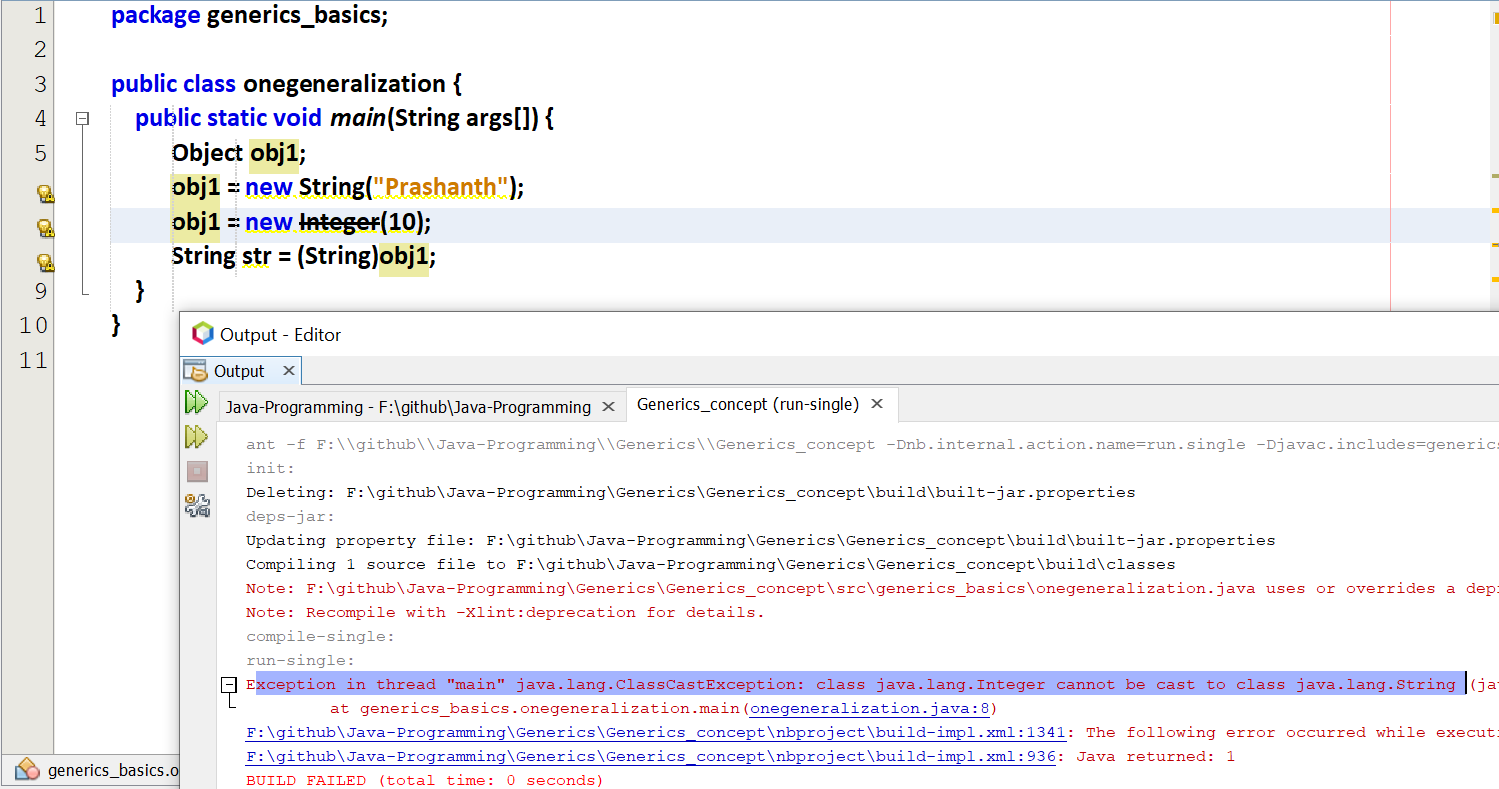
# **Before Generics**

Class Object 🡪 this the built-in-class / mother classes for all the JAVA classes.  
Every class is directly/indirectly from object class.

With the reference of an object of the object [parent class] class we can assign to any class [child class]   
Object obj = new String(“Hi”); [ Generalization ]



At compile time, it is accepting without any prompt. But the error is raised only at the run-time.  
[integer object cannot be converted into string]



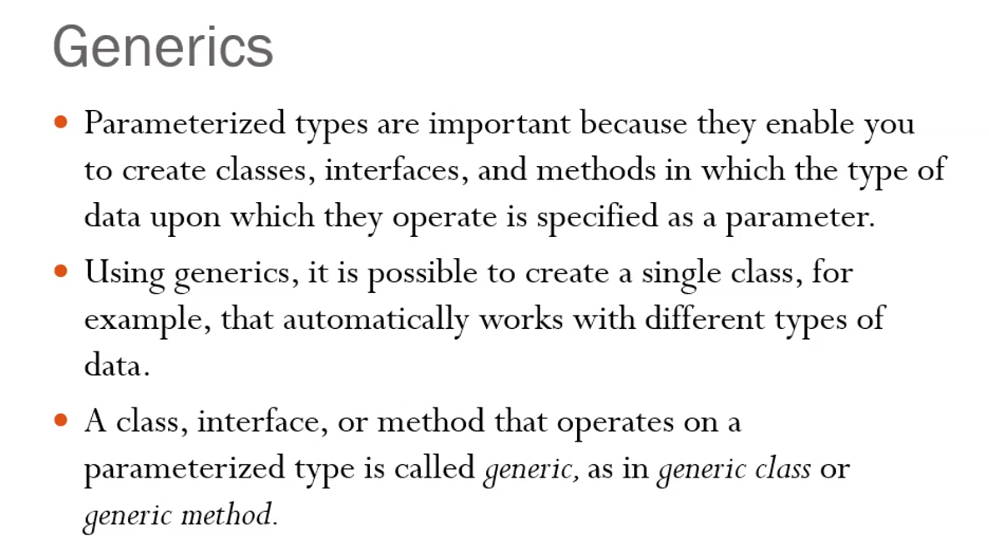
Here the 0th and 1st index 🡪 string  
2nd index 🡪 int  
But the main draw-back is, the error is raised only at the run-time.

Therefore we want an array to store only integers. If the 0th index is integer, then the complete array must be filled with integers.  
Therefore we want an array to store only string. If the 0th index is string , then the complete array must be filled with string.

But the array must be capable of storing any data-type. Before using that array mention that data-type.

