Name: Ulkar Chobanova

Neptun code: WQ1C71

OOP Big Assignment Part 1

Description of the task:

The university library management system maintains a diverse collection of items and facilitates borrowing and returning processes for members.

Items, have an id, a title, an author, page count, topic. The system tracks how many times an item has been borrowed. The item could be one a book, thesis, or research paper. Each item might have multiple copies and new copies can be added. If an item is borrowed, and there are more copies, it should still be available for others to be borrowed; if all copies were borrowed then it would not be available. When returning an item, it would be available for borrowing again if not damaged. A book could get damaged due to misuse and when returning it should be marked as damaged or not, if it's damaged the item needs repair and a repair must be requested.

An item can have a digital copy. By default the item wouldn't have one, but it can be created. When created, it would have same id as the item. Members can view the item, and if they want they can download the digital copy. The number of views and downloads is kept. The library should be able to find most viewed and most downloaded items.

To borrow items, users must be members of the library, they have their own id, name, email, and they have a category of student, faculty, or alumni. Each user keeps an active list of currently borrowed items and initially it should not exceed 3 per member. They should also keep the number of total borrowed items (when a book is successfully returned, then it is added to the list). For each 3 items that are returned, the member's loyalty would be appreciated by increasing the number of allowed items. However, to be able to consider a return of an item as a loyalty point, the condition of the item should be marked as preserved (not damaged).

The Library has a Repairs Section. It keeps count each time an item is sent to be repaired. If damage is severe (use randomness) then the book should be removed from the library to the archives of the library. If successfully repaired then it would normally be back to be available. The Library does a periodical check every now and then to send the items that need repair to repair Section, and to do the repair process.

The library stores the collection of items, It also tracks active transactions (borrowing and returning) and provides search functionalities by title and author to help users locate desired materials. The Library can add new items to its collection and manages the borrowing and returning of items. The library also keeps track of all their members, and can search for most active members. Also the people can find most popular items based on the number of times they have been borrowed: the most popular digital item, and the most popular in general taking in account both physical and digital copies.

The main simulation must initialize the Library. Create items and add them to the library. Some items might have digital copies. Create members and make borrowing and returning transactions and other necessary operations such as searching for specific items, or making repairs. Read the input from a file, and use as many input files as needed.

Input file examples

items.txt

- Format

```
ID Title Author Page_Count Topic Type Copies
```

- Sample input

```
1 Clean_Code Robert_C_Martin 464 Software_Engineering Book 3
2 The_Pragmatic_Programmer Andrew_Hunt 352 Software_Engineering Book 2
3 Design_Patterns Erich_Gamma 395 Software_Engineering Book 2
4 Refactoring Martin_Fowler 448 Software_Engineering Book 3
```

