**Java Inheritance**

*Inheritance*, is one of three object oriented concepts, which helps to separate out *common data and behavior* (or member variables and methods) from multiple related classes.

it is necessary to remove duplicate code across classes. *Inheritance* greatly helps in simplifying the code, enhancing its the re-usability and maintainability.

**Example**

A picture containing diagram

Description automatically generated

To achieve *inheritance*, we need to create a *super class* with the common code and behavior from multiple classes. Then we use the extends keyword to use that common code in the *sub classes*.

class CompareEntertainmentsUsingInheritance  
{  
    public static void main(String arg[])  
    {  
        Movie julai = new Movie();  
        julai.name = "Julai";  
        julai.director = "Trivikram";  
        julai.stuntMaster = "Peter Hein";  
        julai.numberOfArtists = 57;  
        julai.releaseDate = "15-Aug-2012";  
          
        julai.collectionsFirstWeek = 215467.8;  
        julai.collectionsRestOfTheDays = 541132.5;  
          
        Drama ramayan = new Drama();  
        ramayan.name = "Ramayana";  
        ramayan.writer = "Valmiki";  
        ramayan.stageSetter = "Anjaneya";  
        ramayan.numberOfArtists = 200000;  
        ramayan.releaseDate = "17-Mar-1659 BC";  
          
        ramayan.collectionsFirstWeek = 3282937242.86;  
        ramayan.collectionsRestOfTheDays = 93488272349.51;  
          
          
        Circus jumbo = new Circus();  
        jumbo.name = "Jumbo";  
        jumbo.ringMaster = "Antony";  
        jumbo.numberOfArtists = 316;  
        jumbo.releaseDate = "16-Dec-1997";  
          
        jumbo.collectionsFirstWeek = 2123132.21;  
        jumbo.collectionsRestOfTheDays = 234936725.09;  
          
        if((jumbo.getTotalCollections() > julai.getTotalCollections()) && (jumbo.getTotalCollections() > ramayan.getTotalCollections()))  
        {  
            jumbo.print();  
        }  
        else if (julai.getTotalCollections() > ramayan.getTotalCollections())  
        {  
            julai.print();  
        }  
        else  
        {  
            ramayan.print();  
        }  
      
    }  
}  
  
class Entertainment  
{  
    String name;  
    int numberOfArtists;  
    String releaseDate;  
    double collectionsFirstWeek;  
    double collectionsRestOfTheDays;  
  
    double getTotalCollections()  
    {  
        return collectionsFirstWeek + collectionsRestOfTheDays;  
    }  
  
    void printEntertainment()  
    {  
        System.out.println( name + " got the following collections " );  
        System.out.println("First Week : " + collectionsFirstWeek);  
        System.out.println("Rest Of The Days : " + collectionsRestOfTheDays);  
        System.out.println("Total Collections : " + getTotalCollections());  
        System.out.println("Total Artists : " + numberOfArtists);  
        System.out.println("Release Date : " + releaseDate);  
    }  
  
}  
  
class Movie extends Entertainment  
{  
    String director;  
    String stuntMaster;  
  
    void print()  
    {  
        printEntertainment();  
        System.out.println("Director : " + director);  
        System.out.println("Stunt Master : " + stuntMaster);  
    }  
}  
  
class Drama extends Entertainment  
{  
    String writer;  
    String stageSetter;  
  
  
    void print()  
    {  
        printEntertainment();  
        System.out.println("Writer : " + writer);  
        System.out.println("Stage Setter : " + stageSetter);  
    }  
}  
  
class Circus extends Entertainment  
{  
    String ringMaster;  
  
    void print()  
    {  
        printEntertainment();  
        System.out.println("Ring Master : " + ringMaster);  
    }  
}

**Inheritance supports is-a relationship**

Every sub-class object is also a super-class object, but every super-class object need not be a sub-class object. e.g., Every Movie is an Entertainment, but every Entertainment need not be a Movie, similarly, every Drama is an Entertainment, but every Entertainment need not be a Drama.

public void printName(Entertainment e)  
{  
    System.out.println("Name of the entertainment is " + e.name);  
}  
  
Movie businessMan = new Movie();  
businessMan.name = "Business Man";  
businessMan.directorName = "Puri Jagannadh";  
  
Drama devdas = new Drama();  
devdas.name = "Devdas";  
  
printName(businessMan); // Calling printName method by passing the Movie object  
printName(devdas); // Calling printName method by passing the Drama object

**This wont work:**

public void printMovie(Movie m)  
{  
    System.out.println(m.directorName + " is the director for movie " + m.name);  
}  
  
Movie businessMan = new Movie();  
businessMan.name = "Business Man";  
businessMan.directorName = "Puri Jagannadh";  
  
Drama devdas = new Drama();  
devdas.name = "Devdas";  
  
printMovie(businessMan); // Calling printMovie method by passing the Movie object  
printMovie(devdas); // THIS WON'T WORK SINCE DRAMA IS NOT A MOVIE

**Comapre different entertainments:**

public static void compareEntertainments(Entertainment ent1, Entertainment ent2, Entertainment ent3)  
    {  
    if((ent1.getTotalCollections() > ent2.getTotalCollections()) && (ent1.getTotalCollections() > ent3.getTotalCollections()))  
        {  
            System.out.println(ent1.name + " has the highest collections.");  
        }  
        else if (ent2.getTotalCollections() > ent3.getTotalCollections())  
        {  
            System.out.println(ent2.name + " has the highest collections.");  
        }  
        else  
        {  
            System.out.println(ent3.name + " has the highest collections.");  
        }  
    }  
  
}

**Downcasting**

since every sub-class object is also a super-class object, we can assign a sub-class object to a super-class reference. This is also called as *downcasting*. e.g., the object of type Drama can be assigned to the reference of the type Entertainment.

Entertainment e;  
  
Movie m = new Movie();  
e = m; // LINE A  
  
Drama d = new Drama();  
e = d; // LINE B  
  
m = d; // LINE C - WON'T WORK, SINCE A DRAMA OBJECT IS NOT OF TYPE MOVIE.

**Assigning super class reference to A sub class reference**

Entertainment ent1;  
Drama d1 = new Drama();  
ent1 = d1;  
  
Drama d2 = (Drama) ent1; // LINE A

As shown here, we are first creating a reference of Entertainment called ent1, a reference of Drama called d1 and a Drama object. Since we know that ent1 is referring to an object of type Drama, we can up cast the reference as shown in LINE A and assign it to another Drama reference d2.  
Instead, if we try to up cast a reference which does not refer to the correct object type, then it throws a ClassCastException during run time.

Entertainment ent1;  
Circus c1 = new Circus();  
ent1 = c1;  
  
Drama d2 = (Drama) ent1;

Circus c2 = (Circus) ent1;

Here a ClassCastException is thrown at LINE A, since ent1 points to Circus object and hence can not be cast to a Drama. But the casting as done in LINE B will work since ent1 points to the Circus object.

**Multiple Inheritance in Java**

class TestVehiclesHierarchy  
{  
    public static void main(String arg[])  
    {  
        MountainBike mb = new MountainBike();  
        mb.numberOfWheels = 2;  
        mb.registrationNumber = "APXX WWW";  
        mb.hasHelmet = true;  
        mb.maxElevation = 3000.0;  
          
        System.out.print("Mountain Bike with registration Number " + mb.registrationNumber);  
        System.out.println(" is supported till the elevation of " + mb.maxElevation + " feet.");      
    }  
}  
  
class Vehicle  
{  
    int numberOfWheels;  
}  
  
class RegisteredVehicle extends Vehicle  
{  
    String registrationNumber;  
}  
  
class Bike extends RegisteredVehicle // LINE A  
{  
    boolean hasHelmet;  
}  
  
class MountainBike extends Bike  
{  
    double maxElevation;  
}  
  
class Car extends RegisteredVehicle  
{  
    boolean hasAC;  
}  
  
class Cycle extends Vehicle  
{  
    boolean hasBackSeat;  
}

Diamond Problem in Inheritance