|  |  |
| --- | --- |
| **Name** | Shubhan Singh |
| **UID no.** | 2022300118 |
| **Experiment No.** | 2 |

|  |  |
| --- | --- |
| **PROBLEM STATEMENT :** | *The "User" class represents a user on a ‘buy and stream’ movie platform with attributes : name, age, account balance.*  *The “Movie” class represents a movie on the platform with attributes: Movie Title, AgeRestriction, Cost of the movie.*  *The User class should have a method to check whether he can watch a movie based on his age and also account balance.*  *The User class should also have a method to WatchMovie where he has to pay the cost for the Movie to watch it.*  *The Movie class should have methods to get Cost and Age restriction.*  *The main method should create objects of the "User" and "Movie" classes and demonstrate the use of their methods.* |
| **THEORY:** | ARRAYLISTS in java:  In Java, an ArrayList is a dynamic array that can grow or shrink in size as needed. It is a part of the Java Collections Framework and provides a more flexible way to store and manipulate data than traditional arrays. ArrayLists can only store objects, and not elements of primitive data types, and they can be accessed using an index-based system.  In addition to providing dynamic resizing, ArrayLists in Java also come with a range of built-in methods for manipulating and accessing the elements of the list. Some of the most commonly used methods include the **add()** method, which adds an element to the end of the list, the **get()** method, which returns the element at a specified index, and the **size()** method, which returns the current size of the list. Other methods include **remove()**, which removes an element at a specified index, **clear()**, which removes all elements from the list, and **indexOf()**, which returns the index of the first occurrence of a specified element in the list. ArrayLists can also be sorted using the **sort()** method, which uses a natural ordering or a specified comparator to sort the elements in the list. With these methods and more, ArrayLists in Java provide a powerful tool for storing and manipulating collections of data.  “this” keyword in Java:  In Java, the "this" keyword is a reference variable that refers to the current object. It is used within a class to refer to its own instance variables and methods. When a method or constructor is called within an object, the "this" keyword is used to distinguish between local variables or parameters and instance variables with the same name. For example, if a class has an instance variable called "name" and a method parameter also called "name", the "this" keyword can be used to refer to the instance variable and avoid ambiguity. Additionally, the "this" keyword can be used to call other constructors within the same class or to return the current object from a method. |
| **PROGRAM:** | import java.util.\*;//for scanner class and arraylist class Movie{  Scanner sc= new Scanner(System.in);  String Title;  int Age\_res;  float Cost;  Movie(){}  Movie(String *Title*){  this.Title=*Title*;  }//title of movie is passed through a constructor  void getData(){//method to get data from user  System.out.println("Enter the cost of the movie");  Cost=sc.nextFloat();  System.out.println("Enter the Age restriction of the movie");  Age\_res= sc.nextInt();  } } class User{  Movie M= new Movie();  int age;  String name;  float acc\_balance;  User(Movie *m*,int *age*,String *name*,float *bal*){//Method to initialise all relevant variables for the object  M=*m*;  this.age=*age*;  this.name=*name*;  acc\_balance=*bal*;  }  boolean Can\_Watch = age>=M.Age\_res;//Is true if user age is >= age restriction  void can\_watch\_movie(){//Tells whether user can watch movie based on his age  if(Can\_Watch){  System.out.println("You can watch this movie");  }  else{  System.out.println("You cannot watch this movie");  }  }  void Watch(){//deducts cost of movie from balance or tells that balance is inadequate  if(Can\_Watch && (acc\_balance>=M.Cost)) {  acc\_balance -= M.Cost;  }  else{  System.out.println("You cannot watch this movie or balance is insufficient");  }  }  float printbal(){  System.out.println("The updated balance is: "+ acc\_balance);  return acc\_balance;  }//prints balance and returns updated balance to update the balance variable in main } public class Movie\_watch {  public static void main(String[] *args*) {  int usr\_age;  String usr\_name;  float usr\_bal;  Scanner sc= new Scanner (System.in);  String Temp\_Mov\_name;//Temporary variables to be used later  String Temp\_Mov\_name\_user;  ArrayList<Movie>Moviearr = new ArrayList<>();//Dynamic array defined using arraylist,  // as we do not know the number of movies that would be entered  System.out.println("Type the name, age and initial account balance of user");  usr\_name=sc.nextLine();  usr\_age=sc.nextInt();  usr\_bal=sc.nextFloat();  System.out.println("Type 0 to exit the admin interface(or 1 to remain in it)");  while(sc.nextInt()!=0){  sc.nextLine();//To clear input buffer, as nextint reads the integer but leaves the \n behind  System.out.println("Enter name of movie");  Temp\_Mov\_name= sc.nextLine();  Temp\_Mov\_name=Temp\_Mov\_name.toLowerCase();  Movie Tempmov= new Movie(Temp\_Mov\_name);//creating a temporary movie object to add at end of arraylist  Tempmov.getData();  Moviearr.add(Tempmov);//adding the object at the end of the arraylist  System.out.println("Type 0 to exit the admin interface(or 1 to remain in it)");  }  System.out.println("You are now in User interface, press 0 to exit it, 1 to remain");  while(sc.nextInt()!=0) {  sc.nextLine();//To clear \n from buffer  int req\_index=-1;  System.out.println("Enter name of movie");  Temp\_Mov\_name\_user = sc.nextLine();  Temp\_Mov\_name\_user=Temp\_Mov\_name\_user.toLowerCase();  for(int i=0;i<Moviearr.size();i++){//To check for required movie and fetch it from the library  if(Moviearr.get(i).Title.equals(Temp\_Mov\_name\_user)){  req\_index=i;  break;  }  }  if(req\_index==-1){  System.out.println("Movie not found");  }  else{//Driver Code  User usr=new User(Moviearr.get(req\_index),usr\_age,usr\_name,usr\_bal);  usr.can\_watch\_movie();  System.out.println("Do you want to watch this movie? (enter 1 for yes, 0 for no)");  if(sc.nextInt()==1){  usr.Watch();  usr\_bal=usr.printbal();  }  }  System.out.println("Enter 0 to exit program, 1 to check for another movie");  }  } }  Link to program for better readability and copying(it doesn’t get copied properly from the pdf):  <https://github.com/IAmAGoodBoy04/Java_PSOOP/blob/master/Week%202/src/Movie_watch.java> |
| **RESULT:** | |