members.txt

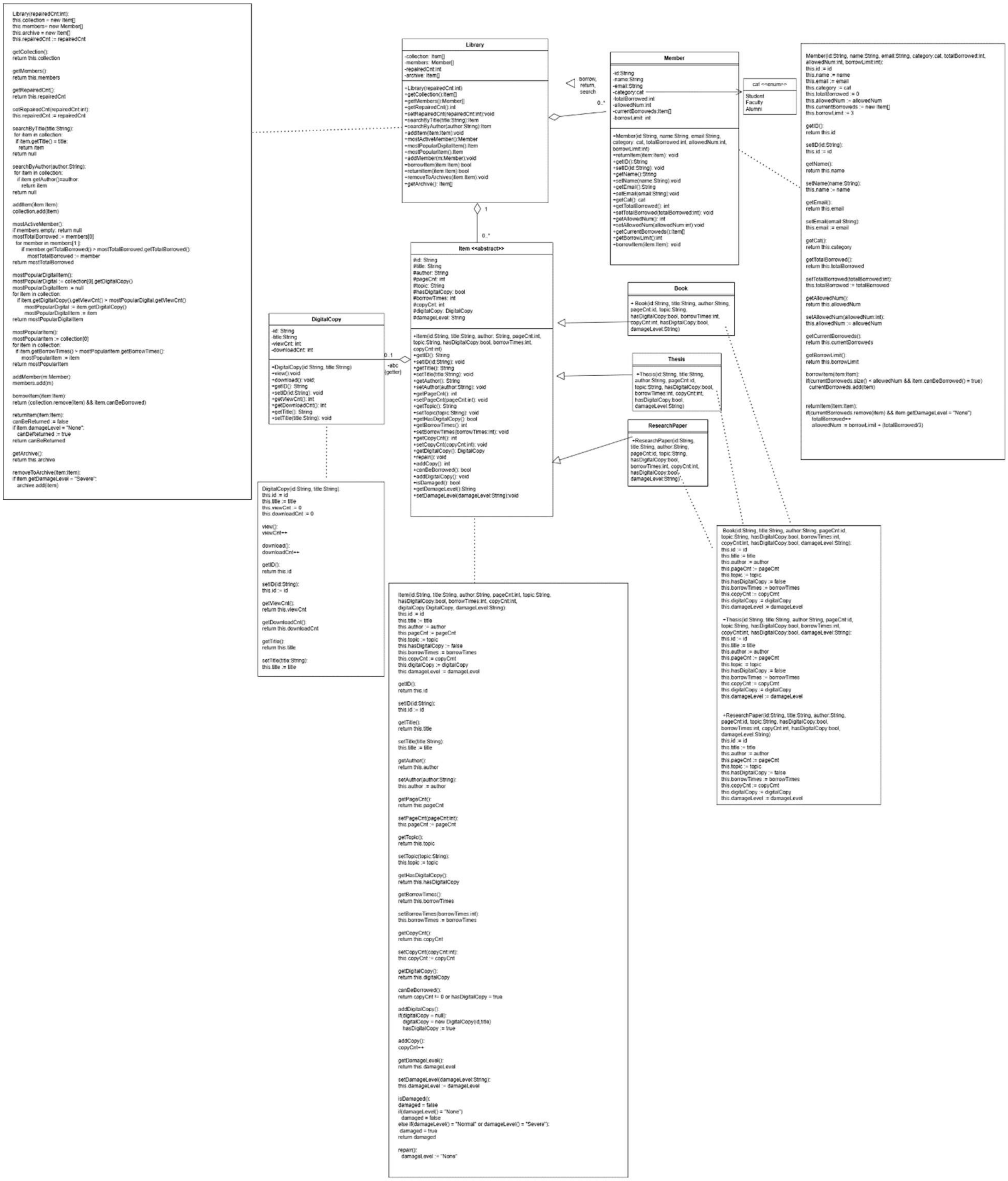
- Format

```
ID Name Email Category
```

- Sample input

```
1 Bence_Nagy bence@elte.hu Student
2 Eszter_Kovacs eszter@elte.hu Faculty
3 John_Smith john@elte.hu Alumni
4 Levente_Toth levente@elte.hu Student
5 Sophia_Martinez sophia@elte.hu Faculty
```

