Method Overloading

Example 1

```
package Overloading3;

/**
 *
 * @author Eshana
 */

// Two overloaded methods,
public class Morning {

 // first sum method performs addition of two numbers
public void sum(int a, int b)
{
System.out.println(a+b);
}

// second sum method performs addition of three numbers
public void sum(int a,int b,int c)
{
System.out.println(a+b+c);
}

   public static void main(String[] args) {
      Morning h=new Morning();
      h.sum(10,20);
      h.sum(10,20,30);
      System.out.println("Hello Java");
   }

}
```

Example 2

```
package Overloading4;

/**
 *
 * @author Eshana
 */
public class hello {

//Two overloaded methods that differs in data type.

   //first sum method receives two integer arguments
   public void sum(int a, int b)
   {
   System.out.println(a+b);
   }
   //second sum method receives two double arguments.
   public void sum(double a, double b)
   {
   System.out.println(a+b);
   }
   public static void main(String[] args) {

      hello h = new hello();
      h.sum(10,20);
      h.sum(10.20, 30.50);
      System.out.println("Hello Java");
   }

}
```

Method Overriding

Example 1

```
package Overriding1;

/**
 *
 * @author Eshana
 */
public class Parent {
   int a=50;
   public void display()
   {
   System.out.println("Display method of Parent class");
   }
}
```

```
package Overriding1;

/**
 *
 * @author Eshana
 */
public class Child extends Parent {
   int b=100;
   public void display()
   {
   System.out.println("Display method of Child class");
   }
}
```

```
package Overriding1;

/**
 *
 * @author Eshana
 */
//display method in the subclass as defined in the parent class but it has some specific implementation.
public class MethodOverriding {

   public static void main(String[] args) {
    Child d=new Child();
    d.display();
   }
}
```

Runtime Polymorphism

Example 1

```
package Polymorphism1;
/**
 *
 * @author Eshana
 */
public class Shape {
   void draw(){
      System.out.println("drawing...");
   }

}
```

```
package Polymorphism1;
/**
 *
 * @author Eshana
 */
public class Circle extends Shape {
  void draw(){
     System.out.println("drawing circle...");
  }
}
```

```
package Polymorphism1;

/**
 *
 * @author Eshana
 */
public class Rectangle extends Shape{
   void draw(){
      System.out.println("drawing rectangle...");
   }
}
```

```
package Polymorphism1;

/**
 *
 * @author Eshana
 */
public class Triangle extends Shape {
   void draw(){
      System.out.println("drawing triangle...");
   }
}
```

```
package Polymorphism1;

/**
 *
 * @author Eshana
 */
public class Test {

   public static void main(String[] args) {
      //created object & implemented the multiple methods
      Shape s;
      s=new Rectangle();
      s.draw();
      s=new Circle();
      s.draw();
      s=new Triangle();
      s.draw();
   }

}
```

Multiple Inheritance

Example 1

```
package MultipleInheritance1;

/**
 *
 * @author Eshana
 */
public interface Showable {
 void show();
}
```

```
package MultipleInheritance1;

/**
 *
 * @author Eshana
 */

//Interface only one specifier is Public
public interface Printable {
 void print();
}
```

```
package MultipleInheritance1;

/**
 *
 * @author Eshana
 */
public class Both implements Printable,Showable {
   public void print(){
      System.out.println("Hello");
   }

   public void show(){
      System.out.println("Welcome");
   }
```

```
    public static void main(String[] args) {
        Both obj = new Both();
        obj.print();
        obj.show();
    }

}
```

Abstract class

Example 1

```
package AbstractClass1;

/**
 *
 * @author Eshana
 */
abstract class Bike {
    Bike(){
        System.out.println("bike is created");
    }

   abstract void run();

   void changeGear(){
      System.out.println("gear changed");
   }

}
```

```
package AbstractClass1;

/**
 *
 * @author Eshana
 */

//Creating a Child class which inherits Abstract class
 class Honda extends Bike {
  void run(){
     System.out.println("running safely..");
  }
}


package AbstractClass1;

/**
 *
 * @author Eshana
 */
public class TestAbstraction {
//An abstract class can have a data member,
//abstract method, method body (non-abstract method), constructor, and even main() method.
   public static void main(String[] args) {
        Bike obj = new Honda();
        obj.run();
        obj.changeGear();
   }

}
```