Spring 2016

Object-Oriented Programming

Programming Assignment #3 - Library Information System

Name : Kim Da Bin
Student Id : 2015004375
Class NO : Class_1
Major : Computer Science Engineering

Prof. : Lee Choon Hwa
Assistant Teacher : Kena Alexander

Due Date : 2016.06.20

1. Code of Assignment_3

- Interface

-Lonable Interface

```
package OOP_assignment_3;

public interface Lonable {

    public abstract boolean loanItem();

}
```

- Reservable Interface

```
package OOP_assignment_3;

public interface Reservable {

    public abstract boolean reserveItem();
}
```

- Class

-LibraryItem Class

```
package OOP_assignment_3;

public abstract class LibraryItem implements Comparable<Object>{

    private String id;
    private String title;
    private String author;
    private int availableUnits;

    public LibraryItem(String id, String title, String author,int availableUnits) {

        this.id = id;
        this.title = title;
        this.author = author;
        this.availableUnits = availableUnits;
    }

    public int getAvailableUnits() {
        return availableUnits;
    }

    public void setAvailableUnits(int availableUnits) {
        this.availableUnits = availableUnits;
    }

    public String getId() {
        return id;
    }

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }

    @Override
    public String toString() {
        return "Title: " + title + "\n"+"ID: " + id + "\n"+ "Author: " + author + "\n"+"Units Available: "
                + availableUnits + "\n"+"---------------------------\n";
    }

    @Override
    public boolean equals(Object obj) {
        if (obj == null) return false;
        if (this.getClass() != obj.getClass()) return false;
        LibraryItem other = (LibraryItem) obj;
        return this.id.equals(other.id);
    }

    public int compareTo(Object obj){

        if(obj==null) throw new NullPointerException("The object is null.\n");
        if(this.getClass()!=obj.getClass()) throw new ClassCastException("The object is not of the same type.\n");
        LibraryItem other=(LibraryItem) obj;
        return this.id.compareTo(other.id);
    }

    public abstract boolean returnItem();
}
```

-GeneralWorks Class

```
package OOP_assignment_3;

public class GeneralWorks extends LibraryItem implements Lonable{

    public GeneralWorks(String id,String title,String author){

        super(id,title,author,5);
    }

    public boolean loanItem(){

        if(this.getAvailableUnits()>0){

            int tmp=this.getAvailableUnits();
            tmp--;
            this.setAvailableUnits(tmp);
            return true;
        }
        else return false;
    }

    public boolean returnItem(){

        int tmp=this.getAvailableUnits();

        if(tmp<5){
            tmp++;
            this.setAvailableUnits(tmp);
            return true;
        }
        else return false;
    }
}
```

-ReservedWorks Class

```java
package OOP_assignment_3;

public class ReservedWorks extends LibraryItem implements Reservable {

    public ReservedWorks(String id,String title,String author){

        super(id,title,author,1);
    }

    public boolean reserveItem(){

        int tmp=this.getAvailableUnits();

        if(tmp>0){
            tmp--;
            this.setAvailableUnits(tmp);
            return true;
        }
        else return false;
    }

    public boolean returnItem(){

        int tmp=this.getAvailableUnits();

        if(tmp==0){
            tmp++;
            this.setAvailableUnits(tmp);
            return true;
        }
        else return false;

    }
}
```