Class Diagram


Library
-collection: Item[]
-members: Member[]
-repairedCnt:int
-archive: Item[]
+Library(repairedCnt:int)
+getCollection():Item[]
+getMembers():Member[]
+getRepairedCnt():int
+setRepairedCnt(repairedCnt:int):void
+searchByTitle(title:String):Item
+searchByAuthor(author:String):Item
+addItem(item:Item):void
+mostActiveMember():Member
+mostPopularDigitalItem():Item
+mostPopularItem():Item
+addMember(m:Member):void
+borrowItem(item:Item):bool
+returnItem(item:Item):bool
+removeToArchives(item:Item):void
+getArchive(): Item[]
borrow,
return,
search
0..*
1
0..*
Member
-id:String
-name:String
-email:String
-category:cat
-totalBorrowed:int
-allowedNum:int
-currentBorroweds:Item[]
-borrowLimit: int
+Member(id:String, name:String, email:String, category: cat, totalBorrowed:int, allowedNum:int, borrowLimit:int)
+returnItem(item:Item): void
+getID():String
+setID(id:String): void
+getName():String
+setName(name:String):void
+getEmail():String
+setEmail(email:String):void
+getCat(): cat
+getTotalBorrowed(): int
+setTotalBorrowed(totalBorrowed:int): void
+getAllowedNum(): int
+setAllowedNum(allowedNum:int):void
+getCurrentBorroweds():Item[]
+getBorrowLimit():int
+borrowItem(item:Item): void
cat <<enum>>
Student
Faculty
Alumni
Item <<abstract>>
#id: String
#title: String
#author: String
#pageCnt: int
#topic: String
#hasDigitalCopy: bool
#borrowTimes: int
#copyCnt: int
#digitalCopy: DigitalCopy
#damageLevel: String
+Item(id:String, title:String, author: String, pageCnt:int, topic:String, hasDigitalCopy:bool, borrowTimes:int, copyCnt:int)
+getID(): String
+setID(id:String): void
+getTitle(): String
+setTitle(title:String): void
+getAuthor(): String
+setAuthor(author:String): void
+getPageCnt(): int
+setPageCnt(pageCnt:int): void
+getTopic(): String
+setTopic(topic:String): void
+getHasDigitalCopy(): bool
+getBorrowTimes(): int
+setBorrowTimes(borrowTimes:int): void
+getCopyCnt(): int
+setCopyCnt(copyCnt:int): void
+getDigitalCopy(): DigitalCopy
+repair(): void
+addCopy(): int
+canBeBorrowed(): bool
+addDigitalCopy(): void
+isDamaged(): bool
+getDamageLevel():String
+setDamageLevel(damageLevel:String):void
DigitalCopy
-id: String
-title:String
-viewCnt: int
-downloadCnt: int
+DigitalCopy(id:String, title:String)
+view():void
+download(): void;
+getID(): String
+setID(id:String): void
+getViewCnt(): int
+getDownloadCnt(): int
+getTitle(): String
+setTitle(title:String): void
0..1
-abc
(getter)
Book
+ Book(id:String, title:String, author:String, pageCnt:id, topic:String, hasDigitalCopy:bool, borrowTimes:int, copyCnt:int, hasDigitalCopy:bool, damageLevel:String)
Thesis
+Thesis(id:String, title:String, author:String, pageCnt:id, topic:String, hasDigitalCopy:bool, borrowTimes:int, copyCnt:int, hasDigitalCopy:bool, damageLevel:String)
ResearchPaper
+ResearchPaper(id:String, title:String, author:String, pageCnt:id, topic:String, hasDigitalCopy:bool, borrowTimes:int, copyCnt:int, hasDigitalCopy:bool, damageLevel:String)

```
Library(repairedCnt:int):
this.collection = new Item[]
this.members= new Member[]
this.archive = new Item[]
this.repairedCnt := repairedCnt

getCollection():
return this.collection

getMembers():
return this.members

getRepairedCnt():
return this.repairedCnt

setRepairedCnt(repairedCnt:int):
this.repairedCnt := repairedCnt

searchByTitle(title:String):
 for item in collection:
  if item.getTitle() = title:
    return item
return null

searchByAuthor(author:String):
 for item in collection:
   if item.getAuthor()=author:
     return item
return null

addItem(item:Item):
collection.add(item)

mostActiveMember():
if members.empty: return null
mostTotalBorrowed := members[0]
  for member in members[1:]:
     if member.getTotalBorrowed() > mostTotalBorrowed.getTotalBorrowed():
         mostTotalBorrowed := member
return mostTotalBorrowed

mostPopularDigitalItem():
mostPopularDigital := collection[0].getDigitalCopy()
mostPopularDigitalItem := null
for item in collection:
   if item.getDigitalCopy().getViewCnt() > mostPopularDigital.getViewCnt()
        mostPopularDigital := item.getDigitalCopy()
        mostPopularDigitalItem := item
return mostPopularDigitalItem

mostPopularItem():
mostPopularItem := collection[0]
for item in collection:
  if item.getBorrowTimes() > mostPopularItem.getBorrowTimes():
     mostPopularItem := item
return mostPopularItem

addMember(m:Member):
members.add(m)

borrowItem(item:Item):
return (collection.remove(item) && item.canBeBorrowed)

returnItem(item:Item):
canBeReturned := false
if item.damageLevel = "None":
   canBeReturned := true
return canBeReturned

getArchive():
return this.archive

removeToArchive(item:Item):
if item.getDamageLevel = "Severe":
   archive.add(item)
```

```
Member(id:String, name:String, email:String, category:cat, totalBorrowed:int,
allowedNum:int, borrowLimit:int):
this.id := id
this.name := name
this.email := email
this.category := cat
this.totalBorrowed := 0
this.allowedNum := allowedNum
this.currentBorroweds := new Item[]
this.borrowLimit := 3

getID():
return this.id

setID(id:String):
this.id := id

getName():
return this.name

setName(name:String):
this.name := name

getEmail():
return this.email

setEmail(email:String):
this.email := email

getCat():
return this.category

getTotalBorrowed():
return this.totalBorrowed

setTotalBorrowed(totalBorrowed:int):
this.totalBorrowed := totalBorrowed

getAllowedNum():
return this.allowedNum

setAllowedNum(allowedNum:int):
this.allowedNum := allowedNum

getCurrentBorroweds():
return this.currentBorroweds

getBorrowLimit():
return this.borrowLimit

borrowItem(item:Item):
if(currentBorroweds.size() < allowedNum && item.canBeBorrowed() = true)
  currentBorroweds.add(item)


returnItem(item:Item):
if(currentBorroweds.remove(item) && item.getDamageLevel = "None")
   totalBorrowed++
   allowedNum := borrowLimit + (totalBorrowed/3)
```

```
DigitalCopy(id:String, title:String):
this.id := id
this.title := title
this.viewCnt := 0
this.downloadCnt := 0

view():
viewCnt++

download():
downloadCnt++

getID():
return this.id

setID(id:String):
this.id := id

getViewCnt():
return this.viewCnt

getDownloadCnt():
return this.downloadCnt

getTitle():
return this.title

setTitle(title:String):
this.title := title
```

