Candidate Id: 2380155

Name : Bharath Magesh

Assignment No : 03

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.

Answer:

Item.java

```
package LibrarySystem;


abstract class Item {
   private int id;
   private String title;
   private int numOfCopies;


   public Item(int id, String title, int numOfCopies) {
this.id = id;
      this.title = title;
      this.numOfCopies = numOfCopies;
   }
```

```
    public int getId() {
        return id;
    }

    public void setId(int id) {
this.id = id;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public int getNumOfCopies() {
        return numOfCopies;
    }

    public void setNumOfCopies(int numOfCopies) {
        this.numOfCopies = numOfCopies;
    }

    public void checkOut() {
        if (numOfCopies > 0) {
            numOfCopies--;
```

```
            System.out.println(title + " checked out.");
        } else {
            System.out.println(title + " is not available.");
        }
    }

    public void checkIn() {
        numOfCopies++;
        System.out.println(title + " checked in.");
    }

    public void addItem(int count) {
        numOfCopies += count;
        System.out.println(count + " copies added.");
    }

    @Override
    public String toString() {
        return "ID: " + id + ", Title: " + title + ", Copies: " + numOfCopies;
    }

    @Override
    public boolean equals(Object obj) {
        if (obj instanceof Item) {
            Item other = (Item) obj;
return this.id == other.id;
        }
        return false;
```

```
  }

  public abstract void print();
}
```

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

Answer:

WrittenItem.java

```
package LibrarySystem;

abstract class WrittenItem extends Item {
  private String author;

  public WrittenItem(int id, String title, int numOfCopies, String author) {
    super(id, title, numOfCopies);
    this.author = author;
  }

  public String getAuthor() {
    return author;
  }

  public void setAuthor(String author) {
    this.author = author;
  }

  @Override
  public String toString() {
```

```
    return super.toString() + ", Author: " + 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.

Answer:

Book.java

```
package LibrarySystem;

class Book extends WrittenItem {
  public Book(int id, String title, int numOfCopies, String author) {
    super(id, title, numOfCopies, author);
  }

  @Override
  public void print() {
    System.out.println("Book Details: " + this);
  }
}
```

3.2. Class JournalPaper: added private data for year published. When needed, override/overload methods from the superclass.

Answer:

JournalPaper.java

```
package LibrarySystem;
```

```
class JournalPaper extends WrittenItem {
    private int yearPublished;

    public JournalPaper(int id, String title, int numOfCopies, String author, int
yearPublished) {
        super(id, title, numOfCopies, author);
        this.yearPublished = yearPublished;
    }

    public int getYearPublished() {
        return yearPublished;
    }

    public void setYearPublished(int yearPublished) {
        this.yearPublished = yearPublished;
    }

    @Override
    public String toString() {
        return super.toString() + ", Year Published: " + yearPublished;
    }

    @Override
    public void print() {
        System.out.println("Journal Paper Details: " + this);
    }
}
```

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

Answer:

MediaItem.java

```
package LibrarySystem;


abstract class MediaItem extends Item {
   private int runtime;


   public MediaItem(int id, String title, int numOfCopies, int runtime) {
      super(id, title, numOfCopies);
      this.runtime = runtime;
   }


   public int getRuntime() {
      return runtime;
   }


   public void setRuntime(int runtime) {
      this.runtime = runtime;
   }


   @Override
   public String toString() {
      return super.toString() + ", Runtime: " + runtime + " mins";
   }
}
```

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.

Answer:

Video.java

```
package LibrarySystem;


class Video extends MediaItem {
   private String director;
   private String genre;
   private int yearReleased;


   public Video(int id, String title, int numOfCopies, int runtime, String director, String genre, int yearReleased) {
      super(id, title, numOfCopies, runtime);
      this.director = director;
      this.genre = genre;
      this.yearReleased = yearReleased;
   }


   public String getDirector() {
      return director;
   }


   public String getGenre() {
      return genre;
   }


   public int getYearReleased() {
```

```
    return yearReleased;
  }

  @Override
  public String toString() {
    return super.toString() + ", Director: " + director + ", Genre: " + genre + ", Year Released: " + yearReleased;
  }

  @Override
  public void print() {
    System.out.println("Video Details: " + this);
  }
}
```

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

Answer:

CD.java

```
package LibrarySystem;

class CD extends MediaItem {
  private String artist;
  private String genre;

  public CD(int id, String title, int numOfCopies, int runtime, String artist, String genre) {
    super(id, title, numOfCopies, runtime);
    this.artist = artist;
    this.genre = genre;
```

```
    }

    public String getArtist() {
        return artist;
    }

    public String getGenre() {
        return genre;
    }

    @Override
    public String toString() {
        return super.toString() + ", Artist: " + artist + ", Genre: " + genre;
    }

    @Override
    public void print() {
        System.out.println("CD Details: " + this);
    }
}
```

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

Answer:

LibrarySystem.java

