DAY-38

-------

INHERITED , OVERRIDDEN , SPECILIZED METHODS.

--------------------------------------------

Refer the diagram:

INHERITED METHOD : any method which is acquired by the child class from the parent class is called as inherited method.

OVERRIDDEN METHOD : changes performed on inherited method is called as overridden methods

SPECIALIZED METHOD : mehods which are unique to child class is called as specialized methods

EXAMPLE:

---------

class Plane

{

String name;

int wheel;

void takeOff()

{

System.out.println("plane is taking off");

}

void fly()

{

System.out.println("plane is flying");

}

void land()

{

System.out.println("plane is landing");

}

}

class CargoPlane extends Plane

{

void fly()

{

System.out.println("Cargoplane is flying with the low speed");

}

void CarryCargo()

{

System.out.println("Cargoplane is carrying the goods");

}

}

class PassengerPlane extends Plane

{

void fly()

{

System.out.println("Passengerplane is flying with the medium speed");

}

void CarryPassenger()

{

System.out.println("Passengerplane is carrying the people");

}

}

class FighterPlane extends Plane

{

void fly()

{

System.out.println("Fighterlane is flying with the high speed");

}

void CarryArms()

{

System.out.println("Fighterplane is carrying the weapons");

}

}

class Demo

{

public static void main(String[] args)

{

CargoPlane cp =new CargoPlane();

PassengerPlane pp = new PassengerPlane();

FighterPlane fp = new FighterPlane();

cp.takeOff();

cp.fly();

cp.land();

cp.CarryCargo();

pp.takeOff();

pp.fly();

pp.land();

pp.CarryPassenger();

}

}

OUTPUT:

---------

plane is taking off

Cargoplane is flying with the low speed

plane is landing

Cargoplane is carrying the goods

plane is taking off

Passengerplane is flying with the medium speed

plane is landing

Passengerplane is carrying the people

METHOD OVERRIDING

--------------------------------

class Parent

{

void marry()

{

s.o.p("marry @ age of 26"); // overriden method w.r.t parent class

}

}

class Child extends Parent

{

void marry()

{

s.o.p("marry @ age of 30"); // overriding method w.r.t child class }

}

The method which is inherited by the child class from the parent class and if the child class modifies or implements the inherited methods such

process is called as METHOD OVERRIDING.

EXAMPLE:

--------

class Vehicle

{

void Move()

{

System.out.println("vehicle is moving"); // overriden method w.r.t parent class

}

}

class Bike extends Vehicle

{

void Move()

{

System.out.println("bike is moving medium speed"); // overriding method w.r.t child class }

}

}

class Car extends Vehicle

{

void Move()

{

System.out.println("Car is moving high speed"); // overriding method w.r.t child class }

}

}

class Truck extends Vehicle

{

void Move()

{

System.out.println("Truck is moving low speed"); // overriding method w.r.t child class }

}

}

class Demo1

{

public static void main(String[] args)

{

Bike b = new Bike();

Car c = new Car();

Truck t = new Truck();

b.Move(); // 1:1

c.Move(); // 1:1

t.Move(); // 1:1

}

}

OUTPUT:

-------

bike is moving medium speed

Car is moving high speed

Truck is moving low speed

PARENT REFERENCE TO CHILD OBJECT:

---------------------------------

Parent reference to child object in method overriding is all about proving the parent object refernce to the child object to achive the POLYMORPHISM.

EXAMPLE:

---------

class Vehicle

{

void Move()

{

System.out.println("vehicle is moving"); // overriden method w.r.t parent class

}

}

class Bike extends Vehicle

{

void Move()

{

System.out.println("bike is moving medium speed"); // overriding method w.r.t child class }

}

}

class Car extends Vehicle

{

void Move()

{

System.out.println("Car is moving high speed"); // overriding method w.r.t child class }

}

}

class Truck extends Vehicle

{

void Move()

{

System.out.println("Truck is moving low speed"); // overriding method w.r.t child class }

}

}

class Demo1

{

public static void main(String[] args)

{

Vehicle v = new Vehicle();

v = new Bike(); //[child object with parent refernce]

v.Move();

v = new Car();

v.Move();

v = new Truck();

v.Move(); // 1:3

}

}

OUTPUT:

-------

bike is moving medium speed

Car is moving high speed

Truck is moving low speed

POLYMORPHISM

---------------------------------

Polymorphism is one of the importent object oriented feature in java.

To achive polymorphism we need the concept of 'parent refernce to child object'.

POLY means many MORPHS means forms. An object with many forms is called as POLYMORPHISM.

NOTE: In the above program v.Move() is polymorphic statement because it is providing 3 differnt outputs ---> 1:3

POLYMORPHISM is used to achive the 'code reduction' and flexibilty of the program.