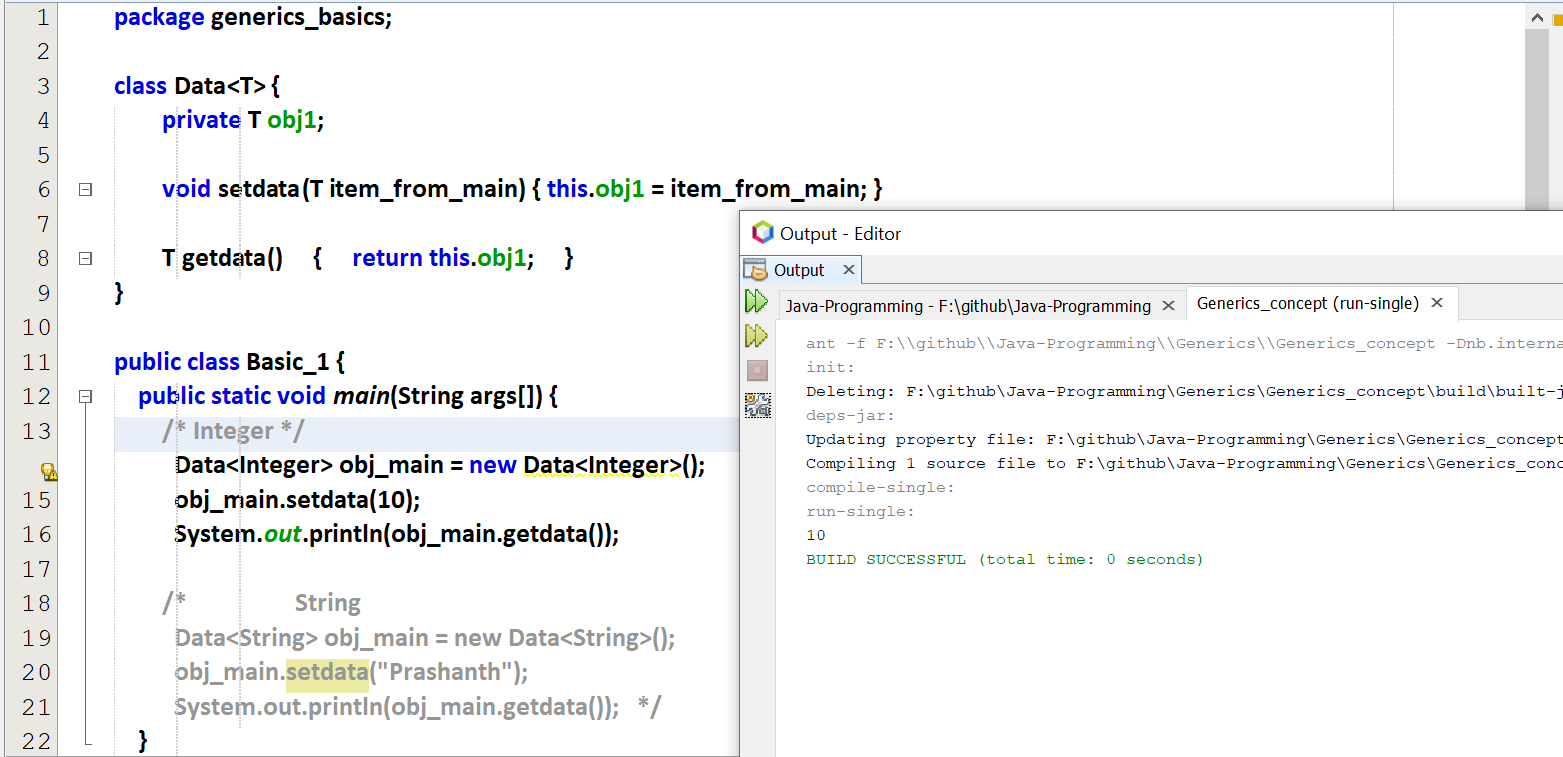
This can be done by using generics

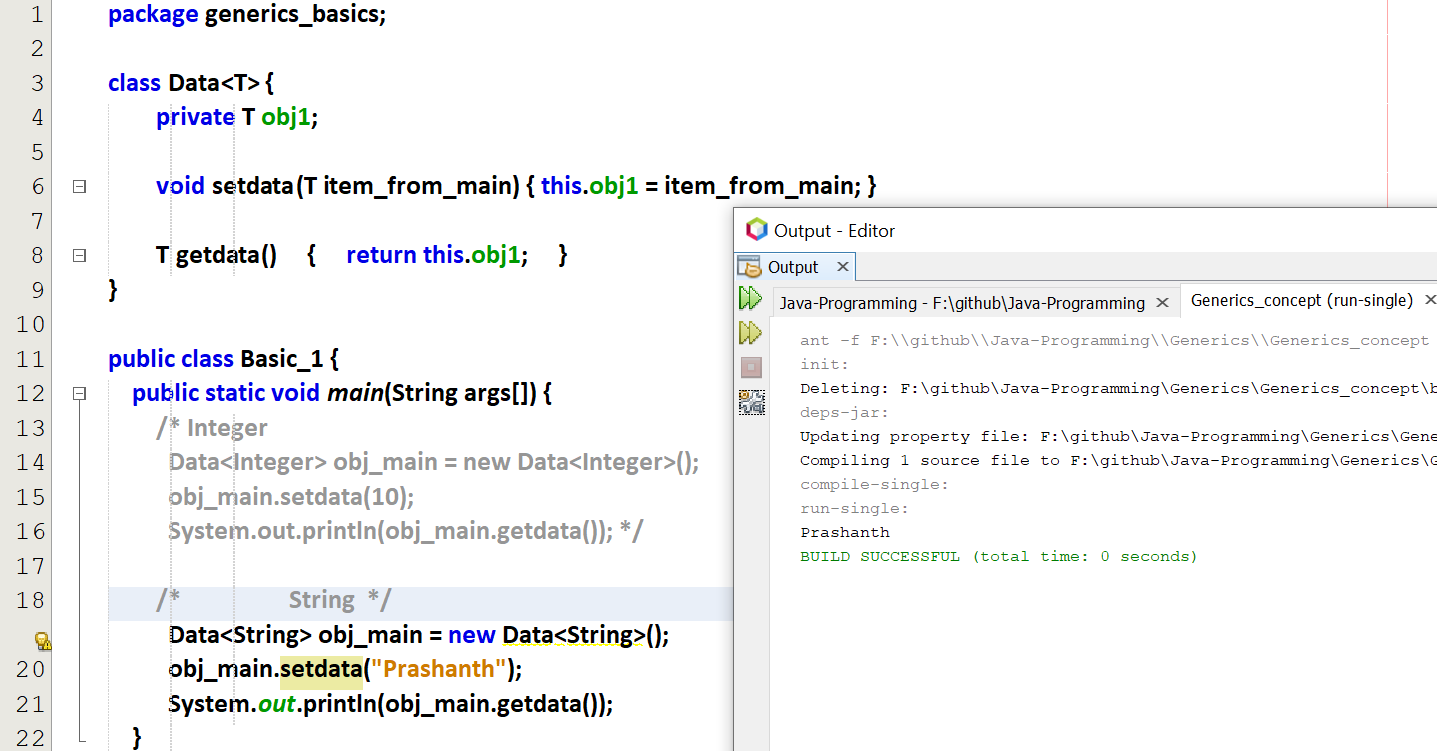
# **Generics**