-Student Class

```
package OOP_assignment_3;

import java.util.*;

public class Student implements Comparable<Object> {

    private String name;
    private String id;
    private int numLoans;
    private int numReserves;

    HashSet<GeneralWorks> onLoan;
    HashSet<ReservedWorks> onReserve;

    public Student(String name,String id){
        this.name=name;
        this.id=id;
        this.numLoans=0;
        this.numReserves=0;

        onLoan=new HashSet<GeneralWorks>();
        onReserve=new HashSet<ReservedWorks>();
    }

    public String getName() {
        return name;
    }

    public String getId() {
        return id;
    }

    public int getNumLoans() {
        return numLoans;
    }

    public int getNumReserves() {
        return numReserves;
    }

    @Override
    public boolean equals(Object obj) {

        if (obj == null) return false;
        else if (getClass() != obj.getClass()) return false;
        else {
            Student other = (Student) obj;
            return this.id.equals(other.id);
        }

    }

public int compareTo(Object obj){

    if(obj==null) throw new NullPointerException("The object is null.");
    if(this.getClass()!=obj.getClass()) throw new ClassCastException("The object is not of the same type.");
    Student other=(Student) obj;
    return this.id.compareTo(other.id);
}

public String toString(){

    String str="============Student Record============\n"+"Name: "+this.name+"\n"+"id number: "+this.id+"\n"
            +"onLoan: "+this.numLoans+"\n"+"onReserve: "+this.numReserves+"\n"
            +"------------onloan------------\n";


    for(GeneralWorks it:onLoan){

        str+=it.toString();
    }
    str+="------------onreserve------------\n";
    for(ReservedWorks it_2:onReserve){

        str+=it_2.toString();
    }
    str+="=====================================\n";

    return str;

}
```

```
    public boolean acquireItem(LibraryItem obj){

        ReservedWorks t=new ReservedWorks("1","tmp","tmp");

        if(obj.getClass()==t.getClass()){
            if(!onReserve.contains(obj)){

                ReservedWorks tmp=(ReservedWorks)obj;
                if(tmp.reserveItem()==false) return false;
                else {
                    this.numReserves++;
                    onReserve.add(tmp);
                    return true;
                }
            }
            else return false;
        }
        else{
            if(!onLoan.contains(obj)){

                GeneralWorks tmp=(GeneralWorks)obj;
                if(tmp.loanItem()==false) return false;
                else{
                    this.numLoans++;
                    onLoan.add(tmp);
                    return true;
                }
            }
            else return false;
        }
    }

    public boolean releaseItem(LibraryItem obj){


        if(obj.returnItem()==true){

            if(onReserve.contains(obj)){
                onReserve.remove(obj);
                this.numReserves--;
                return true;
            }
            else if(onLoan.contains(obj)){
                onLoan.remove(obj);
                this.numLoans--;
                return true;
            }
            else return false;
        }
        else return false;
    }

}
```

