**Inheritance**

* **derived class** (child) - the class that inherits from another class
* **base class** (parent) - the class being inherited from

class Super {

.....

.....

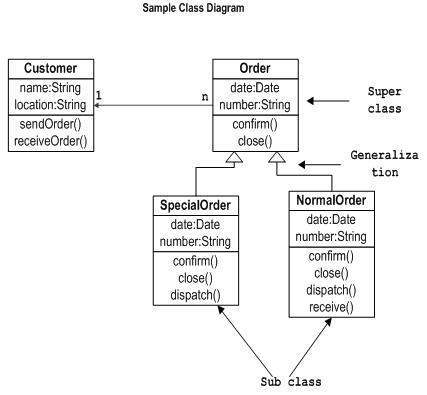
}

class Sub extends Super {

.....

.....

}



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);

}

}

The super keyword

The **super** keyword is similar to **this** keyword. Following are the scenarios where the super keyword is used.

* It is used to **differentiate the members** of superclass from the members of subclass if they have the same names.
* It is used to **invoke the superclass** constructor from the subclass.
* super.variable
* super.method();

**Example 1**

class Super\_class {

int num = 20;

// display method of superclass

public void display() {

System.out.println("This is the display method of superclass");

}

}

public class Sub\_class extends Super\_class {

int num = 10;

// display method of sub class

public void display() {

System.out.println("This is the display method of subclass");

}

public void my\_method() {

// Instantiating subclass

Sub\_class sub = new Sub\_class();

// Invoking the display() method of sub class

sub.display();

// Invoking the display() method of superclass

super.display();

// printing the value of variable num of subclass

System.out.println("value of the variable named num in sub class:"+ sub.num);

// printing the value of variable num of superclass

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

}

public static void main(String args[]) {

Sub\_class obj = new Sub\_class();

obj.my\_method();

}

}

**Example 2**

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();

}

}

**Polymorphism**

Polymorphism means "many forms", and it occurs when we have many classes that are related to each other by inheritance.

Like we specified in the previous chapter; [**Inheritance**](https://www.w3schools.com/cpp/cpp_inheritance.asp) lets us inherit attributes and methods from another class. **Polymorphism** uses those methods to perform different tasks. This allows us to perform a single action in different ways.

For example, think of a base class called Animal that has a method called animalSound(). Derived classes of Animals could be Pigs, Cats, Dogs, Birds - And they also have their own implementation of an animal sound (the pig oinks, and the cat meows, etc.):

**Encapsulation**

The meaning of **Encapsulation**, is to make sure that "sensitive" data is hidden from users. To achieve this, you must declare class variables/attributes as private (cannot be accessed from outside the class). If you want others to read or modify the value of a private member, you can provide public **get** and **set** methods.

## IS-A Relationship

## IS-A is a way of saying: This object is a type of that object. Let us see how the extends keyword is used to achieve inheritance.

public class Animal {

}

public class Mammal extends Animal {

}

public class Reptile extends Animal {

}

public class Dog extends Mammal {

}

Method Overloading

**public** **class** MethodOverloading {

**static** **int** plusMethod(**int** x, **int** y) {

**return** x + y;

}

**static** **double** plusMethod(**double** x, **double** y) {

**return** x + y;

}

**public** **static** **void** main(String[] args) {

**int** myNum1 = *plusMethod*(8, 5);

**double** myNum2 = *plusMethod*(4.3, 6.26);

System.***out***.println("int: " + myNum1);

System.***out***.println("double: " + myNum2);

}

}