class Teacher {

//fields of parent class

String designation = "Teacher";

String collegeName = "DCS UBIT";

//method of parent class

void does(){

System.out.println("Teaching");

}

}

public class PhysicsTeacher extends Teacher{

//field of child class

String mainSubject = "Physics";

public static void main(String args[]){

PhysicsTeacher obj = new PhysicsTeacher();

//accessing the fields of parent class

System.out.println(obj.collegeName);

System.out.println(obj.designation);

System.out.println(obj.mainSubject);

//accessing the method of parent class

obj.does();

}

}

Instaneof example:

class A{

}

class B extends A{

}

public class JavaExample extends B{

public static void main(String args[]) {

A obj1 = new A();

B obj2 = new B();

JavaExample obj3 = new JavaExample();

System.out.println(obj1 instanceof A);

System.out.println(obj2 instanceof A);

System.out.println(obj1 instanceof B);

System.out.println(obj3 instanceof B);

}

}

**Inheritance: Access Specifiers Example**

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() {

return collegeName;

}

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[]){

JavaExample obj = 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();

}

}

**Super keyword: Inherentance Example**

class ParentClass{

//Parent class constructor

ParentClass(){

System.out.println("Constructor of Parent");

}

}

class JavaExample extends ParentClass{

JavaExample(){

/\* It by default invokes the constructor of parent class

\* You can use super() to call the constructor of parent.

\* It should be the first statement in the child class

\* constructor, you can also call the parameterized constructor

\* of parent class by using super like this: super(10), now

\* this will invoke the parameterized constructor of int arg

\*/

System.out.println("Constructor of Child");

}

public static void main(String args[]){

//Creating the object of child class

new JavaExample();

}

}

**Inheritance: Method Overloading**

class ParentClass{

//Parent class constructor

ParentClass(){

System.out.println("Constructor of Parent");

}

void disp(){

System.out.println("Parent Method");

}

}

class JavaExample extends ParentClass{

JavaExample(){

System.out.println("Constructor of Child");

}

void disp(){

System.out.println("Child Method");

//Calling the disp() method of parent class

super.disp();

}

public static void main(String args[]){

//Creating the object of child class

JavaExample obj = new JavaExample();

obj.disp();

}

}

**Super Keyword in Java Examples**

//Parent class or Superclass or base class

class Superclass

{

int num = 100;

}

//Child class or subclass or derived class

class Subclass extends Superclass

{

/\* The same variable num is declared in the Subclass

\* which is already present in the Superclass

\*/

int num = 110;

void printNumber(){

System.out.println(num);

}

public static void main(String args[]){

Subclass obj= new Subclass();

obj.printNumber();

}

}

**Now accessing the num variable of parent class:**

class Superclass

{

int num = 100;

}

class Subclass extends Superclass

{

int num = 110;

void printNumber(){

/\* Note that instead of writing num we are

\* writing super.num in the print statement

\* this refers to the num variable of Superclass

\*/

System.out.println(super.num);

}

public static void main(String args[]){

Subclass obj= new Subclass();

obj.printNumber();

}

}

**2) Use of super keyword to invoke constructor of parent class**

class Parentclass

{

Parentclass(){

System.out.println("Constructor of parent class");

}

}

class Subclass extends Parentclass

{

Subclass(){

/\* Compile implicitly adds super() here as the

\* first statement of this constructor.

\*/

System.out.println("Constructor of child class");

}

Subclass(int num){

/\* Even though it is a parameterized constructor.

\* The compiler still adds the no-arg super() here

\*/

System.out.println("arg constructor of child class");

}

void display(){

System.out.println("Hello!");

}

public static void main(String args[]){

/\* Creating object using default constructor. This

\* will invoke child class constructor, which will

\* invoke parent class constructor

\*/

Subclass obj= new Subclass();

//Calling sub class method

obj.display();

/\* Creating second object using arg constructor

\* it will invoke arg constructor of child class which will

\* invoke no-arg constructor of parent class automatically

\*/

Subclass obj2= new Subclass(10);

obj2.display();

}

}

**Output:**

Constructor of parent class

Constructor of child class

Hello!

Constructor of parent class

arg constructor of child class

Hello!