-LibraryInfoSystem Class

```
package OOP_assignment_3;

import java.io.*;
import java.util.*;

public class LibraryInfoSystem {

    private static final String BOOKSINPUTFILE ="/Users/DabinKIM/Documents/workspace/1_OOP_2015004375_KIM/src/OOP_assignment_3/booksinputfile.txt";
    private static final String STUDENTSINPUTFILE ="/Users/DabinKIM/Documents/workspace/1_OOP_2015004375_KIM/src/OOP_assignment_3/studentsinputfile.txt";
    private static HashMap<String,LibraryItem> Books;
    private static HashMap<String,Student> Students;

     private void init(){

         try{
             Scanner scanner=new Scanner(new File(BOOKSINPUTFILE)).useDelimiter("#");
             Books=new HashMap<String,LibraryItem>();

             while(scanner.hasNext()){

                 String[] in=new String[4];

                 for(int i=0;i<4;i++){
                     in[i]=scanner.next();
                 }
                 scanner.nextLine();

                 if(in[3].equals("G")){
                     GeneralWorks tmp=new GeneralWorks(in[0],in[1],in[2]);
                     Books.put(in[0],tmp);
                 }
                 else{
                     ReservedWorks tmp=new ReservedWorks(in[0],in[1],in[2]);
                     Books.put(in[0],tmp);
                 }
             }

             scanner.close();

         }catch(FileNotFoundException e){
             System.out.println(e.getMessage()+" Cannot find the FILE!");
             return;
         }
         try{
             Scanner scanner=new Scanner(new File(STUDENTSINPUTFILE)).useDelimiter("#");
             Students=new HashMap<String,Student>();

             while(scanner.hasNext()){

                 String[] in=new String[2];

                 for(int i=0;i<2;i++){
                     in[i]=scanner.next();
                 }
                 scanner.nextLine();

                 Student tmp=new Student(in[0],in[1]);
                 Students.put(in[1],tmp);
             }

             scanner.close();

        } catch(FileNotFoundException e){
            System.out.println(e.getMessage()+" Cannot find the FILE!");
            return;
        }
    }
```

```
    public static void main(String[] args) {

        LibraryInfoSystem input=new LibraryInfoSystem();
        input.init();
        int number=0;

        while(number!=5){

            System.out.println("*******Library Management System*******");
            System.out.println("(1) Borrow an Item:");
            System.out.println("(2) Return an Item:");
            System.out.println("(3) Search for Book");
            System.out.println("(4) Search for Student");
            System.out.println("(5) Exit the program");
            System.out.println("****************************************");

            Scanner keyboard=new Scanner(System.in);
            number=keyboard.nextInt();

           switch(number){
               case 1:
                   System.out.println("Please enter Student's ID:");
                   String id=keyboard.next();
                   if(Students.containsKey(id)){
                       Student working=Students.get(id);
                       System.out.println("Please enter the id of the item to borrow:");
                       String book_id=keyboard.next();
                       if(Books.containsKey(book_id)){
                           if(working.acquireItem(Books.get(book_id))) System.out.println("Book successfully loaned.");
                           else System.out.println("Unable to loan.");
                       }
                       else System.out.println("The book doesn't exist!");
                   }
                   else System.out.println("The student doesn't enrolled!");
                   break;
               case 2:
                   System.out.println("Please enter Student's ID:");
                   String id_2=keyboard.next();
                   if(Students.containsKey(id_2)){
                       Student working=Students.get(id_2);
                       System.out.println("Please enter the id of the item to return:");
                       String book_id=keyboard.next();
                       if(Books.containsKey(book_id)){
                           if(working.releaseItem(Books.get(book_id))) System.out.println("Book successfully returned.");
                           else System.out.println("Unable to return.");
                       }
                       else System.out.println("The book doesn't exist!");
                   }
                   else System.out.println("The student doesn't enrolled!");
                   break;
               case 3:
                   System.out.println("Please enter the item's id number:");
                   String book_id=keyboard.next();
                   if(Books.containsKey(book_id)){
                       LibraryItem search=Books.get(book_id);
                       System.out.println(search.toString());
                   }
                   else System.out.println("The book doesn't exist!");
                   break;
               case 4:
                   System.out.println("Please enter the student's id number:");
                   String id_3=keyboard.next();
                   if(Students.containsKey(id_3)){
                       Student search=Students.get(id_3);
                       System.out.println(search.toString());
                   }
                   else System.out.println("The student doesn't enrolled!");
                   break;
               case 5:
                   System.out.println("Exits the Program!");
                   break;
           }

       }
   }
}
```

2. Output of Assignment_3

-Borrow a book

```
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
1
Please enter Student's ID:
2015877082
Please enter the id of the item to borrow:
4
Book successfully loaned.
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
1
Please enter Student's ID:
2015877082
Please enter the id of the item to borrow:
4
Unable to loan.
```

-Return a book

```
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
2
Please enter Student's ID:
2015877082
Please enter the id of the item to return:
4
Book successfully returned.
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
2
Please enter Student's ID:
2015877082
Please enter the id of the item to return:
5
Unable to return.
```

-Searching for a student

```
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
4
Please enter the student's id number:
2015877082
============Student Record============
Name: Torrie Oberlander
id number: 2015877082
onLoan: 1
onReserve: 1
------------onloan------------
Title: One Day in the Life of Ivan Denisovitch
ID: 4
Author: Alexander Solzhenitsyn
Units Available: 4
--------------------------
------------onreserve------------
Title: The Merchant of venice
ID: 5
Author: William shakespeare
Units Available: 0
--------------------------
====================================
```

