**13-08-24 Java Lab Codes**

Program 1:

public class Main

{

    public static void main(String[] args)

    {

        Person myObj=new Person();

        myObj.setName("John");

        System.out.println(myObj.getName());

    }

}

public class Person

{

    private String name;

    public String getName()

    {

        return name;

    }

    public void setName(String newName)

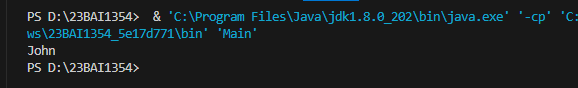
    {

        this.name=newName;

    }

}

Output:



Program 2:

class Vehicle

{

    protected String brand="Ford";

    public void honk()

    {

        System.out.println("Tuut, tuut!");

    }

}

class Car extends Vehicle{

    private String modelName="Mustang";

    public static void main(String[] args)

    {

        Car myFastCar=new Car();

        myFastCar.honk();

        System.out.println(myFastCar.brand+" "+myFastCar.modelName);

    }

}

Output:



Program 3:

class Animal

{

    public void animalSound()

    {

        System.out.println("The animal makes a sound");

    }

}

class Pig extends Animal

{

    public void animalSound()

    {

        System.out.println("The pig says: wee wee");

    }

}

class Dog extends Animal{

    public void animalSound()

    {

        System.out.println("The dog says: bow wow");

    }

}

class Main

{

    public static void main(String[] args)

    {

        Animal myAnimal=new Animal();

        Animal myPig=new Pig();

        Animal myDog=new Dog();

        myAnimal.animalSound();

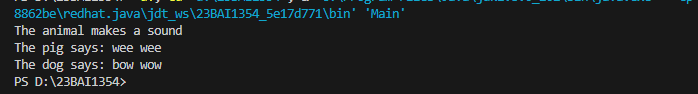
        myPig.animalSound();

        myDog.animalSound();

    }

}

Output:



Program 4:

class OuterClass

{

    int x=10;

    class InnerClass

    {

        int y=5;

    }

}

public class Main

{

    public static void main(String[] args) {

        OuterClass myOuter=new OuterClass();

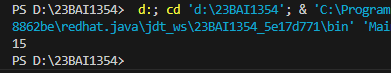
        OuterClass.InnerClass myInner= myOuter.new InnerClass();

        System.out.println(myInner.y+myOuter.x);

    }

}

Output:



Program 5:

class OuterClass

{

    int x=10;

    static class InnerClass

    {

        int y=5;

    }

}

public class Main

{

    public static void main(String[] args) {

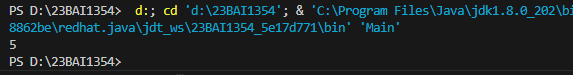
        OuterClass.InnerClass myInner= new OuterClass.InnerClass();

        System.out.println(myInner.y);

    }

}

Output:



Program 6:

class OuterClass

{

    int x=10;

    class InnerClass

    {

        public int myInnerMethod()

        {

            return x;

        }

    }

}

public class Main

{

    public static void main(String[] args) {

        OuterClass myOuter = new OuterClass();

        OuterClass.InnerClass myInner = myOuter.new InnerClass();

        System.out.println(myInner.myInnerMethod());

    }

}

Output:



Program 7:

abstract class Animal

{

    public abstract void animalSound();

    public void sleep()

    {

        System.out.println("Zzz");

    }

}

class Pig extends Animal{

    public void animalSound()

    {

        System.out.println("The pig says: wee wee");

    }

}

class Main

{

    public static void main(String[] args)

    {

        Pig myPig=new Pig();

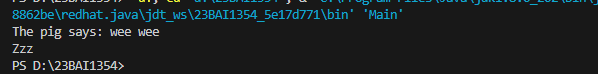
        myPig.animalSound();

        myPig.sleep();

    }

}

Output:



Program 8:

interface Animal

{

    public void AnimalSound();

    public void sleep();//interface method -> does not have a body

}

class Pig implements Animal

{

    public void AnimalSound()

    {

        System.out.println("The pig says: wee wee");

    }

    public void sleep()

    {

        System.out.println("Zzz");

    }

}

class Main

{

    public static void main(String[] args) {

        Pig myPig=new Pig();

        myPig.AnimalSound();;

        myPig.sleep();

    }

}

Output:

