**Overloading**

Overloading is when you have multiple methods in the same scope, with the same name but different signatures.

//Overloading

public class test

{

public void getStuff(int id)

{}

public void getStuff(string name)

{}

}

**Overriding**

Overriding is a principle that allows you to change the functionality of a method in a child class.

//Overriding

public class test

{

public virtual getStuff(int id)

{

//Get stuff default location

}

}

public class test2 : test

{

public override getStuff(int id)

{

//base.getStuff(id);

//or - Get stuff new location

}

}

Having more than one methods/constructors with same name but different parameters is called overloading. This is a compile time event.

Class Addition

{

int add(int a, int b)

{

return a+b;

}

int add(int a, int b, int c)

{

return a+b+c;

}

public static main (String[] args)

{

Addition addNum = new Addition();

System.out.println(addNum.add(1,2));

System.out.println(addNum.add(1,2,3));

}

}

O/p:

3

6

Overriding is a run time event, meaning based on your code the output changes at run time.

class Car

{

public int topSpeed()

{

return 200;

}

}

class Ferrari extends Car

{

public int topSpeed()

{

return 400;

}

public static void main(String args[])

{

Car car = new Ferrari();

int num= car.topSpeed();

System.out.println("Top speed for this car is: "+num);

}

}

Notice there is a common method in both classes topSpeed(). Since we instantiated a Ferrari, we get a different result.

O/p:

Top speed for this car is: 400