the format: **code**, **make**, **name**, **year**.