**Parameterized super() call to invoke parameterized constructor of parent class**

class Parentclass

{

//no-arg constructor

Parentclass(){

System.out.println("no-arg constructor of parent class");

}

//arg or parameterized constructor

Parentclass(String str){

System.out.println("parameterized constructor of parent class");

}

}

class Subclass extends Parentclass

{

Subclass(){

/\* super() must be added to the first statement of constructor

\* otherwise you will get a compilation error. Another important

\* point to note is that when we explicitly use super in constructor

\* the compiler doesn't invoke the parent constructor automatically.

\*/

super("Hahaha");

System.out.println("Constructor of child class");

}

void display(){

System.out.println("Hello");

}

public static void main(String args[]){

Subclass obj= new Subclass();

obj.display();

}

}

**Output:**

parameterized constructor of parent class

Constructor of child class

Hello

**3) How to use super keyword in case of method overriding**

class Parentclass

{

//Overridden method

void display(){

System.out.println("Parent class method");

}

}

class Subclass extends Parentclass

{

//Overriding method

void display(){

System.out.println("Child class method");

}

void printMsg(){

//This would call Overriding method

display();

//This would call Overridden method

super.display();

}

public static void main(String args[]){

Subclass obj= new Subclass();

obj.printMsg();

}

}

**Output:**

Child class method

Parent class method

**What if the child class is not overriding any method: No need of super**

class Parentclass

{

void display(){

System.out.println("Parent class method");

}

}

class Subclass extends Parentclass

{

void printMsg(){

/\* This would call method of parent class,

\* no need to use super keyword because no other

\* method with the same name is present in this class

\*/

display();

}

public static void main(String args[]){

Subclass obj= new Subclass();

obj.printMsg();

}

}

### Example 3: super Keyword in Inheritance

class Animal {

// method in the superclass

public void eat() {

System.out.println("I can eat");

}

}

// Dog inherits Animal

class Dog extends Animal {

// overriding the eat() method

@Override

public void eat() {

// call method of superclass

super.eat();

System.out.println("I eat dog food");

}

// new method in subclass

public void bark() {

System.out.println("I can bark");

}

}

class Main {

public static void main(String[] args) {

// create an object of the subclass

Dog labrador = new Dog();

// call the eat() method

labrador.eat();

labrador.bark();

}

}

### Example 4: protected Members in Inheritance

class Animal {

protected String name;

protected void display() {

System.out.println("I am an animal.");

}

}

class Dog extends Animal {

public void getInfo() {

System.out.println("My name is " + name);

}

}

class Main {

public static void main(String[] args) {

// create an object of the subclass

Dog labrador = new Dog();

// access protected field and method

// using the object of subclass

labrador.name = "Rocky";

labrador.display();

labrador.getInfo();

}

}

**Inheritance Example:**

class Calculation {

int z;

public void addition(int x, int y) {

z = x + y;

System.out.println("The sum of the given numbers:"+z);

}

public void Subtraction(int x, int y) {

z = x - y;

System.out.println("The difference between the given numbers:"+z);

}

}

public class My\_Calculation extends Calculation {

public void multiplication(int x, int y) {

z = x \* y;

System.out.println("The product of the given numbers:"+z);

}

public static void main(String args[]) {

int a = 20, b = 10;

My\_Calculation demo = new My\_Calculation();

demo.addition(a, b);

demo.Subtraction(a, b);

demo.multiplication(a, b);

}

}

**Inheritance Example”**

class Superclass {

int age;

Superclass(int age) {

this.age = age;

}

public void getAge() {

System.out.println("The value of the variable named age in super class is: " +age);

}

}

public class Subclass extends Superclass {

Subclass(int age) {

super(age);

}

public static void main(String args[]) {

Subclass s = new Subclass(24);

s.getAge();

}

}

**Inheritance Example:**

class Animal {

}

class Mammal extends Animal {

}

class Reptile extends Animal {

}

public class Dog extends Mammal {

public static void main(String args[]) {

Animal a = new Animal();

Mammal m = new Mammal();

Dog d = new Dog();

System.out.println(m instanceof Animal);

System.out.println(d instanceof Mammal);

System.out.println(d instanceof Animal);

}

}