EXAMPLE:

--------

class Plane

{

String name;

int wheel;

void takeOff()

{

System.out.println("plane is taking off");

}

void fly()

{

System.out.println("plane is flying");

}

void land()

{

System.out.println("plane is landing");

}

}

class CargoPlane extends Plane

{

void takeOff()

{

System.out.println("Cargoplane is taking off");

}

void fly()

{

System.out.println("Cargoplane is flying with the low speed");

}

void CarryCargo()

{

System.out.println("Cargoplane is carrying the goods");

}

}

class PassengerPlane extends Plane

{

void fly()

{

System.out.println("Passengerplane is flying with the medium speed");

}

void CarryPassenger()

{

System.out.println("Passengerplane is carrying the people");

}

}

class FighterPlane extends Plane

{

void fly()

{

System.out.println("Fighterlane is flying with the high speed");

}

void CarryArms()

{

System.out.println("Fighterplane is carrying the weapons");

}

}

class Demo

{

public static void main(String[] args)

{

Plane ref = new Plane();

//CargoPlane cp =new CargoPlane(); //[child refernce to child object]

ref = new CargoPlane();

ref.takeOff();

ref.fly();

ref.land();

//ref.CarryCargo();

ref = new PassengerPlane();

ref.takeOff();

ref.fly();

ref.land();

//ref.CarryPassenger();

ref = new FighterPlane();

ref.takeOff();

ref.fly();

ref.land();

//ref.CarryArms();

}

}

OUTPUT:

--------

Cargoplane is taking off

Cargoplane is flying with the low speed

plane is landing

plane is taking off

Passengerplane is flying with the medium speed

plane is landing

plane is taking off

Fighterlane is flying with the high speed

plane is landing

NOTE: In the above program we have achived polymorphism by using the concept of 'parent ref to child object' but by this technique we can not access

the specialized methods.

EXAMPLE:

--------

class Plane

{

String name;

int wheel;

void takeOff()

{

System.out.println("plane is taking off");

}

void fly()

{

System.out.println("plane is flying");

}

void land()

{

System.out.println("plane is landing");

}

}

class CargoPlane extends Plane

{

void takeOff()

{

System.out.println("Cargoplane is taking off");

}

void fly()

{

System.out.println("Cargoplane is flying with the low speed");

}

void CarryCargo()

{

System.out.println("Cargoplane is carrying the goods");

}

}

class PassengerPlane extends Plane

{

void fly()

{

System.out.println("Passengerplane is flying with the medium speed");

}

void CarryPassenger()

{

System.out.println("Passengerplane is carrying the people");

}

}

class FighterPlane extends Plane

{

void fly()

{

System.out.println("Fighterlane is flying with the high speed");

}

void CarryArms()

{

System.out.println("Fighterplane is carrying the weapons");

}

}

class Airport

{

void allow(Plane ref)

{

ref.takeOff();

ref.fly();

ref.land();

}

}

class Demo

{

public static void main(String[] args)

{

CargoPlane cp =new CargoPlane(); //[child refernce to child object]

PassengerPlane pp = new PassengerPlane();

FighterPlane fp = new FighterPlane();

Airport a = new Airport();

a.allow(cp);

a.allow(pp);

a.allow(fp);

}

}

OUTPUT:

--------

Cargoplane is taking off

Cargoplane is flying with the low speed

plane is landing

plane is taking off

Passengerplane is flying with the medium speed

plane is landing

plane is taking off

Fighterlane is flying with the high speed

plane is landing

STEPS TO ACHIEVE POLYMORPHISM:

---------------------------------

1. INHERITANCE must be present.

2. classes must be related

3. METHOD OVERRIDING should be present

4. use the concept of 'PARENT REF TO CHILD OBJECT'

5. POLYMORPHISM

NOTE: To access the specialized methods we must peroform 'DOWNCASTING'.

EXAMPLE:

--------

class Plane

{

String name;

int wheel;

void takeOff()

{

System.out.println("plane is taking off");

}

void fly()

{

System.out.println("plane is flying");

}

void land()

{

System.out.println("plane is landing");

}

}

class CargoPlane extends Plane

{

void takeOff()

{

System.out.println("Cargoplane is taking off");

}

void fly()

{

System.out.println("Cargoplane is flying with the low speed");

}

void CarryCargo()

{

System.out.println("Cargoplane is carrying the goods");

}

}

class PassengerPlane extends Plane

{

void fly()

{

System.out.println("Passengerplane is flying with the medium speed");

}

void CarryPassenger()

{

System.out.println("Passengerplane is carrying the people");

}

}

class FighterPlane extends Plane

{

void fly()

{

System.out.println("Fighterlane is flying with the high speed");

}

void CarryArms()

{

System.out.println("Fighterplane is carrying the weapons");

}

}

class Demo

{

public static void main(String[] args)

{

Plane ref = new Plane();

ref =new CargoPlane(); // DOWNCASTING

//ref = new CargoPlane(); // UPCASTING

ref.fly();

ref.land();

((CargoPlane)(ref)).CarryCargo();

//ref = new PassengerPlane();

//ref = new FighterPlane();

}

}

OUTPUT:

-------

Cargoplane is flying with the low speed

plane is landing

Cargoplane is carrying the goods

UPCASTING and DOWNCASTING:

---------------------------

UPCASTING : upcasting refers to the creation of child object and assigning the parent ref to it.

advantage of upcasting is achiving the polymorphism

DOWNCASTING: it means tempory converting the parent ref to child type to access the specialized method