**Exercise1:** Create Person and Account Class as shown below in class diagram. Ensure minimum balance of INR 500 in a bank account is available.



**Figure 14: Association of person with account class**

* Create Account for smith with initial balance as INR 2000 and for Kathy with initial balance as 3000.(accNum should be auto generated).
* Deposit 2000 INR to smith account.
* Withdraw 2000 INR from Kathy account.
* Display updated balances in both the account.
* Extend the functionality through Inheritance and polymorphism. Inherit two classes Savings Account and Current Account from account class. And Implement the following in the respective classes.
* **Savings Account**
* Add a variable called minimum Balance and assign final modifier.
* Override method called withdraw (This method should check for minimum balance and allow withdraw to happen)
* **Current Account**
* Add a variable called overdraft Limit
* Override method called withdraw (checks whether overdraft limit is reached and returns a Boolean value accordingly)

Code-

**package** oopsassignment;

**class** Person {

String name;

**float** age;

**public** Person(String name, **float** age) {

**super**();

**this**.name = name;

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **float** getAge() {

**return** age;

}

**public** **void** setAge(**float** age) {

**this**.age = age;

}

@Override

**public** String toString() {

**return** "Person [name=" + name + ", age=" + age + "]";

}

}

**class** Account {

**long** accNum;

**double** balance;

Person accHolder;

**double** minimumBalance;

**public** Account(**long** accNum, **double** balance, Person accHolder) {

**super**();

**this**.minimumBalance = 500;

**if** (balance >= minimumBalance) {

**this**.accNum = (**long**) *getUniqeNumber*();

**this**.balance = balance;

**this**.accHolder = accHolder;

} **else**

System.***out***.println("Dear customer the minimum balance must be 500 to open an account as per bank policy ");

}

**public** **static** **double** getUniqeNumber() {

**double** acc = 65461149;

**for** (**int** i = 1; i <= 4; i++) {

acc = acc + (**int**) (Math.*random*() \* 10);

}

**return** acc;

}

**public** **long** getAccNum() {

**return** accNum;

}

**public** **void** setAccNum(**long** accNum) {

**this**.accNum = accNum;

}

**public** Person getAccHolder() {

**return** accHolder;

}

**public** **void** setAccHolder(Person accHolder) {

**this**.accHolder = accHolder;

}

**public** **double** getMinimumbalance() {

**return** minimumBalance;

}

**public** **void** setBalance(**double** balance) {

**this**.balance = balance;

}

**public** **void** deposit(**double** desposit) {

balance += desposit;

System.***out***.println(

"Balance after desposit of " + desposit + " in " + accHolder + "'s account is :" + getBalance());

}

**public** **void** withdraw(**double** withdraw) {

**if** (balance >= withdraw) {

balance -= withdraw;

System.***out***.println(

"Balance after withdraw of " + withdraw + " in " + accHolder + "'s account is :" + getBalance());

} **else**

System.***out***.println("Insufficient Amount for withdraw of " + withdraw);

}

**public** **double** getBalance() {

**return** balance;

}

@Override

**public** String toString() {

**return** "Account [accNum=" + accNum + ", balance=" + balance + ", accHolder=" + accHolder + "]";

}

}

**class** SavingsAccount **extends** Account {

**final** **double** minimumBalance = 500;

**public** SavingsAccount(**long** accNum, **double** balance, Person accHolder) {

**super**(accNum, balance, accHolder);

}

**public** **double** getMinimumBalance() {

**return** minimumBalance;

}

@Override

**public** **void** withdraw(**double** withdraw) {

**if** (balance >= withdraw && balance >= minimumBalance) {

balance -= withdraw;

System.***out***.println("Savings Account Balance after withdraw of " + withdraw + " is :" + getBalance());

} **else**

System.***out***.println("Insufficient Amount for withdraw of " + withdraw);

}

}

**class** CurrentAccount **extends** Account {

**double** overdraftLimit = 800;

**public** CurrentAccount(**long** accNum, **double** balance, Person accHolder) {

**super**(accNum, balance, accHolder);

}

**public** **double** getOverdraftLimit() {

**return** overdraftLimit;

}

**public** **void** setOverdraftLimit(**double** overdraftLimit) {

**this**.overdraftLimit = overdraftLimit;

}

**public** **void** withdraw(**double** withdraw) {

**if** (withdraw >= overdraftLimit)

System.***out***.println("Overdraft limit reached of this Current Account");

**else**

System.***out***.println("Overdraft limit not reached of this Current Account");

}

}

**public** **class** Banking {

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

Person p1 = **new** Person("smith", 20); // p1-->smith

Person p2 = **new** Person("Kathy", 21); // p2-->Kathy

Account a1 = **new** Account(0, 2000, p1); // smith account created

Account a2 = **new** Account(0, 3000, p2); // Kathy account created

Account a3 = **new** Account(0, 200, p2); // Kathy another not created as it's minimumbalance is below 500

System.***out***.println();

System.***out***.println("Balance of Smith :" + a1.getBalance());

System.***out***.println("Balance of Kathy :" + a2.getBalance());

a1.deposit(2000);

a2.withdraw(2000);

System.***out***.println("Balance of Smith :" + a1.getBalance());

System.***out***.println("Balance of Kathy :" + a2.getBalance());

System.***out***.println(a1);

System.***out***.println(a2);

System.***out***.println();

Person p3 = **new** Person("max", 24);

SavingsAccount s = **new** SavingsAccount(514541, 1000, p3);

System.***out***.println("Balance of SavingsAccount of " + p3 + " :" + s.getBalance());

s.withdraw(200);

s.withdraw(500);

s.withdraw(400);

System.***out***.println(s.toString());

System.***out***.println();

Person p4 = **new** Person("alex", 22);

CurrentAccount z = **new** CurrentAccount(5626352, 2000, p4);

System.***out***.println("Balance of CurrentAccount of " + p4 + " :" + z.getBalance());

z.withdraw(1000);

System.***out***.println(z.toString());

}

}

**Exercise 2:** Using an inheritance hierarchy, design a Java program to model items at a library (books, journal articles, videos and CDs.) Have an abstract superclass called Item and include common information that the library must have for every item (such as unique identification number, title, and number of copies). No actual objects of type Item will be created - each actual item will be an object of a (non-abstract) subclass. Place item-type-specific behavior in subclasses (such as a video's year of release, a CD's musical genre, or a book's author).  
More in detail:

1. Implement an abstract superclass called Item and define all common operations on this class (constructors, getters, setters, equals, toString, print, checkIn, checkOut, addItem, etc). Have private data for: identification number, title, and number of copies.

2. Implement an abstract subclass of Item named WrittenItem and define all common operations on this class. Added private data for author.

3. Implement 2 subclasses of WrittenItem: Book and JournalPaper.

3.1. Class Book: no new private data. When needed, override/overload methods from the superclass.  
3.2. Class JournalPaper: added private data for year published. When needed, override/overload methods from the superclass.

4. Implement another abstract subclass of Item named MediaItem and define all common operations on this class. Added private data for runtime (integer).

5. Implement 2 subclasses of MediaItem: Video and CD.

5.1. Class Video: added private data for director, genre and year released. When needed, override/overload methods from the superclass.  
5.2. Class CD: added private data for artist and genre. When needed, override/overload methods from the superclass.

Write the definitions of these classes and a client program (your choice!) showing them in use.

Code-

**package** oopsassignment;

**import** java.util.Objects;

**class** Library {

Item books;

Item journalArticles;

Item videos;

Item CDs;

**long** uniqueID;

String title;

**int** noOfCopies;

}

**abstract** **class** Item {

**private** **long** uniqueID;

**private** String title;

**private** **int** noOfCopies;

**public** Item() {

uniqueID = 0;

title = " ";

noOfCopies = 0;

}

**public** Item(**long** uniqueID, String title, **int** noOfCopies) {

**super**();

**this**.uniqueID = uniqueID;

**this**.title = title;

**this**.noOfCopies = noOfCopies;

}

**public** **long** getUniqueID() {

**return** uniqueID;

}

**public** **void** setUniqueID(**long** uniqueID) {

**this**.uniqueID = uniqueID;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** **int** getNoOfCopies() {

**return** noOfCopies;

}

**public** **void** setNoOfCopies(**int** noOfCopies) {

**this**.noOfCopies = noOfCopies;

}

@Override

**public** **int** hashCode() {

**return** Objects.*hash*(noOfCopies, title, uniqueID);

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

Item other = (Item) obj;

