**Generic class**

A class that can refer to any type is known as generic class. Here, we are using **T** type parameter to create the generic class of specific type.

Let’s see the simple example to create and use the generic class.

**Creating generic class:**

class MyGen<T>{

T obj;

void add(T obj){this.obj=obj;}

T get(){return obj;}

}

The T type indicates that it can refer to any type (like String, Integer, Employee etc.). The type you specify for the class, will be used to store and retrieve the data.

**Using generic class:**

Let’s see the code to use the generic class.

class TestGenerics3{

public static void main(String args[]){

MyGen<Integer> m=new MyGen<Integer>();

m.add(2);

//m.add("vivek");//Compile time error

System.out.println(m.get());

}}

Output:2

**Type Parameters**

The type parameters naming conventions are important to learn generics thoroughly. The commonly type parameters are as follows:

1. T - Type
2. E - Element
3. K - Key
4. N - Number
5. V - Value

**Generic Method**

Like generic class, we can create generic method that can accept any type of argument.

Let’s see a simple example of java generic method to print array elements. We are using here **E** to denote the element.

public class TestGenerics4{

   public static < E > void printArray(E[] elements) {

        for ( E element : elements){

            System.out.println(element );

         }

         System.out.println();

    }

    public static void main( String args[] ) {

        Integer[] intArray = { 10, 20, 30, 40, 50 };

        Character[] charArray = { 'J', 'A', 'V', 'A', 'T','P','O','I','N','T' };

        System.out.println( "Printing Integer Array" );

        printArray( intArray  );

       System.out.println( "Printing Character Array" );

        printArray( charArray );

    }

}

[Test it Now](http://www.javatpoint.com/opr/test.jsp?filename=TestGenerics4)

Output:Printing Integer Array

10

20

30

40

50

Printing Character Array

J

A

V

A

T

P

O

I

N

T

**Wildcard in Java Generics**

The ? (question mark) symbol represents wildcard element. It means any type. If we write <? extends Number>, it means any child class of Number e.g. Integer, Float, double etc. Now we can call the method of Number class through any child class object.

Let's understand it by the example given below:

1. import java.util.\*;
2. abstract class Shape{
3. abstract void draw();
4. }
5. class Rectangle extends Shape{
6. void draw(){System.out.println("drawing rectangle");}
7. }
8. class Circle extends Shape{
9. void draw(){System.out.println("drawing circle");}
10. }

13. class GenericTest{
14. //creating a method that accepts only child class of Shape
15. public static void drawShapes(List<? extends Shape> lists){
16. for(Shape s:lists){
17. s.draw();//calling method of Shape class by child class instance
18. }
19. }
20. public static void main(String args[]){
21. List<Rectangle> list1=new ArrayList<Rectangle>();
22. list1.add(new Rectangle());
24. List<Circle> list2=new ArrayList<Circle>();
25. list2.add(new Circle());
26. list2.add(new Circle());
28. drawShapes(list1);
29. drawShapes(list2);
30. }}

drawing rectangle

drawing circle

drawing circle