|  |  |
| --- | --- |
| **Name** | Shubhan Singh |
| **UID no.** | 2022300118 |
| **Experiment No.** | 5-A |

|  |  |
| --- | --- |
| **PROBLEM STATEMENT :** | *Define****class Production****that has attributes String title, String director, String writer. Production class has 3 argument constructor that sets the values. It also has getter and setter methods and Overridden toString() of object class  to display details of class.*    ***class Play****is a sub class of Production with getter and setter methods and has an attribute****int performances that is incremented every time a play happens****.*  *Add Overridden toString() of object class  to display details of class*    ***class Musical****is a Play with songs. Musical object has all attributes of Play as well as String composer and String lyricist along with getter and setter methods. Override toString display all attributes of Musical object*    *In main create 3 objects of Play and 2 objects of Musical. Every time an object of Play or Musical is created,****performances****get incremented. Also add the number of seats booked for each play or musical.*  *Find the total box office collection, provided cost of 1 seat for Play is Rs 500(can be variable) and cost of 1 seat for Musical is Rs 800(can be variable)*    *Display total No. of performances as 5 and display the box office collection.*    *Create****class Tester****with main method* |
| **THEORY:** | **Inheritance in Java:**  Inheritance is a fundamental concept in object-oriented programming (OOP) languages like Java. It allows one class to inherit the properties (fields) and behaviors (methods) of another class, establishing a parent-child relationship between the two. The class being inherited from is called the superclass or parent class, while the class inheriting is known as the subclass or child class.  In Java, inheritance is achieved using the **extends** keyword. By extending a class, the subclass gains access to all public and protected members of the superclass, such as fields, methods, and nested classes. It promotes code reuse and enables the creation of more specialized classes based on existing ones.  Some of the key aspects of inheritance in Java are:   1. **Inheriting Fields**: Subclasses inherit the fields of the superclass, except for private fields that are not directly accessible. However, they can be accessed through public or protected methods defined in the superclass. The subclass can also add new fields or override the inherited fields. 2. **Inheriting Methods**: Subclasses can inherit and use the methods defined in the superclass. They can also override these methods to provide their own implementation. Method overriding allows the subclass to redefine the behavior of a method inherited from the superclass. It involves using the **@Override** annotation to ensure that the method is correctly overridden. 3. **Constructors and Inheritance**: Constructors are not inherited by subclasses. However, a subclass constructor can call the constructor of the superclass using the **super()** keyword as its first statement. This allows the initialization of inherited fields and the execution of superclass constructors before the subclass constructor's specific code. 4. **Access Modifiers**: Inheritance is affected by access modifiers (public, protected, private, and default). The subclass can access public and protected members of the superclass directly. Private members are not accessible. The default access modifier allows access within the same package, but subclasses in different packages can only access default members if they are in the same package as the superclass. 5. **Single Inheritance**: Java supports single inheritance, meaning a class can only inherit from a single superclass. This restriction prevents ambiguity and simplifies the language design. However, interfaces can be implemented multiple times by a single class. 6. **The Object Class**: In Java, all classes implicitly inherit from the **Object** class, which is at the root of the class hierarchy. Therefore, even if a class does not explicitly specify a superclass, it still has the methods and fields inherited from **Object**. This includes methods like **toString()**, **equals()**, and **hashCode()**, which can be overridden as needed. 7. **The super Keyword**: The **super** keyword is used in a subclass to access members of the superclass that have been hidden or overridden. It is often used to invoke superclass constructors, access superclass methods, or access and modify hidden superclass fields. 8. **Inheritance Hierarchies**: Inheritance can create hierarchical relationships where subclasses can further act as superclasses for other classes. This forms an inheritance hierarchy or a class hierarchy, allowing for multiple levels of inheritance and specialization. |
| **PROGRAM:** | class Production{  private String title,writer,director;  Production(){};  public Production(String *title*, String *director*, String *writer*){  this.title=*title*;  this.director=*director*;  this.writer=*writer*;  }  public void setTitle(String *title*) {  this.title = *title*;  }  public void setDirector(String *director*) {  this.director = *director*;  }  public void setWriter(String *writer*) {  this.writer = *writer*;  }  public String getTitle() {  return title;  }  public String getDirector() {  return director;  }  public String getWriter() {  return writer;  }  public String tostring() {  return "The production "+title + " is directed by " + director + " and written by " + writer + "\n";  }  public int no\_of\_seats;  public static int **BOC**=0;  public static int getBOC(){  return **BOC**;  } } class Play extends Production{  private static int **performances**=0;  public Play(){}  public Play(String *title*, String *director*, String *writer*,int *seats*){  super(*title*, *director*, *writer*);  no\_of\_seats=*seats*;  **performances**++;  }  public void incr\_boc\_play(){  **BOC**+=500\*no\_of\_seats;  }  *@Override* public String tostring() {  return "The play "+getTitle() + " is directed by " + getDirector() + " and written by " + getWriter()+ ".\n" +  "With "+no\_of\_seats+" seats sold and a Box office collection of "+this.no\_of\_seats\*500 + " rupees.\n";  }  public static void incr\_performances(){  **performances**++;  }  static int getPerformances(){  return **performances**;  }  } class Musical extends Play{  private String composer,lyricist;  public Musical(String *title*, String *director*, String *writer*,String *composer*, String *lyricist*,int *seats*){  super(*title*, *director*, *writer*,*seats*);  this.composer=*composer*;  this.lyricist=*lyricist*;  }   public void incr\_boc\_musical(){  **BOC**+=800\*no\_of\_seats;  }  public void setComposer(String *composer*) {  this.composer = *composer*;  }  public void setLyricist(String *lyricist*) {  this.lyricist = *lyricist*;  }  public String getComposer() {  return composer;  }  public String getLyricist() {  return lyricist;  }  *@Override* public String tostring(){  return "The musical "+getTitle() + " is directed by " + getDirector() + " and written by " + getWriter()+ " with music by "  +getComposer() + " and lyrics by "+ getLyricist()+".\nWith "+no\_of\_seats+" seats sold and a Box office collection of "+this.no\_of\_seats\*800 + " rupees.\n";  } } public class Tester {  public static void main(String[] *args*) {  Musical[] musicalarr=new Musical[2];  Play[] playarr=new Play[3];  playarr[0]=new Play("Macbeth","Dir1","Shakespeare",27);  playarr[1]=new Play("Andha Yug","Dir2","Dharamveer Bharati",22);  playarr[2]=new Play("Shakuntala","Dir3","Kalidasa",31);  musicalarr[0]=new Musical("Yayati","Dir4","Girish Karnad","composer1","lyricist1", 29);  musicalarr[1]=new Musical("Natsamrat","Dir5","Girish Karnad","composer2","lyricist2", 33);  for(int i=0;i<3;i++){  playarr[i].incr\_boc\_play();  }  for(int i=0;i<2;i++){  musicalarr[i].incr\_boc\_musical();  }  for(int i=0;i<3;i++){  System.out.printf(playarr[i].tostring());  }  for(int i=0;i<2;i++){  System.out.printf(musicalarr[i].tostring());  }  System.out.printf("The total Box office collection of all 5 productions is %d.\n", Production.**getBOC**());  } } |
| **RESULT:** | |