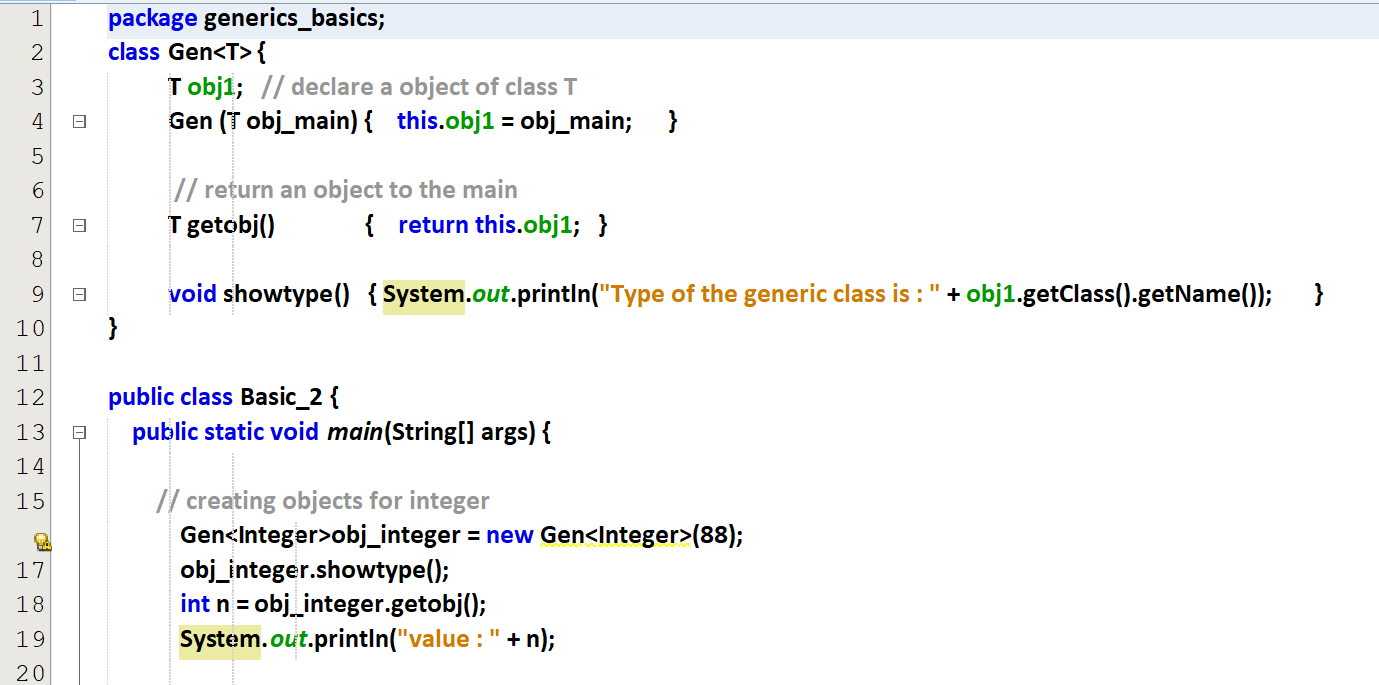
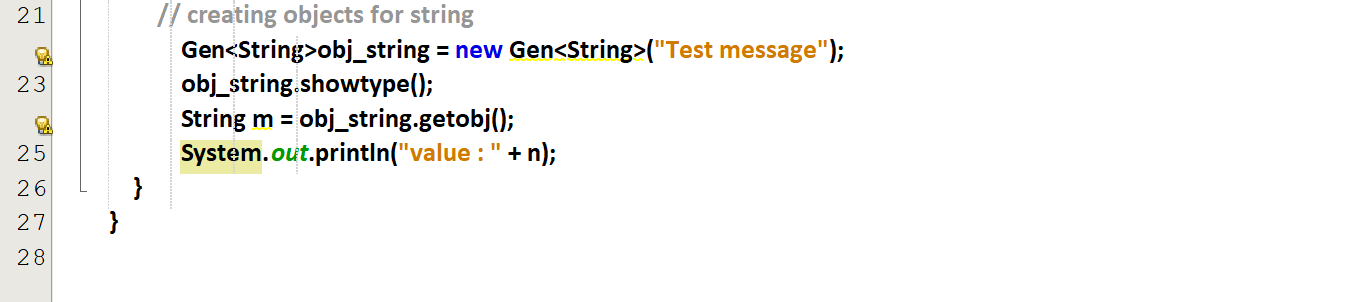
Templates in C++ 🡪 primitive data-types  
Generics in JAVA 🡪 objects

## **Refer Basic\_1.java**

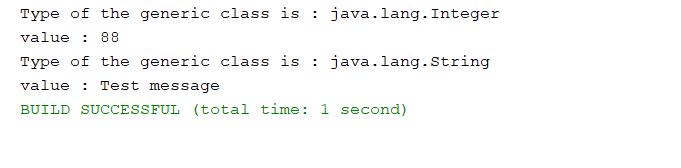




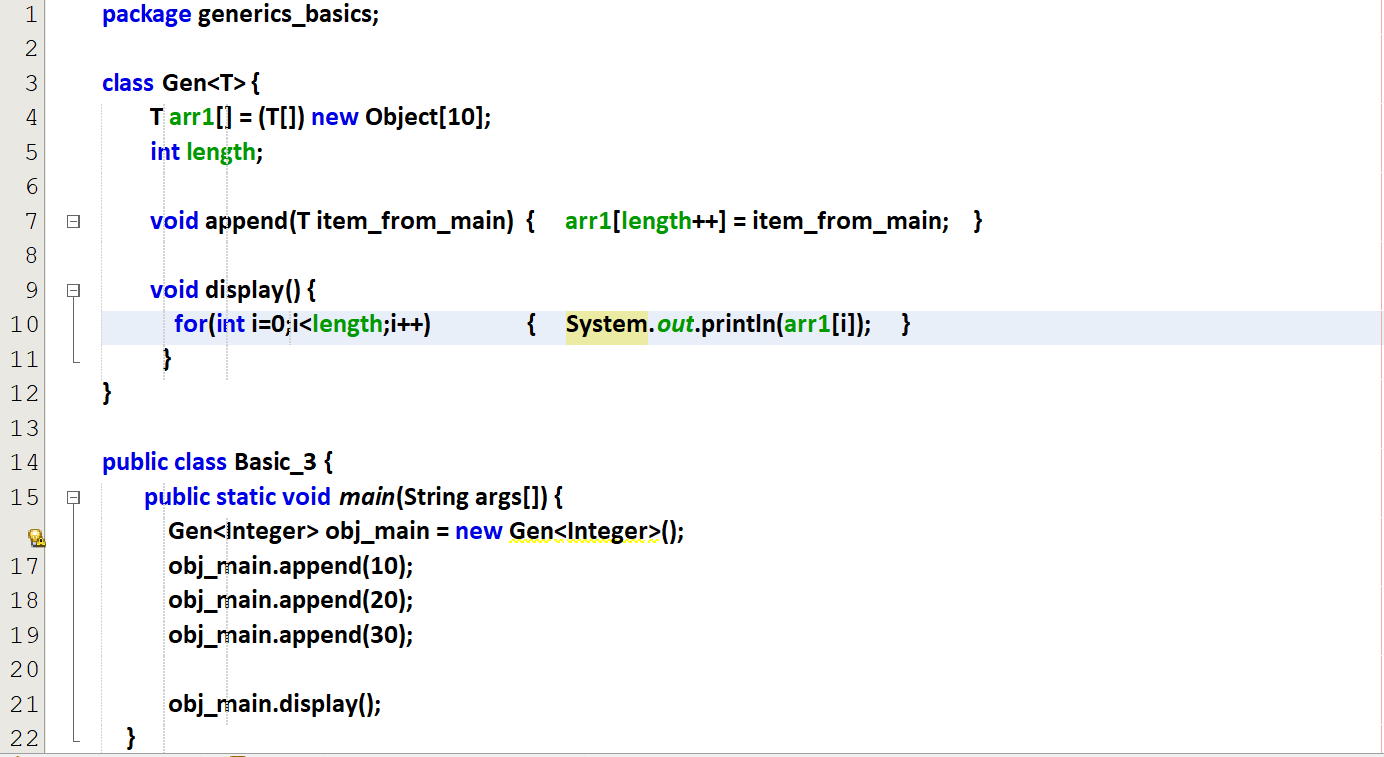
## **Refer Basic\_2.java**

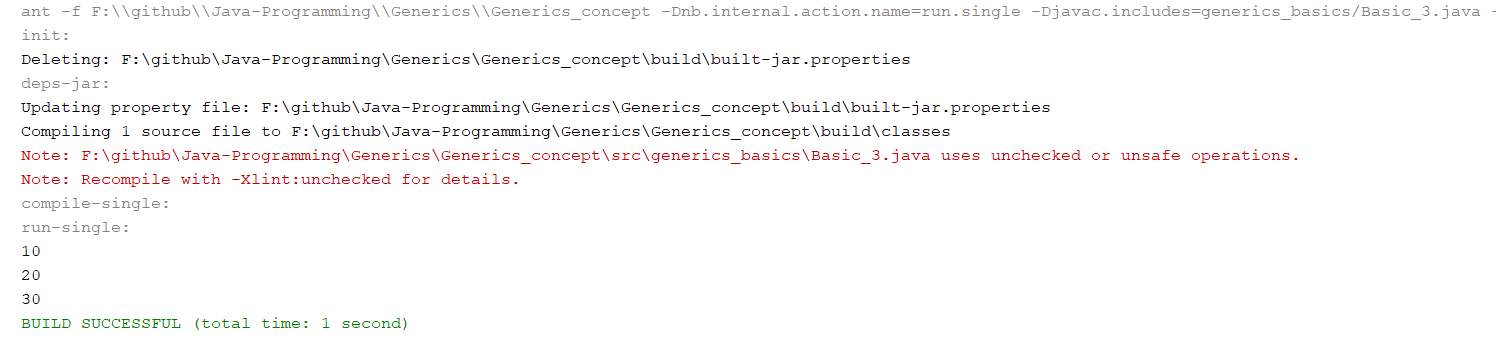
Output



