

Coursework 4wcm0022

Library Loan System

Task 1:

I started off by making individual classes for the four items. Each having four string attributes:

```
Private String Title, Private String Details, Private String Loan,  
Private String ReturnDate
```

I used the keyword "Private" to ensure that these attributes are only being used in the class only.

Whereas all the other methods and constructors are declared as "Public" to be used later on in the tester class.

Using the "is a" Inheritance I linked all the childclasses to one superclass "Library" which contained the ID numbers of the items which were to be unique and thus i used the "protected" when declaring the variable LibraryID.

Each of the item classes has multiple constructors for member initialization and accessors and mutators. The setters and getters made were for each individual part of the array of the item classes such as the Item name , Title , Author/Publisher/Artist, whether the item was on loan or not and the Return Date. I used the keyword "this" to assign the value of each attribute without having to actually assign it with a constant value.

Task 2:

In the tester class I created 4 arrays for each item, of not more than 5 indexes and populated them with real examples within the same class, using Polymorphism to create a link between the Library class and the item arrays.

```
Library [] books = new Book[5];
```

An item counter and a while loop were used to print out the populated arrays in order, where item = 1 represented books []

For the correct formatting of the report I created a method in the Library class (`DisplayItemNumber()`), which self generated a counter for the Library ID of the items (in this case 20).

So as to generate a report exactly the same as the sample report provided I kept variable names for each of the attributes needed

```
(l1.DisplayItemNumber(), ItemName, Title, Loan , Date)
```

For the corresponding counter `"item"` I assigned the variable `"ItemName"` as that of the corresponding item.

```
while (item == 1)
{
    ItemName = "Book";
```

Along with the while loop, an internal for loop was used to loop through all the 5 indexes of each array.

```
For (int ItemCounter = 0; ItemCounter <= 4; ItemCounter++)
```

For the variable `"Title"` Casting of the array was used along with an accessor which extracted data from the selective array

```
Title = (((Book)books[ItemCounter]).getBookTitle());
```

The same was done when obtaining the loan value

```
Loan = (((Book)books[ItemCounter]).getLoan());
```

I predefined the variable `Date` as a string with the value of `"000000"` for those items that weren't on loan and had no return date. I made it a `static` variable so as to share it along with all the other instances.

```
String Date = "000000";
```

Since a summary of the report had to be shown too an `if` condition was used to count the items that were on loan, the method from the `LoanItem` class was used that increments itself each time it is called

```
if (Loan == "yes")
{
    loanitem.GetOnLoanCounter();
```

While the loan equals to yes a return date is automatically calculated, by adding the current date with the loanable duration. This was all done through a series of `if` loops along with another array that stored all the days of the month which used the keyword `"final"`

```
final int[] daysPerMonth = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
```

To get the current day, month and year I imported

```
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
```

Using selective parts of ("yyyy/MM/dd") I stored the individual values of the day, month and year in the day, month and year variables .

Since the three were a string data type I converted them integer an integer value using "parseInt"

```
Day = Integer.parseInt(dayFormat.format(currentDate));
```

After converting it into an integer I added the loan duration, "**int LoanDuration = 14**" which I kept as 2 weeks for all the items, and checked whether the value of day was greater than the total number of days in that particular month (making use of the daysPerMonth array).

If the value of the day was greater than the days of that month then I subtracted the days of the month with the current variable day and incremented the value of month by 1.

To acquire the proper format of the 6 digit date I added another if statement that added a "0" to the beginning of any digit that was below 10.

```
if (day < 10)
{
    Date = String.valueOf(year) + String.valueOf(month) + "0" +
String.valueOf(day);
}
else if (month < 10)
{
    Date = String.valueOf(year) + "0" + String.valueOf(month) +
String.valueOf(day);
}
```

After doing so I converted the integer values of day, month and year back to a string format since the original return type of the loanable items was string in the same format (YY/MM/DD)

```
Date = String.valueOf(year) + String.valueOf(month) +
String.valueOf(day);
```

After this I printed just one line of the report, whilst incrementing the ItemsCount by 1 which was a variable to count the total number of Items in the whole Library System.

```
ItemCount++;
```

After this to continue the while loop for the next item another variable item was used which was also incremented by 1.

```
item++;
```

Getting reference from the sample report shown i only displayed the Item Number , Item Name , Title , Loan and Return Date

This whole process was repeated for all the other items.

Just to make the Report stand out and look more presentable I added a header dotted line, beneath it on the right hand side I displayed the current date as well as the format it was shown in. Apart from this I showed what data was to be shown under each respective column.

```
----- REPORT -----  
Date: 21/01/07 (YYMMDD)  
Item# | Type | Title | Loan | Date |
```

The summary of the report was shown right at the end of the report with the total number of items as well as the number of items on loan.

Task 3:

I created three methods within the test class, one for the headings , another to print the items as a report and a third to write the report to a text file.

```
PrintHeadings()  
PrintItems()  
FileHandling()
```

The `PrintHeading()` method is used to print the top-most part of the report which shows the heading “Report” as well as the current date today in the format (YYMMDD). I also added the headings for each item to be displayed in the report.

```
"Item# | Type | Title | Loan | Date |"
```

The second method `PrintItems()` is used to print all the items in the arrays, written in the parameters for each of the items arrays. Using the `"System.out.print"` keyword to print the whole report. With the help of `Escape Sequences` to align the data as desired (`\n` & `\t`).

I declared the arrays that i had made above for all the items inside the parentheses of the method name.

```
public static String PrintItems(Library[] books , Library[] dvd ,  
Library[] cd , Library[] magazine)
```

There were three objects that were used in this method:

```
Library library = new Library();  
LoanItem loanitem = new LoanItem();  
Date currentDate = new Date();
```

The Library and LoanItem objects were used as counters while the Date object was used to obtain the current date.

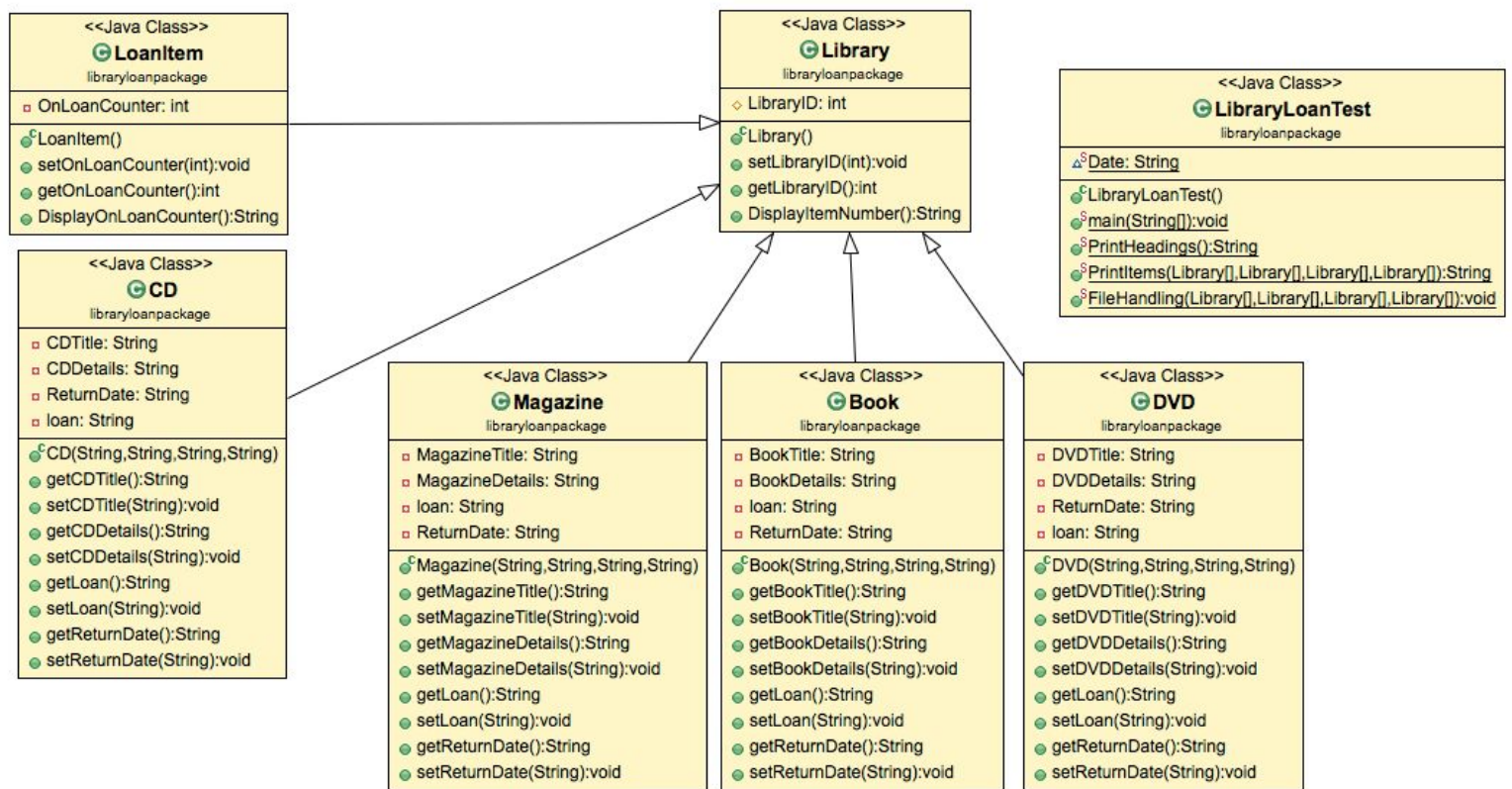
The third method `FileHandling()` is used to write the whole report into a text file. I used two imports

```
import java.io.FileWriter;  
import java.io.IOException;
```

To both write in the file using `"myWriter.write"` when obtaining the values from each item I used `Casting` alongside using the array and the get method and a loop ensuring the code to pass through all 5 individual indexes of each item.







For the summary of the whole report, since my `getLibraryId()` method always increments itself whenever called to so i had to subtract the answer by one and as far as the on loan counter is concerned the `GetOnLoanCounter()` was used.

Using the `try...catch` technique if the file is successfully written to the text file it prints out "successful" else the exception is dealt with.



I created a UML for my project using Eclipse to give a representation of the whole code.

Key:

-  Methods
-  Protected
-  Static
-  Private
-  Classes
-  Static declared in the main test class