/\*Inheritance\*/

class Teacher {

String designation = "Teacher";

String collegeName = "Beginnersbook";

void does(){

System.out.println("Teaching");

}

}

public class PhysicsTeacher extends Teacher{

String mainSubject = "Physics";

public static void main(String args[]){

PhysicsTeacherobj = new PhysicsTeacher();

System.out.println(obj.collegeName);

System.out.println(obj.designation);

System.out.println(obj.mainSubject);

obj.does();

}

}

What if we define members of class “Teacher”as private? (Bonus point)

class Teacher {

private String designation = "Teacher";

private String collegeName = "Beginnersbook";

public String getDesignation() {

return designation;

}

protected void setDesignation(String designation) {

this.designation = designation;

}

protected String getCollegeName() {

returncollegeName;

}

protected void setCollegeName(String collegeName) {

this.collegeName = collegeName;

}

void does(){

System.out.println("Teaching");

}

}

public class JavaExample extends Teacher{

String mainSubject = "Physics";

public static void main(String args[]){

JavaExampleobj = new JavaExample();

/\* Note: we are not accessing the data members

\* directly we are using public getter method

\* to access the private members of parent class

\*/

System.out.println(obj.getCollegeName());

System.out.println(obj.getDesignation());

System.out.println(obj.mainSubject);

obj.does(); }}

/\*Ploymorphism\*/ //Runtime

public class Animal{

public void sound(){

System.out.println("Animal is making a sound");

}

}

public class Cat extends Animal{

@Override

public void sound(){

System.out.println("Meow");

}

public static void main(String args[]){

Animal obj = new Cat();

obj.sound();

}

}

/\*Polymorphism\*/ //Compile Time (Bonus Point)

class Overload

{

void demo (int a)

{

System.out.println ("a: " + a);

}

void demo (int a, int b)

{

System.out.println ("a and b: " + a + "," + b);

}

double demo(double a) {

System.out.println("double a: " + a);

return a\*a;

}

}

classMethodOverloading

{

public static void main (String args [])

{

Overload Obj = new Overload();

double result;

Obj .demo(10);

Obj .demo(10, 20);

result = Obj .demo(5.5);

System.out.println("O/P : " + result);

}

}

/\*Encapsulation\*/

classEncapsulationDemo{

private intssn;

private String empName;

privateintempAge;

//Getter and Setter methods

publicintgetEmpSSN(){

returnssn;

}

public String getEmpName(){

returnempName;

}

publicintgetEmpAge(){

returnempAge;

}

public void setEmpAge(intnewValue){

empAge = newValue;

}

public void setEmpName(String newValue){

empName = newValue;

}

public void setEmpSSN(intnewValue){

ssn = newValue;

}

}

public class EncapsTest{

public static void main(String args[]){

EncapsulationDemoobj = new EncapsulationDemo();

obj.setEmpName("Mario");

obj.setEmpAge(32);

obj.setEmpSSN(112233);

System.out.println("Employee Name: " + obj.getEmpName());

System.out.println("Employee SSN: " + obj.getEmpSSN());

System.out.println("Employee Age: " + obj.getEmpAge());

}

}

/\*Abstraction\*/

abstract class Animal{

//abstract method

public abstract void sound();

}

//Dog class extends Animal class

public class Dog extends Animal{

public void sound(){

System.out.println("Woof");

}

public static void main(String args[]){

Animal obj = new Dog();

obj.sound();

}

}

What if I create an object of class Animal?

Animal obj = new Animal();