## **Refer Basic\_3.java**

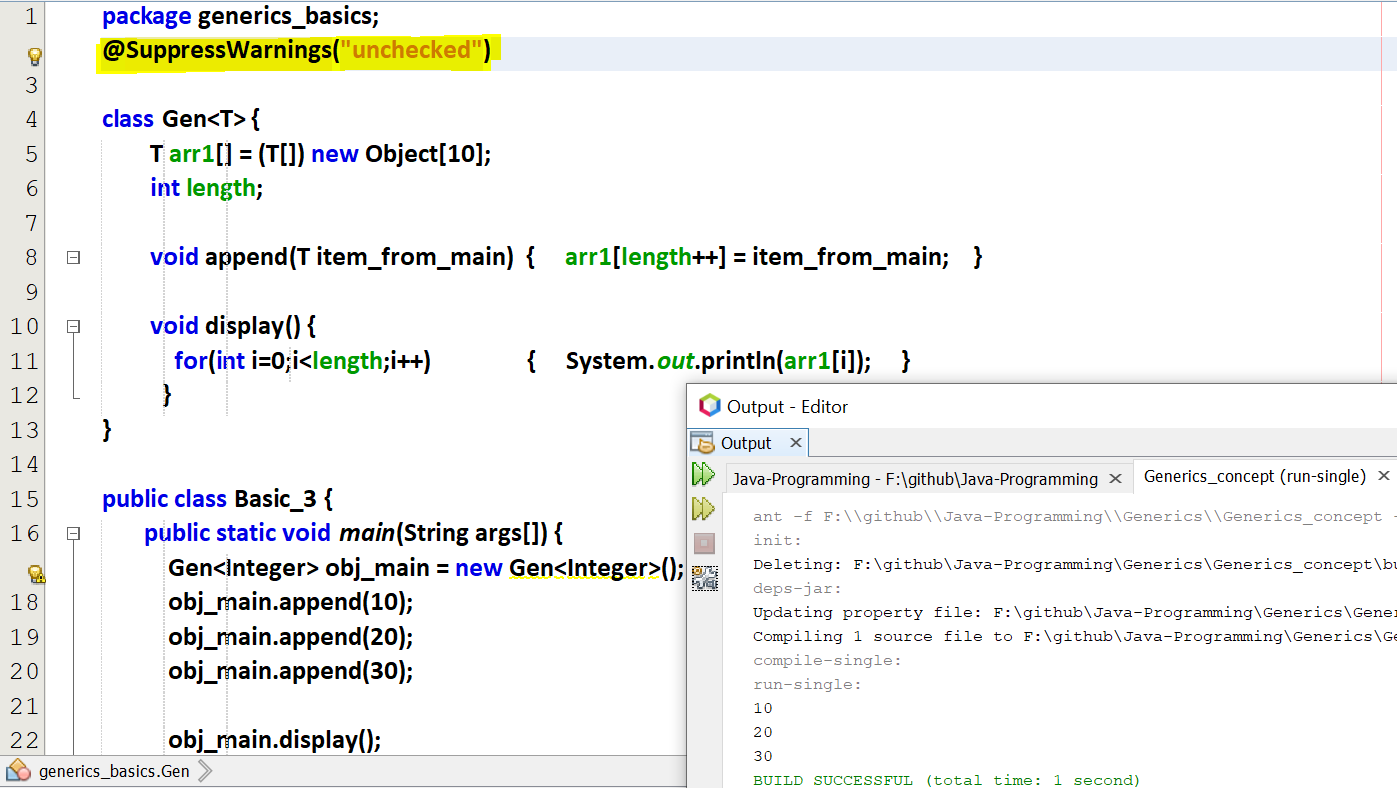


Output

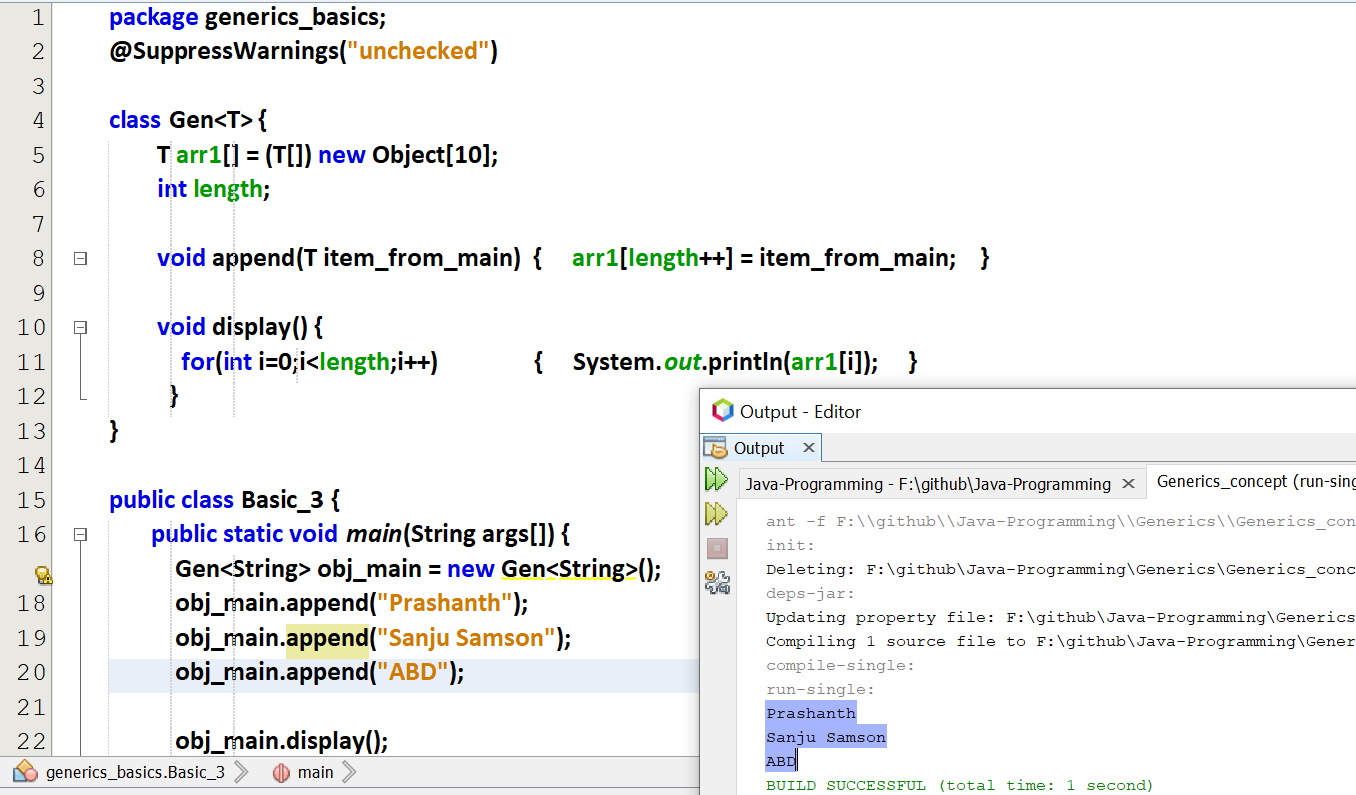


Reason for warning 🡪 Here the object is converted into Generic, we can avoid this by using   
@SuppressWarnings(“unchecked”)