-Searching for a book

```
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
3
Please enter the item's id number:
10
Title: Origin of species
ID: 10
Author: charles Darwin
Units Available: 1
--------------------------
```

-End the program

```
*******Library Management System*******
(1) Borrow an Item:
(2) Return an Item:
(3) Search for Book
(4) Search for Student
(5) Exit the program
****************************************
5
Exits the Program!
```

3. Explanation of Assignment_2

- Lonable interface
  - This interface is for all Loanable classes.
  - public abstract boolean loanItem() : This method is abstract. So, it will be defined in the class which implements this interface.

- Reservable interface
  - This interface is for all Reservable classes.
  - public abstract boolean reserveItem():This method is abstract. So, it will be defined in the class which implements this interface.

- LibraryItem Class
  - This class implements the Comparable interface, and it is abstract class.
  - instance variables : It has 4 private instance variables.
    - String id : It will store the id of the book.
    - String title : It will store the Title of the book.
    - String author : It will store the author of the book
    - int availableUnits : It will store how many books student can reserve or loan.
  - public LibraryItem(String id, String title, String author,int availableUnits) : It's a constructor of this class. It will set all the instance variables.
  - public int getSomething() : Its name is getter. It will get instance variable(Something) which can't access out of the class.
  - public void setSomething : Its name is setter. It will set instance variable(Something) which can't access out of the class.
  - public String toString() : It will change all of instance type to String and store in a String instance variable, and return it. Than, we can see the information easily. It will show us the information of the book.
  - public boolean equals(Object obj) : It will compare obj with other object which you want. So, if their ids are same, it will return a true. But other cases will all return false.
  - public int compareTo(Object obj) : It will compare two object's id and return true or false. But if it has some error, it will throw the error.
  - public abstract boolean returnItem() : Because it is an abstract class, so it has a abstract method. this method will get the body in the class which extends this class.

- GeneralWorks Class

  - This class extends LibraryItem class and implements Lonable interface.
  - public GeneralWorks(String id,String title,String author) : It's a constructor of this class. It will set 3 of the instance variables(id, title, author) use parameter, and set availableUnits as "5" by super.
  - public boolean loanItem() : It will check the loan item can borrow or not. It will get the availableUnits of the item and if it's bigger than 0, it will be decreased and return true. this mean student can loan the Item. Other cases will return false.
  - public boolean returnItem() : It will used in to return the item. You will get the availableUnits and if it is smaller than "5", it will be increased and return true. this mean student can return it. Other cases will return false.

- GeneralWorks Class

  - This class extends LibraryItem class and implements Reservable interface.
  - public ReservedWorks(String id,String title,String author) : It's a constructor of this class. It will set 3 of the instance variables(id, title, author) use parameter, and set availableUnits as "1" by super.
  - public boolean reserveItem() : It will check the loan item can borrow or not. It will get the availableUnits of the item and if it's bigger than 0, it will be decreased and return true. this mean student can loan the Item. Other cases will return false.
  - public boolean returnItem() : It will used in to return the item. You will get the availableUnits and if it is "0", it will be increased and return true. this mean student can return it. Other cases will return false.

- Student Class

  - This class implements the Comparable interface.
  - instance variables : It has private 6 instance variables.
    - String name : It will store the name of the student.
    - String id : It will store the id of the student.
    - int numLoans : It will store how many books the student loans.
    - int numReserves : It will store how many books student reserves.
    - HashSet<GeneralWorks> onLoan : It will store information of the book which student loaned.

- HashSet<ReservedWorks> onReserve : It will store information of the book which student reserved.
- Why HashSet? HashSet is easy to add, find or remove the element and if I enter same information, it will ignore it by itself. I don't have to make another check. So I chose HashSet.

