**1.**

public class Person{

    private String name;

    public String getName(){

        return name;

    }

    public void setName(String newName){

        this.name= newName;}

    }

public class Encap{

    public static void main(String[] args){

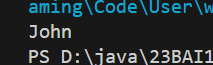
        Person myobj = new Person();

        myobj.setName("John");

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

    }

}

****

**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){

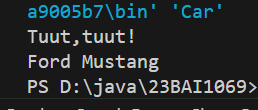
            Car myFastCar = new Car();

            myFastCar.honk();

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

        }

}

****

**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 bow");

    }

}

public class Poly {

    public static void main(String[] args) {

     Animal Sound = new Animal();

     Animal Pig = new Pig();

     Animal Dog = new Dog();

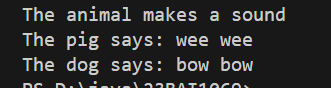
    Sound.animalSound();

    Pig.animalSound();

    Dog.animalSound();

    }

}

****

**4.**

public class Outerclass{

    int x=10;

    public class Innerclass{

        int y = 5;

    }

}

public class Nesting {

    public static void main(String[] args) {

        Outerclass outer= new Outerclass();

        Outerclass.Innerclass inner = outer.new Innerclass();

        System.out.println(outer.x+inner.y);

    }

}

****

**5.**

public class Outerclass{

    int x=10;

    static class Innerclass{

        int y = 5;

    }

}

public class Nesting {

    public static void main(String[] args) {

        Outerclass.Innerclass inner = new Outerclass.Innerclass();

        System.out.println(inner.y);

    }

}

****

**6.**

public class Access {

    public static void main(String[] args){

        Outeraccess oa = new Outeraccess();

        Outeraccess.Innerclass ia = oa.new Innerclass();

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

    }

}

****

**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 Abstractt{

    public static void main(String[] args) {

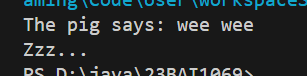
        pig mypig = new pig();

        mypig.animalSound();

        mypig.sleep();

    }

}

****

**8.**

interface Animal{

    public void animalSound();

    public void sleep();}

class pig implements Animal{

    public void animalSound(){

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

    }

    public void sleep(){

        System.out.println("Zzz...");

    }

}

    public class Interface {

    public static void main(String[] args) {

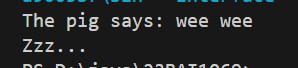
        pig myPig = new pig();

        myPig.animalSound();

        myPig.sleep();

    }

}

****