With integer

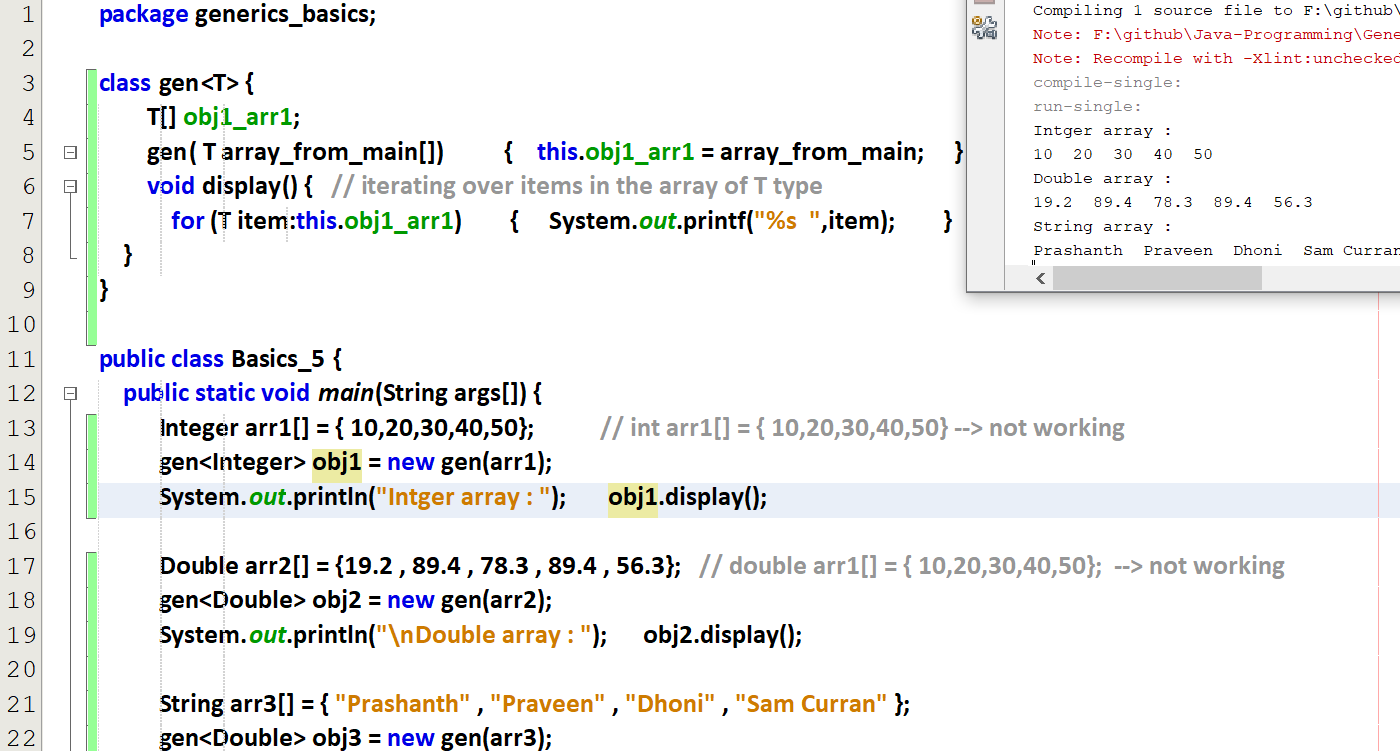


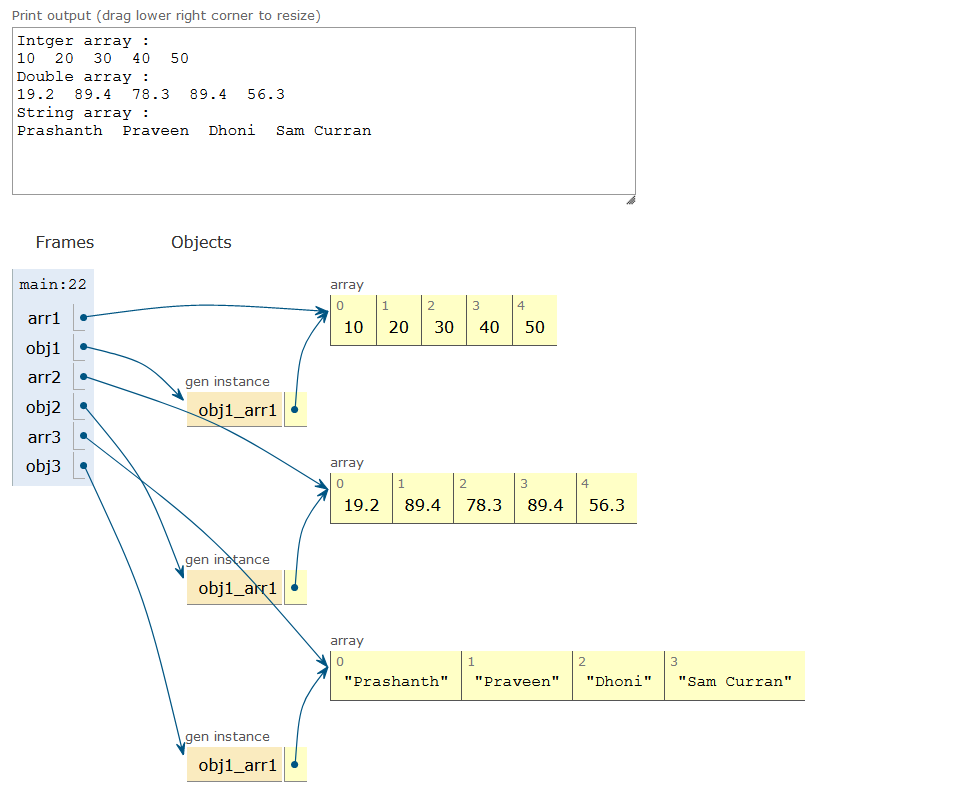
With strings

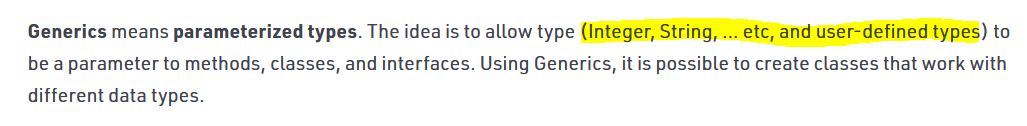


## **Refer Basic\_4.java Array as a Parameter in generic class**

## **Refer Basic\_5.java Array**

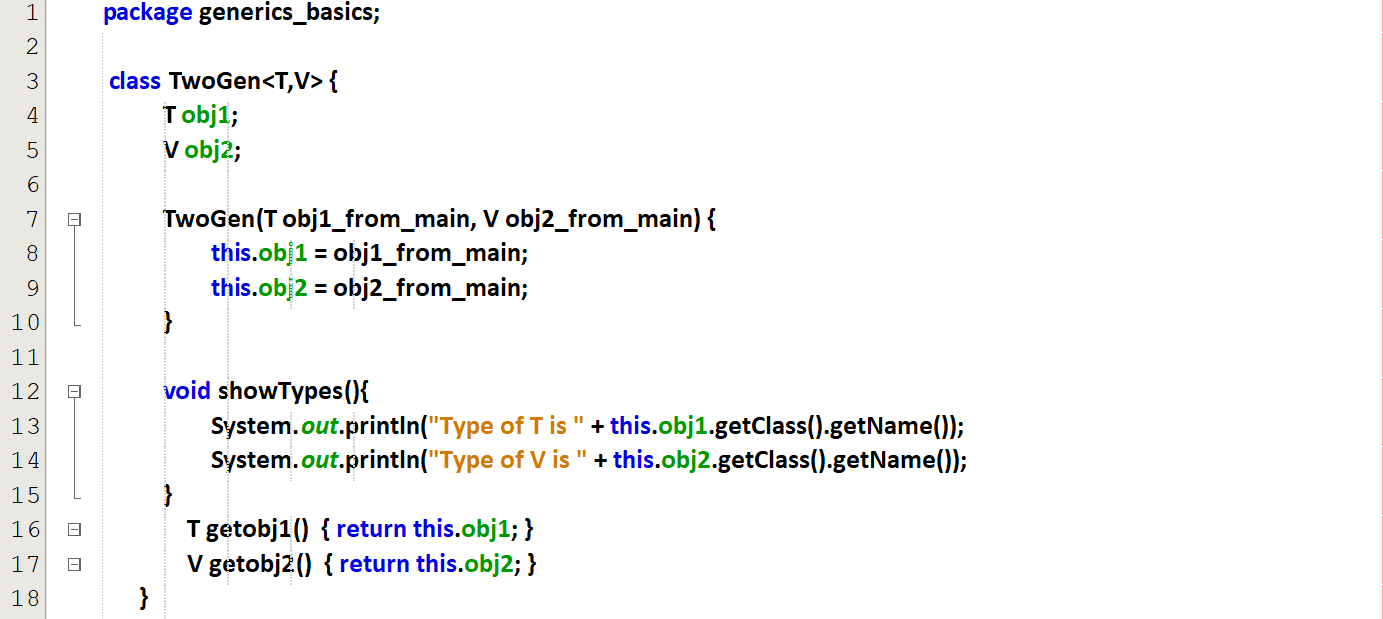
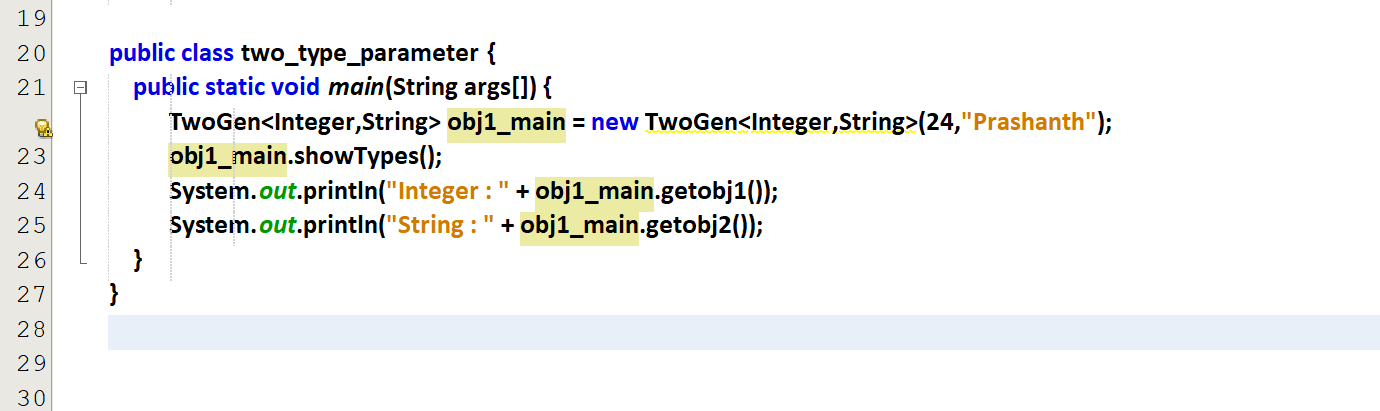