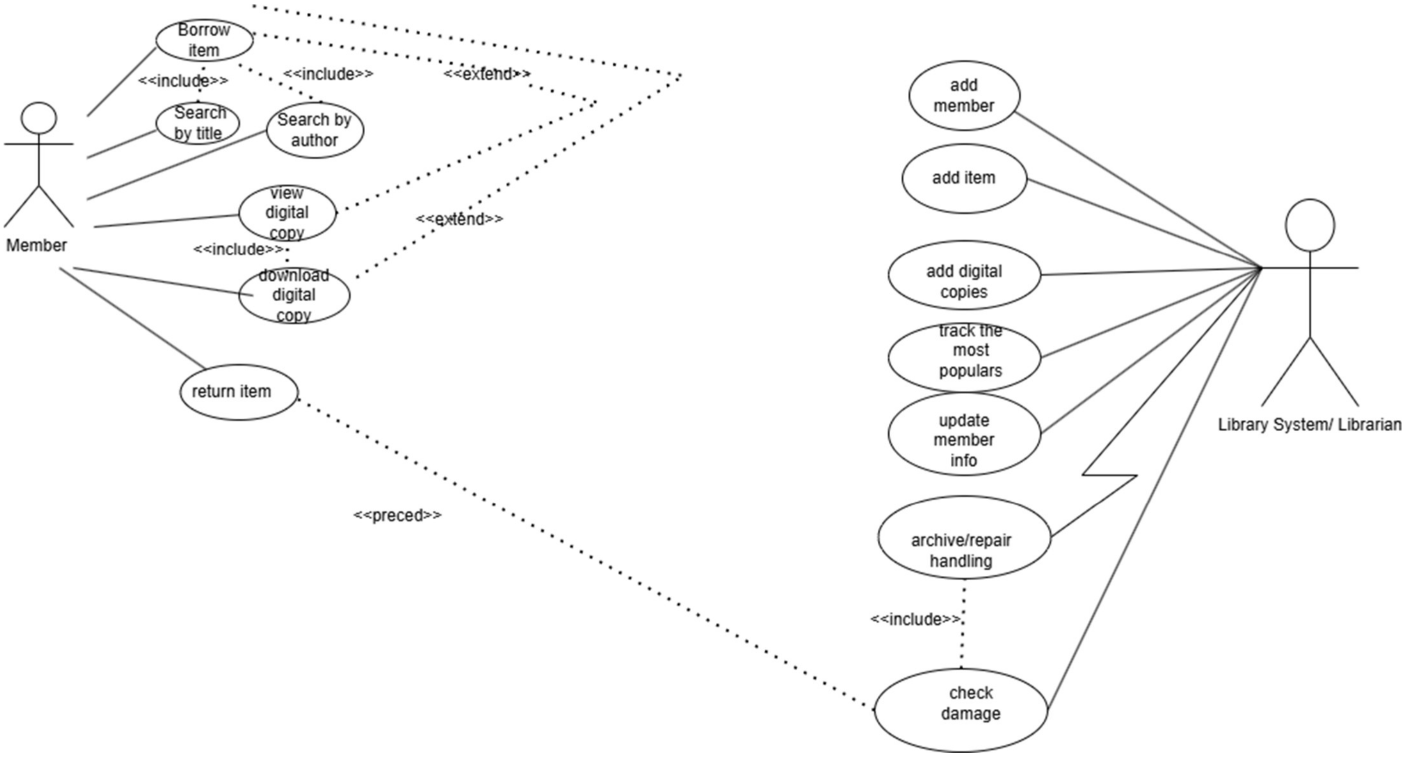
```
Item(id:String, title:String, author:String, pageCnt:int, topic:String,
hasDigitalCopy:bool, borrowTimes:int, copyCnt:int,
digitalCopy:DigitalCopy, damageLevel:String):
this.id := id
this.title := title
this.author := author
this.pageCnt := pageCnt
this.topic := topic
this.hasDigitalCopy := false
this.borrowTimes := borrowTimes
this.copyCnt := copyCmt
this.digitalCopy := digitalCopy
this.damageLevel := damageLevel

getID():
return this.id

setID(id:String):
this.id := id

getTitle():
return this.title

setTitle(title:String):
this.title := title

getAuthor():
return this.author

setAuthor(author:String):
this.author := author

getPageCnt():
return this.pageCnt

setPageCnt(pageCnt:int):
this.pageCnt := pageCnt

getTopic():
return this.topic

setTopic(topic:String):
this.topic := topic

getHasDigitalCopy():
return this.hasDigitalCopy

getBorrowTimes():
return this.borrowTimes

setBorrowTimes(borrowTimes:int):
this.borrowTimes := borrowTimes

getCopyCnt():
return this.copyCnt

setCopyCnt(copyCnt:int):
this.copyCnt := copyCnt

getDigitalCopy():
return this.digitalCopy

canBeBorrowed():
return copyCnt != 0 or hasDigitalCopy = true

addDigitalCopy():
if(digitalCopy = null):
   digitalCopy = new DigitalCopy(id,title)
   hasDigitalCopy := true

addCopy():
copyCnt++

getDamageLevel():
return this.damageLevel

setDamageLevel(damageLevel:String):
this.damageLevel := damageLevel

isDamaged():
damaged = false
if(damageLevel() = "None")
  damaged = false
else if(damageLevel() = "Normal" or damageLevel() = "Severe"):
 damaged = true
return damaged

repair():
  damageLevel := "None"
```

```
 Book(id:String, title:String, author:String, pageCnt:id,
