**Classes,Objects and Inheritance Code:**

**Q.** Write a python program to implement a library using Classes and objects. Create a class students and two other classes for se and te students that inherit the student class. A se student can issue 4 books and a te student can issue 6 books.

Create a class and object for books.

Create a master access class for operating the library, create store and manipulate the objects.

**Soln:**

class student():

def \_\_init\_\_(self,name,id):

self.name=name

self.id=id

self.issued=list()

self.issues = 0

class studentSE(student):

def \_\_init\_\_(self,name,id):

super().\_\_init\_\_(name,id)

self.max = 4

self.year = 2

class studentTE(student):

def \_\_init\_\_(self,name,id):

super().\_\_init\_\_(name,id)

self.max = 6

self.year = 3

class book():

def \_\_init\_\_(self,name,isbn,count):

self.name=name

self.isbn=isbn

self.count=count

class masteraccess():

se=list()

te=list()

books=list()

def getPosnInList(self,lst,name):

flag=0

for k in range(0,len(lst),1):

if lst[k].name==name:

flag=1

break

if flag==0:

k=-1

return k

def register(self,se,te):

name = input("Enter the username you wish to enroll with: ")

id=int(input("Enter your id: "))

year=int(input("Which year are you studying in? ie.2nd or 3rd?"))

if year==2:

se.append(studentSE(name,id))

elif year==3:

te.append(studentTE(name,id))

def unregister(self,se,te):

name = input("Enter the username you wish to unregister:" )

s=masteraccess.getPosnInList(self,se,name)

t= masteraccess.getPosnInList(self, te, name)

if s>=0:

se.remove(s)

if t>=0:

te.remove(t)

def getStudentDetails(self,se,te):

try:

name = input("Enter the name of the student:")

k=masteraccess.getPosnInList(self,se,name)

if k>=0:

#if se(k).name==name:

print("Name: " + se[k].name + " Id: " + str(se[k].id) + " No of copies issued:" + str(se[k].issues))

print("The student has currently issued the following books: ",se[k].issued)

t=masteraccess.getPosnInList(self,te,name)

if t>=0:

#if te[t].name==name:

print("Name: "+te[t].name+" Id: "+str(te[t].id)+" No of copies issued:"+str(te[t].issues))

print("The student has currently issued the following books: ", te[t].issued)

if k==-1 and t==-1:

print("Student doesnt exist in records.....")

except KeyError:

print("Error Occured....Username may not exist...")

def addbook(self,books):

try:

name = input("Enter the name of the book you wish to add: ")

isbn=int(input("Enter book's ISBN: "))

no = int(input("How many book to add?"))

books.append(book(name, isbn, no))

except TypeError:

print("Invalid number by default 1 book added....")

books.append(book(name, isbn, 1))

def issuebook(self,books,se,te):

try:

name = input("Enter the name of the book you wish to issue: ")

k=masteraccess.getPosnInList(self,books,name)

if k>=0 and books[k].count>0:

na=input("Enter your name: ")

s=masteraccess.getPosnInList(self,se,na)

t=masteraccess.getPosnInList(self,te,na)

if s>=0 and se[s].issues<=se[s].max:

se[s].issued.append(books[k].name)

se[s].issues=se[s].issues+1

books[k].count=books[k].count-1

elif t>=0 and te[t].issues<=te[t].max:

te[t].issued.append(books[t].name)

te[t].issues=te[t].issues+1

books[k].count=books[k].count

else:

print("Invalid username...")

else:

print("Book not available in stock.......")

except KeyError:

print("An error occured....The book may not exist...")

def returnbook(self,books,se,te):

try:

name = input("Enter the name of the book you wish to return: ")

k=masteraccess.getPosnInList(self,books,name)

if k>=0:

if books[k].count>= 5:

print("Book stock full, Cannot accept more books.....")

else:

uname = input("Enter ur username:")

t=masteraccess.getPosnInList(self,te,uname)

s=masteraccess.getPosnInList(self,se,uname)

if t>=0:

te[t].issued.remove(name)

te[t].issues=te[t].issues-1

elif s>= 0:

se[s].issued.remove(name)

se[s].issues = se[s].issues - 1

else:

print("Username not registered....Register into the library before issuing...")

else:

print("Invalid Book")

except KeyError:

print("Error occurred.....Book may not exist......")

def display(self,se,te,books):

print("The Library has the following books: ")

for k in range(len(books)):

print("isbn: ",books[k].isbn," Name: ",books[k].name," Copies: ",books[k].count)

print("Member details are: ")

for k in range(len(se)):

print("Name: "+se[k].name+" Id: "+str(se[k].id)+" No of copies issued: "+str(se[k].issues)+"The books issued are:",se[k].issued)

for k in range(len(te)):

print("Name: "+te[k].name+" Id: "+str(te[k].id)+" No of copies issued: "+str(te[k].issues)+"The books issued are:",te[k].issued)

def accesslib(self,se,te,books):

choice = 1

print(

"Menu:\n1=Issue \n2=Return \n3=Add new book \n\n4=Unregister \n5=Register\n6=Get user details\n7=Display everything")

while choice >= 1 and choice <= 7:

try:

choice = int(input("Enter your choice:"))

except ValueError:

print("Error Occurred in choice....")

if choice == 1:

masteraccess.issuebook(self,books,se,te)

elif choice == 2:

masteraccess.returnbook(self,books,se,te)

elif choice == 3:

masteraccess.addbook(self,books)

elif choice == 4:

masteraccess.unregister(self,se,te)

elif choice == 5:

masteraccess.register(self,se,te)

elif choice == 6:

masteraccess.getStudentDetails(self,se,te)

elif choice==7:

masteraccess.display(self,se,te,books)

else:

print("Invalid choice...Exiting program....")

masteraccess.display(se,te,books)

def main():

m=masteraccess()

masteraccess.accesslib(m,[],[],[])

main()

**Exception Handling Code:**

**Q.** Create a userdefined exception issuesRestrictedError that is triggered if a student exceeds the maximum nuber of book issues allowed to him/her. Apply other inbuilt exceptions wherever required.

**Soln:**

1.**Defining a new exception class by inheriting the inbuilt exception class**.

class issuesRestrictedError(Exception):

def display\_error(self):

print("Library Rule: 2nd year students can issue not more than 4 and 3rd years can issue not more than 6 books...")

--------------------------------------------------------------------------------------------

2.**Code where the exception is raised caught and dealt with…**

def issuebook(self,books,se,te):

try:

name = input("Enter the name of the book you wish to issue: ")

k=masteraccess.getPosnInList(self,books,name)

if k>=0 and books[k].count>0:

na=input("Enter your name: ")

s=masteraccess.getPosnInList(self,se,na)

t=masteraccess.getPosnInList(self,te,na)

if s>=0 :

if se[s].issues < se[s].max:

se[s].issued.append(books[k].name)

se[s].issues=se[s].issues+1

books[k].count=books[k].count-1

else:

raise issuesRestrictedError

elif t>=0:

if te[t].issues < te[t].max:

te[t].issued.append(books[t].name)

te[t].issues=te[t].issues+1

books[k].count=books[k].count-1

else:

raise issuesRestrictedError

else:

print("Invalid username...")

else:

print("Book not available in stock.......")

except KeyError:

print("An error occured....The book may not exist...")

except issuesRestrictedError:

issuesRestrictedError.display\_error(self)