**return** noOfCopies == other.noOfCopies && Objects.*equals*(title, other.title) && uniqueID == other.uniqueID;

}

@Override

**public** String toString() {

**return** "Item [uniqueID=" + uniqueID + ", title=" + title + ", noOfCopies=" + noOfCopies + "]";

}

**public** **void** print() {

System.***out***.println("Title :" + title);

System.***out***.println("Unique ID :" + uniqueID);

System.***out***.println("Number of Copies :" + noOfCopies);

}

**public** **void** checkIn() {

noOfCopies--;

}

**public** **void** checkOut() {

noOfCopies++;

}

**public** **void** addItem(**long** uniqueID, String title, **int** noOfCopies) {

setUniqueID(uniqueID);

setTitle(title);

setNoOfCopies(noOfCopies);

noOfCopies++;

}

}

**class** WrittenItem **extends** Item {

**private** String author;

**public** WrittenItem() {

**super**();

author = " ";

}

**public** WrittenItem(**long** uniqueID, String title, **int** noOfCopies, String author) {

**super**(uniqueID, title, noOfCopies);

**this**.author = author;

}

**public** String getAuthor() {

**return** author;

}

**public** **void** setAuthor(String author) {

**this**.author = author;

}

@Override

**public** **int** hashCode() {

**final** **int** prime = 31;

**int** result = **super**.hashCode();

result = prime \* result + Objects.*hash*(author);

**return** result;

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (!**super**.equals(obj))

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

WrittenItem other = (WrittenItem) obj;

**return** Objects.*equals*(author, other.author);

}

@Override

**public** String toString() {

**return** "WrittenItem [author=" + author + "]";

}

@Override

**public** **void** print() {

**super**.print();

System.***out***.println("Author :" + author);

}

@Override

**public** **void** checkIn() {

**super**.checkIn();

}

@Override

**public** **void** checkOut() {

**super**.checkOut();

}

**public** **void** addItem(**long** uniqueID, String title, **int** noOfCopies, String author) {

**super**.addItem(uniqueID, title, noOfCopies);

**this**.author = author;

}

}

**class** Book **extends** WrittenItem {

**public** Book() {

**super**();

}

**public** Book(**long** uniqueID, String title, **int** noOfCopies, String author) {

**super**(uniqueID, title, noOfCopies, author);

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj)

**return** **true**;

**if** (!**super**.equals(obj))

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

**return** **true**;

}

@Override

**public** String toString() {

**return** "Book [Author=" + getAuthor() + ", UniqueID=" + getUniqueID() + ", Title=" + getTitle() + ", NoOfCopies="

+ getNoOfCopies() + ", Class=" + getClass() + ", hashCode=" + hashCode() + "]";

}

**public** **void** print() {

System.***out***.println("Displaying information about a book :");

**super**.print();

}

**public** **void** checkIn() {

**super**.checkIn();

}

**public** **void** checkOut() {

**super**.checkOut();

}

**public** **void** addItem(**long** uniqueID, String title, **int** noOfCopies, String author) {

**super**.addItem(uniqueID, author, noOfCopies, author);

noOfCopies++;

}

}

**class** JournalPaper **extends** WrittenItem {

**private** **int** yearPublished;

**public** JournalPaper() {

**super**();

}

**public** JournalPaper(**long** uniqueID, String title, **int** noOfCopies, String author, **int** yearPublished) {

**super**(uniqueID, title, noOfCopies, author);

**this**.yearPublished = yearPublished;

}

@Override

**public** String toString() {

**return** "JournalPaper [yearPublished=" + yearPublished + ", Author=" + getAuthor() + ", UniqueID="

+ getUniqueID() + ", Title=" + getTitle() + ", NoOfCopies=" + getNoOfCopies() + ", Class=" + getClass()

+ ", hashCode=" + hashCode() + "]";

}

**public** **void** print() {

**super**.print();

}

**public** **void** checkIn() {

**super**.checkIn();

}

**public** **void** checkOut() {

**super**.checkOut();

}

**public** **void** addItem(**long** uniqueID, String title, **int** noOfCopies, String author) {

**super**.addItem(uniqueID, author, noOfCopies, author);

noOfCopies++;