topic:String, hasDigitalCopy:bool, borrowTimes:int,
copyCnt:int, hasDigitalCopy:bool, damageLevel:String):
this.id := id
this.title := title
this.author := author
this.pageCnt := pageCnt
this.topic := topic
this.hasDigitalCopy := false
this.borrowTimes := borrowTimes
this.copyCnt := copyCmt
this.digitalCopy := digitalCopy
this.damageLevel := damageLevel

 +Thesis(id:String, title:String, author:String, pageCnt:id,
topic:String, hasDigitalCopy:bool, borrowTimes:int,
copyCnt:int, hasDigitalCopy:bool, damageLevel:String):
this.id := id
this.title := title
this.author := author
this.pageCnt := pageCnt
this.topic := topic
this.hasDigitalCopy := false
this.borrowTimes := borrowTimes
this.copyCnt := copyCmt
this.digitalCopy := digitalCopy
this.damageLevel := damageLevel


 +ResearchPaper(id:String, title:String, author:String,
pageCnt:id, topic:String, hasDigitalCopy:bool,
borrowTimes:int, copyCnt:int, hasDigitalCopy:bool,
damageLevel:String)
this.id := id
this.title := title
this.author := author
this.pageCnt := pageCnt
this.topic := topic
this.hasDigitalCopy := false
this.borrowTimes := borrowTimes
this.copyCnt := copyCmt
this.digitalCopy := digitalCopy
this.damageLevel := damageLevel
```

Link to the class diagram:

https://drive.google.com/file/d/1mVlSIjSHWL76g8X8TrOHBrG_rpxE4Fv3/view?usp=sharing

Use-case diagram:


Borrow item
<<include>>
<<include>>
<<extend>>
Search by title
Search by author
Member
view digital copy
<<extend>>
<<include>>
download digital copy
return item
<<preced>>
add member
add item
add digital copies
track the most populars
update member info
archive/repair handling
<<include>>
check damage
Library System/ Librarian