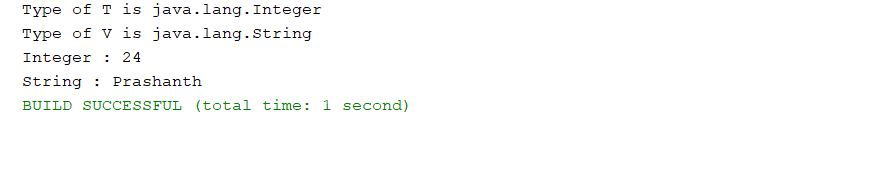


## **Two Type Parameters**

Refer two type parameter.java

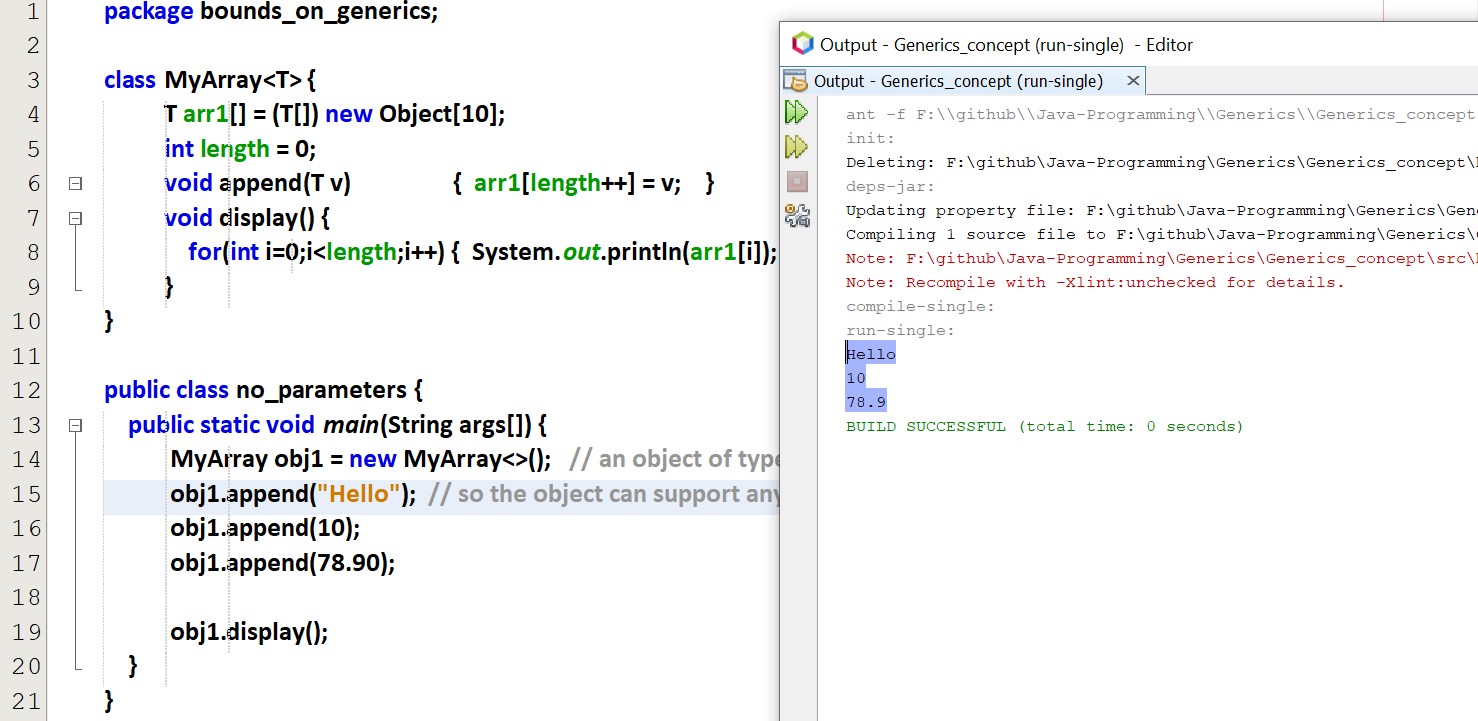
Output



# **Bounds on Generics**

## **No Parameters**

Refer no\_parameters.java



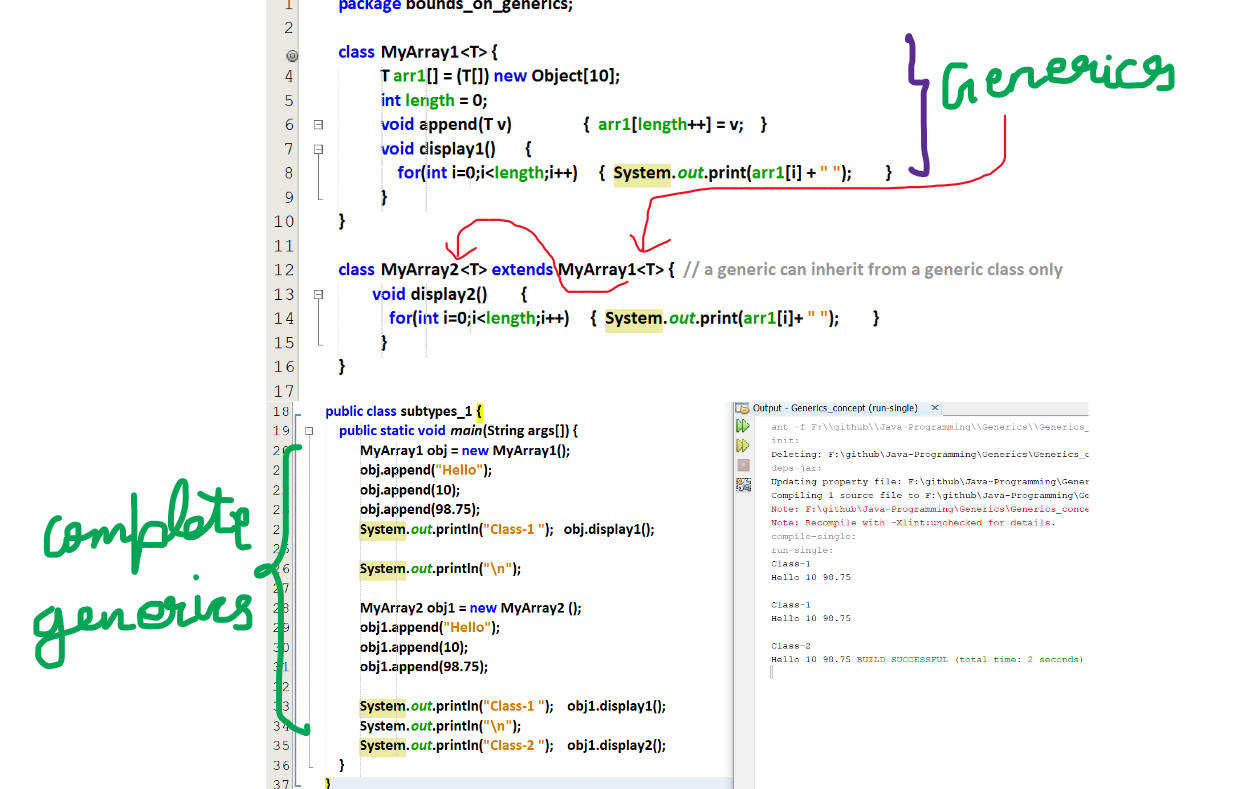
obj1 🡪 an object of type generic, since parameters are not given.  
Since the class is also Generic, it supports any type of object.

## **Multiple Parameters**

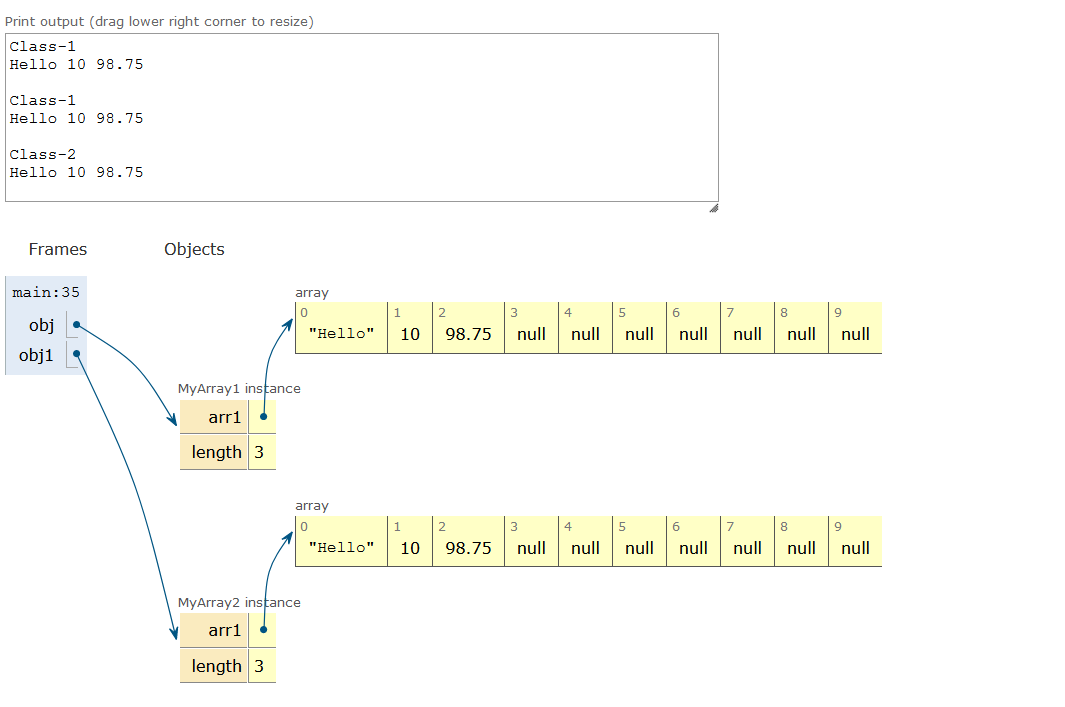
## **Subtypes**

A normal class cannot able to inherit Generic classes.  
A generic class only can able to inherit Generic classes.