}

}

**abstract** **class** MediaItem **extends** Item {

**private** **int** runtime;

**public** MediaItem(**int** runtime) {

**super**();

**this**.runtime = runtime;

}

**public** **int** getRuntime() {

**return** runtime;

}

**public** **void** setRuntime(**int** runtime) {

**this**.runtime = runtime;

}

@Override

**public** String toString() {

**return** "MediaItem [runtime=" + runtime + "]";

}

}

**class** Video **extends** MediaItem {

**private** String director;

**private** String genre;

**private** **int** year;

**public** Video(**int** runtime, String director, String genre, **int** year) {

**super**(runtime);

**this**.director = director;

**this**.genre = genre;

**this**.year = year;

}

**public** String getDirector() {

**return** director;

}

**public** **void** setDirector(String director) {

**this**.director = director;

}

**public** String getGenre() {

**return** genre;

}

**public** **void** setGenre(String genre) {

**this**.genre = genre;

}

**public** **int** getYear() {

**return** year;

}

**public** **void** setYear(**int** year) {

**this**.year = year;

}

@Override

**public** String toString() {

**return** "Video [director=" + director + ", genre=" + genre + ", year=" + year + "]";

}

**public** **void** print() {

System.***out***.println("Runtime(hrs) :" + getRuntime());

System.***out***.println("Director :" + director);

System.***out***.println("Genre :" + genre);

System.***out***.println("Year of Release :" + year);

}

}

**class** CD **extends** MediaItem {

**private** String artist;

**private** String genre;

**public** CD(**int** runtime, String artist, String genre) {

**super**(runtime);

**this**.artist = artist;

**this**.genre = genre;

}

**public** String getArtist() {

**return** artist;

}

**public** **void** setArtist(String artist) {

**this**.artist = artist;

}

**public** String getGenre() {

**return** genre;

}

**public** **void** setGenre(String genre) {

**this**.genre = genre;

}

@Override

**public** String toString() {

**return** "CD [artist=" + artist + ", genre=" + genre + "]";

}

**public** **void** print() {

System.***out***.println("Runtime(mins) :" + getRuntime());

System.***out***.println("Artist :" + artist);

System.***out***.println("Genre :" + genre);

}

}

**public** **class** Client {

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

Book b = **new** Book();

Book b1 = **new** Book(6126612, "The Inheritance of Loss", 80, "Kiran Desai");

Book b2 = **new** Book(1651651, "A Suitable Boy", 100, "Vikram Seth");

Book b3 = **new** Book(7954612, "Midnight's Children", 50, "Salman Rushdie");

Book b4 = **new** Book(1321656, "The Palance of Illusions", 150, "Chitra Banerjee Divakarni");

b1.print();

b1.checkIn();

System.***out***.println("Number of Copies after checkin :" + b1.getNoOfCopies());

b1.checkOut();

System.***out***.println("Number of Copies after checkout :" + b1.getNoOfCopies());

System.***out***.println(b1.toString());

b.addItem(5946113, "A Fine Balance", 40, "Rohinton Mistry");

System.***out***.println();

JournalPaper j1 = **new** JournalPaper(564116, "International Journal of Life Sciences Scientific Research", 200,

"Poonam Verma", 2015);

JournalPaper j2 = **new** JournalPaper(554165, "Pure and Applied Mathematics Letter", 300, "HCTM Technical Campus",

2013);

System.***out***.println("Displaying information about a journal paper :");

j1.print();

j1.checkIn();

System.***out***.println("Number of Copies after checkin :" + j1.getNoOfCopies());

j1.checkOut();

System.***out***.println("Number of Copies after checkout :" + j1.getNoOfCopies());

System.***out***.println(j1.toString());

System.***out***.println();

Video v1 = **new** Video(2, "Christopher Nolar", "Sci-fic", 2010);

Video v2 = **new** Video(2, "Ridley Scott", "Sci-fic", 2015);

System.***out***.println("Displaying information about a video :");

v1.print();

v1.checkIn();

System.***out***.println("Number of Copies after checkin :" + v1.getNoOfCopies());

v1.checkOut();

System.***out***.println("Number of Copies after checkout :" + v1.getNoOfCopies());

System.***out***.println(v1.toString());

System.***out***.println();

CD c1 = **new** CD(5, "A R Rahman", "Romcom");

CD c2 = **new** CD(4, "Arjit Singh", "Darbar");

System.***out***.println("Displaying information about a cd :");

c1.print();

c1.checkIn();

System.***out***.println("Number of Copies after checkin :" + c1.getNoOfCopies());

c1.checkOut();

System.***out***.println("Number of Copies after checkout :" + c1.getNoOfCopies());

System.***out***.println(c1.toString());

}

}