- public Student(String name,String id) : It's a constructor of this class. It will set 2 of the instance variables(name, id) use parameter, set numLoans and numReserves as "0" and initialized onLoan and onReserve.
- public int getSomething() : Its name is getter. It will get instance variable(Something) which can't access out of the class.
- public void setSomething : Its name is setter. It will set instance variable(Something) which can't access out of the class.
- public boolean loanItem() : It will check the loan item can borrow or not. It will get the availableUnits of the item and if it's bigger than 0, it will be decreased and return true. this mean you can loan the Item. Other cases will return false.
- public boolean equals(Object obj) : It will compare obj with other object which you want. So, if their ids are same, it will return a true. But other cases will all return false.
- public int compareTo(Object obj) : It will compare two object's id and return true or false. But if it has some error, it will throw the error.
- public String toString() : It will change all of instance type to String and store in a String instance variable, and return it. Than, we can see the information easily. It also include the informations of the student and the books in the onLoan and onReserved.
- public boolean acquireItem(LibraryItem obj) : This method will help the student loan or reserve the books. if the book is reservable and same book isn't in the onReserve, information of it will be put in the onReserve and numReserves will increase. This mean student reserves a book, so method will return true. Other case about reserving will return false. If the book is lonable and same book isn't in the onLoan, information of it will be put in the onLoan and numLoans will increase. This mean student loans a book, so method will return true. Other case about loan will return false.
- public boolean releaseItem(LibraryItem obj) : This method is used to return the item. If the book can release and in the onReserve, it will be removed in the onReserve and numReserves will decrease. This mean student is success to return the item, so return the true. If the book can release and in the onLoan, it will be removed in the onLoan and numLoans will decrease. This mean student is success, so return the true. Other cases will return false.

- LibraryInfoSystem Class

  - instance variables : It has 4 private static instance variables.

- final String BOOKSINPUTFILE : It will store the path of the input text which save all date about Books.
- final String STUDENTSINPUTFILE : It will store the path of the input text which save all date about Students.
- HashMap<String,LibraryItem> Books : It will store information of all books.
- HashMap<String,Student> Students : It will store information of all students.
- Why HashMap? It is similar to HashSet, but it implements Map interface. So, it use “key” and “Value” to manage all of the data. It is easy to put element in the HashMap, and If I know the “Key”, I can get the information easily. So I chose HashMap.

- private void init() :
  - try-catch for books : This method will use scanner call the file by BOOKSINPUTFILE. and store the informations in Books(HashMap) until all the informations are stored. If the book is General, it will stored as GeneralWorks type. If it is Reserved, it will stored as ReservedWorks type. If the path of file is not correct or can’t find the file, this error will be catch by the FileNotFoundException and get error message.
    +) scanner.nextLine() : It is for remove unnecessary “#” which can disturb the program.
  - try-catch for students : This method will use scanner call the file by STUDENTSINPUTFILE. and store the informations in Students(HashMap) until all the informations are stored. If the path of file is not correct or can’t find the file, this error will be catch by the FileNotFoundException and get error message.
    +) scanner.nextLine() : It is for remove unnecessary “#” which can disturb the program.
- public static void main(String[] args) : All of process will start in this method. First, it will call init method to have all the date about books and students. and while number is not “5”, It will show you the menu. You can choose a work which program should do. I use switch to make process more simple.
  - 1) Through the student id and book id, student will borrow book by the entered book id. If student id is not enrolled or the book doesn’t exist or other error occurs, it will get a error message.
  - 2) Through the student id and book id, student will return book by the entered book id. If student id is not enrolled or the book doesn’t exist or other error occurs, it will get a error message.
  - 3) Through the entered book id, It will show you the information of the book. If books doesn’t exist, it will get a error message.
  - 4) Through the entered student id, It will show you the information of the student include the information about loaned or reserved books. If student id is not enrolled, it will get a error message.
  - 5) Exit the Program.