**Generics-2022**

.Usage of Generics in Java

# **Type-1**

Let use create a generic type java class.

**public class GenericQ1<T> {**

private LinkedList<T> list = new LinkedList<>();  
  
 public void offer(T t) {  
 list.add(t);  
 }  
  
 public T poll() {  
 T t = list.removeFirst();  
 return t;  
 }  
  
 @Override  
 public String toString() {  
 return list.toString();  
 }  
  
 public static void main(String[] args) {  
 GenericQ1<Person> personQ = new GenericQ1<>();  
  
 for (int i = 0; i < 5; i++) {  
 Person p = new Person("Name-" + i);  
 personQ.offer(p);  
 }  
  
 System.*out*.println("All Persons : " + personQ);  
 Person person1 = personQ.poll();  
 System.*out*.println("Removed Person :::" + person1);  
 }  
}

In the above case the class has been defined with generic type <T>. The structure is given below.

**public class GenericQ1<T> {**

}

So the class is fully generic type.

# **Type – 2**

Let use create a general class and create generic method. The code is given below.

public class GenericQ2 {

private LinkedList<Object> list = new LinkedList<>();

public <T> void offer(T t) {

list.add(t);

}

public <T> T poll() {

T t = (T) list.removeFirst();

return t;

}

@Override

public String toString() {

return list.toString();

}

public static void main(String[] args) {

GenericQ2 personQ = new GenericQ2();

for (int i = 0; i < 5; i++) {

Person p = new Person("Name-" + i);

personQ.offer(p);

}

System.out.println("All Persons : " + personQ);

Person person1 = personQ.poll();

System.out.println("Removed Person :::" + person1);

}

}

Now you can mark the above highlighted area. Here the class is not of type generic. Let us see the basic difference.

public class GenericQ2<T>

public class GenericQ2

Always remember that , if the class is not generic type and if you are defining pure generic method, you have to define the method in the following manner.

**<accessModifier> <T> returnType(can be void) method() {**

**}**

Example is given below.

public **<T> void** offer(T t) { }

public **<T> T** poll() { }

Trick: First write the method with method return type, add <T> before

**How to create a Generic Type Class**

public class Gen<T> {  
  
 public void doSomething(T t) {  
 if(t instanceof String) {  
 System.*out*.println("It is a String");  
 } else {  
 System.*out*.println("Not...");  
 }  
 }  
  
 public T getSomething(T t) {  
 return t;  
 }  
  
 public T get(T t) {  
 **T[] ts = (T[]) Array.*newInstance*(t.getClass(), 5);**  
 ts[0] = t;  
 T t1 = ts[0];  
 return t;  
 }  
}

Test

public static void main(String[] args) {  
 Gen<String> gen = new Gen<String>();  
 gen.doSomething("abcd");  
 String val1 = gen.get("PQRS");  
 System.*out*.println(val1);  
 String value2 = gen.getSomething("Hati");  
 System.*out*.println(value2);  
}