```
package LibrarySystem;

public class LibrarySystem {
```

```
    public static void main(String[] args) {
        Book book = new Book(101, "Java Programming", 5, "James Gosling");
        JournalPaper journal = new JournalPaper(102, "AI Research", 2, "John Doe", 2023);
        Video video = new Video(103, "Inception", 3, 148, "Christopher Nolan", "Sci-Fi",
2010);
        CD cd = new CD(104, "Thriller", 4, 42, "Michael Jackson", "Pop");

        book.print();
        book.checkOut();
        book.checkIn();

        journal.print();
        video.print();
        cd.print();
    }
}
```

OUTPUT:

Book Details: ID: 101, Title: Java Programming, Copies: 5, Author: James Gosling

Java Programming checked out.

Java Programming checked in.

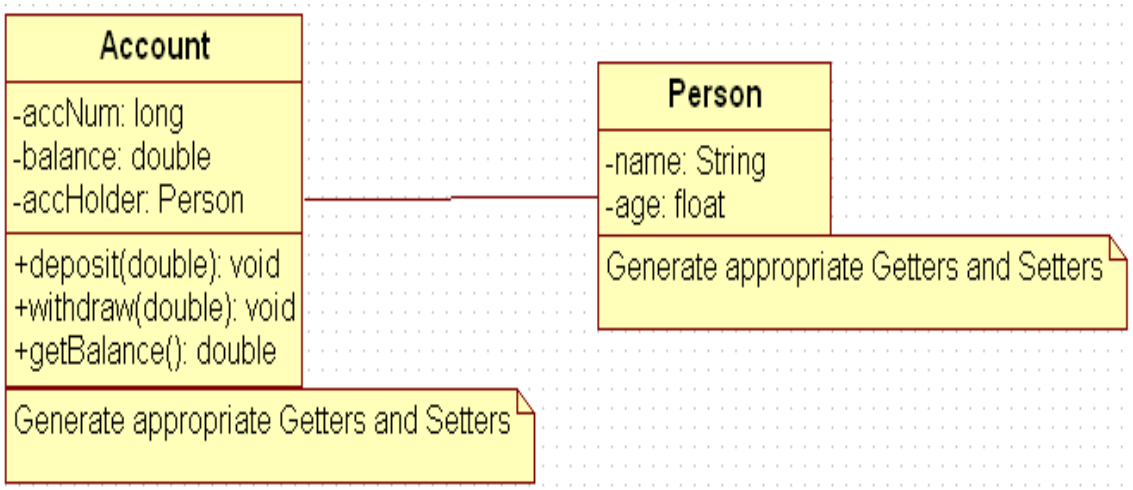
Journal Paper Details: ID: 102, Title: AI Research, Copies: 2, Author: John Doe, Year Published: 2023

Video Details: ID: 103, Title: Inception, Copies: 3, Runtime: 148 mins, Director: Christopher Nolan, Genre: Sci-Fi, Year Released: 2010

CD Details: ID: 104, Title: Thriller, Copies: 4, Runtime: 42 mins, Artist: Michael Jackson, Genre: Pop

Exercise 1:

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



a) Create Account for smith with initial balance as INR 2000 and for Kathy with initial balance as 3000.(accNum should be auto generated).

b) Deposit 2000 INR to smith account.

c) Withdraw 2000 INR from Kathy account.

d) Display updated balances in both the account.

e) 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

a. Add a variable called minimum Balance and assign final modifier.

b. Override method called withdraw (This method should check for minimum balance and allow withdraw to happen)

- Current Account

a. Add a variable called overdraft Limit

b. Override method called withdraw (checks whether overdraft limit is reached and print a Boolean value using printing statement)

Answer:

Main.java

```
package Bank;


class Person {
   private String name;
   private float age;

   // Constructor
   public Person(String name, float age) {
this.name = name;
      this.age = age;
   }

   // Getters and Setters
   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;
    }
}

class Account {
    private static long accNumCounter = 1000;
    private long accNum;
    private double balance;
    private Person accHolder;

    // Constructor
    public Account(Person accHolder, double initialBalance) {
        this.accNum = accNumCounter++;
        this.balance = initialBalance;
        this.accHolder = accHolder;
    }

    // Getters
    public long getAccNum() {
        return accNum;
    }

    public double getBalance() {
        return balance;
    }

    public Person getAccHolder() {
        return accHolder;
```

```
    }

    // Deposit method
    public void deposit(double amount) {
        balance += amount;
    }

    // Withdraw method
    public void withdraw(double amount) {
        if (balance - amount >= 500) {
            balance -= amount;
        } else {
            System.out.println("Insufficient balance! Minimum balance of INR 500 must be maintained.");
        }
    }
}

// Savings Account
class SavingsAccount extends Account {
    private final double minimumBalance = 500;

    public SavingsAccount(Person accHolder, double initialBalance) {
        super(accHolder, initialBalance);
    }

    @Override
    public void withdraw(double amount) {
```

```
        if (getBalance() - amount >= minimumBalance) {
            super.withdraw(amount);
        } else {
            System.out.println("Withdrawal denied! Minimum balance of INR " + minimumBalance + " must be maintained.");
        }
    }
}

// Current Account
class CurrentAccount extends Account {
    private double overdraftLimit = 1000;

    public CurrentAccount(Person accHolder, double initialBalance) {
        super(accHolder, initialBalance);
    }

    @Override
    public void withdraw(double amount) {
        if (getBalance() - amount >= -overdraftLimit) {
            super.withdraw(amount);
        } else {
            System.out.println("Withdrawal denied! Overdraft limit of INR " + overdraftLimit + " reached.");
        }
    }
}

public class Main {
```

```
    public static void main(String[] args) {
        // Creating persons
        Person smith = new Person("Smith", 30);
        Person kathy = new Person("Kathy", 25);

        // Creating accounts
        SavingsAccount smithAccount = new SavingsAccount(smith, 2000);
        CurrentAccount kathyAccount = new CurrentAccount(kathy, 3000);

        // Depositing to Smith's account
        smithAccount.deposit(2000);

        // Withdrawing from Kathy's account
        kathyAccount.withdraw(2000);

        // Display updated balances
        System.out.println("Smith's account balance: INR " + smithAccount.getBalance());
        System.out.println("Kathy's account balance: INR " + kathyAccount.getBalance());
    }
}
```

OUTPUT:

Smith's account balance: INR 4000.0

Kathy's account balance: INR 1000.0