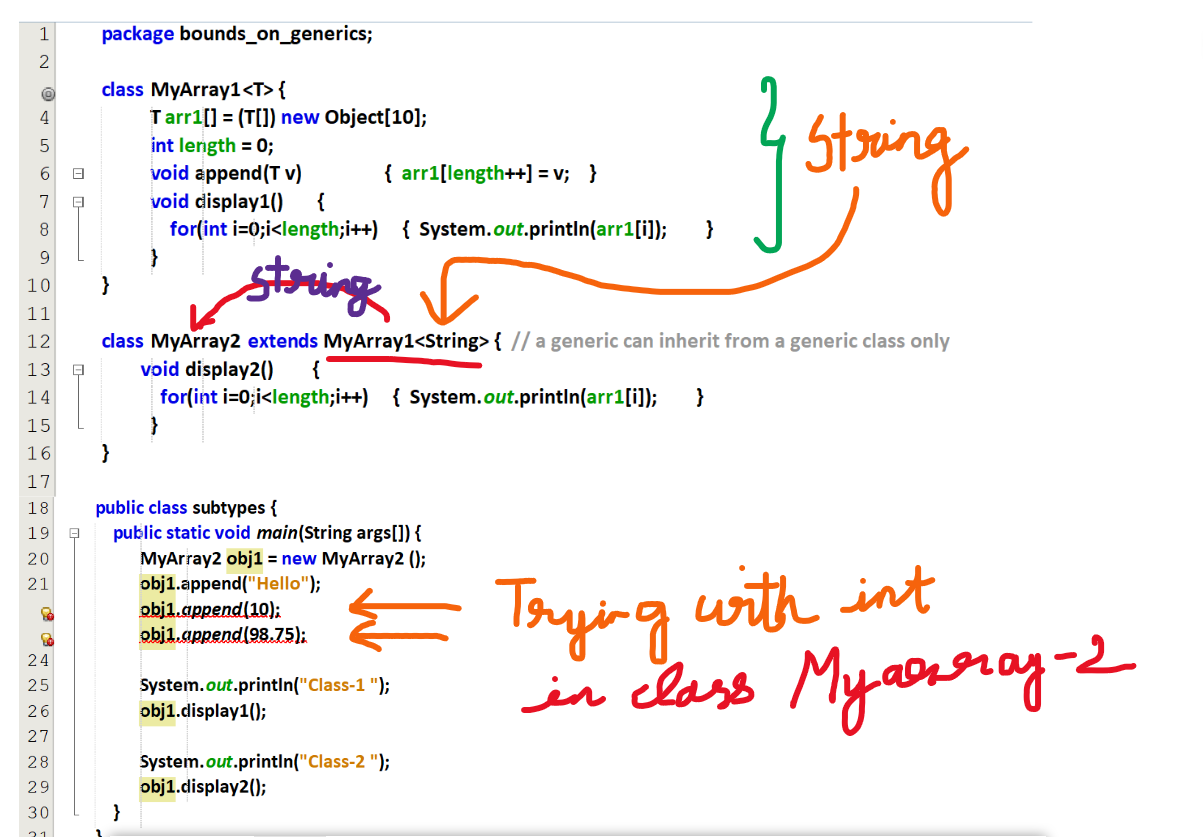
### **Refer subtypes\_1.java**

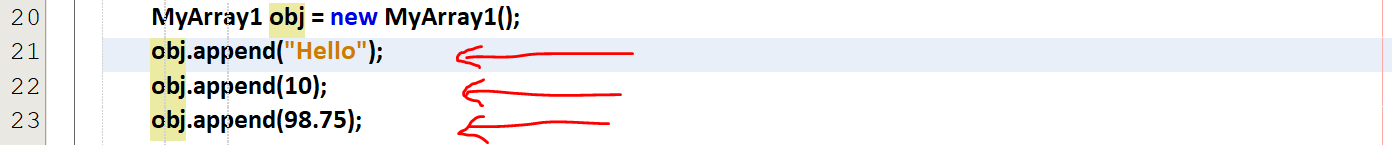


Both the classes are generic



### **Refer subtypes\_2.java**





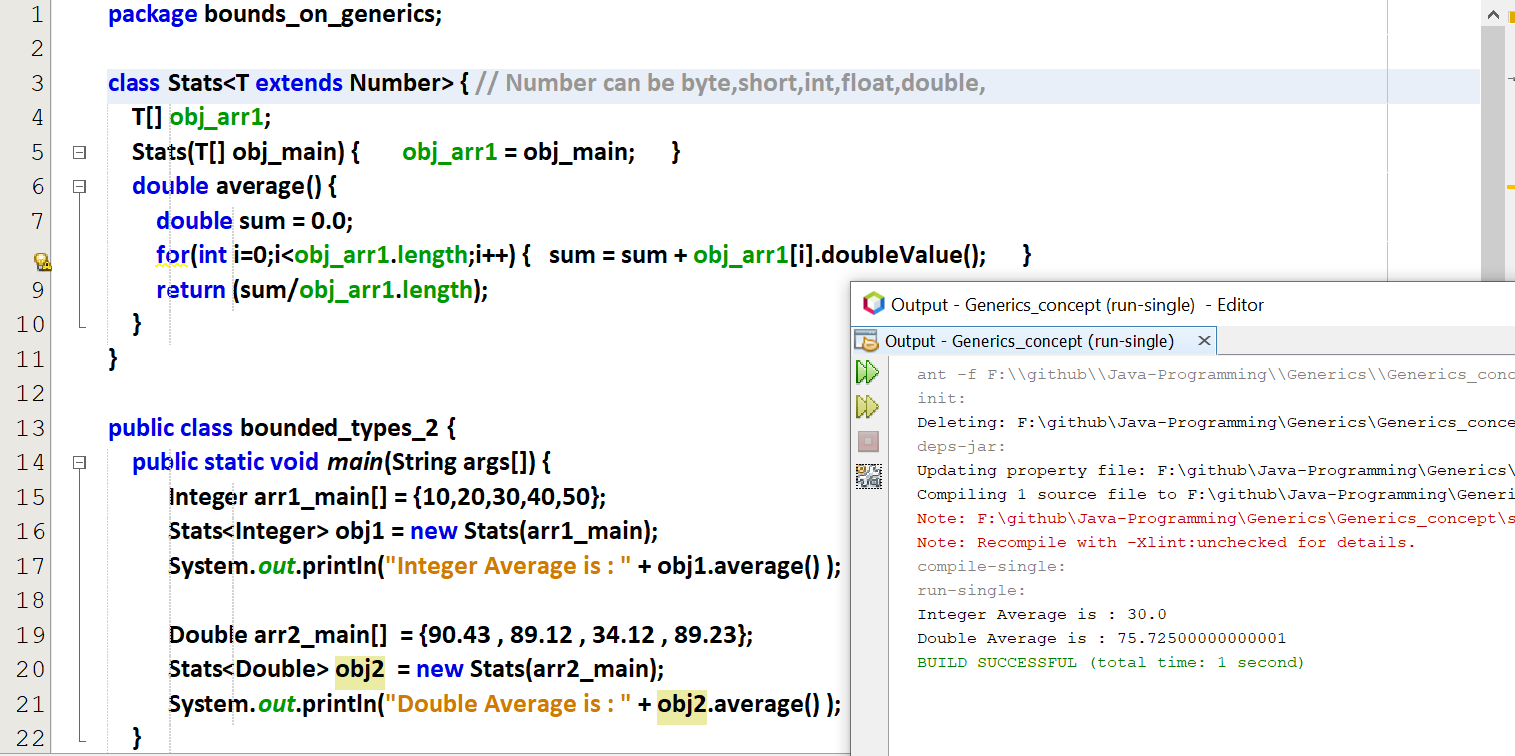
Myarray1 🡪 Generic   
object for Myarray1 🡪 Generic

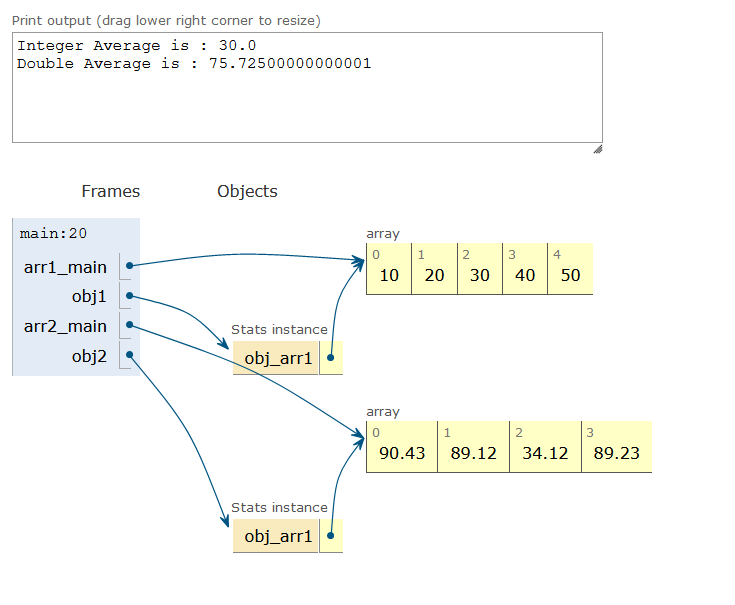
Myarray2 extends Myarray1<String>  
Myarray2 🡪 String  
object for Myarray2 🡪 String

## **Bounded Types**

### **Refer bound\_types\_1.java**

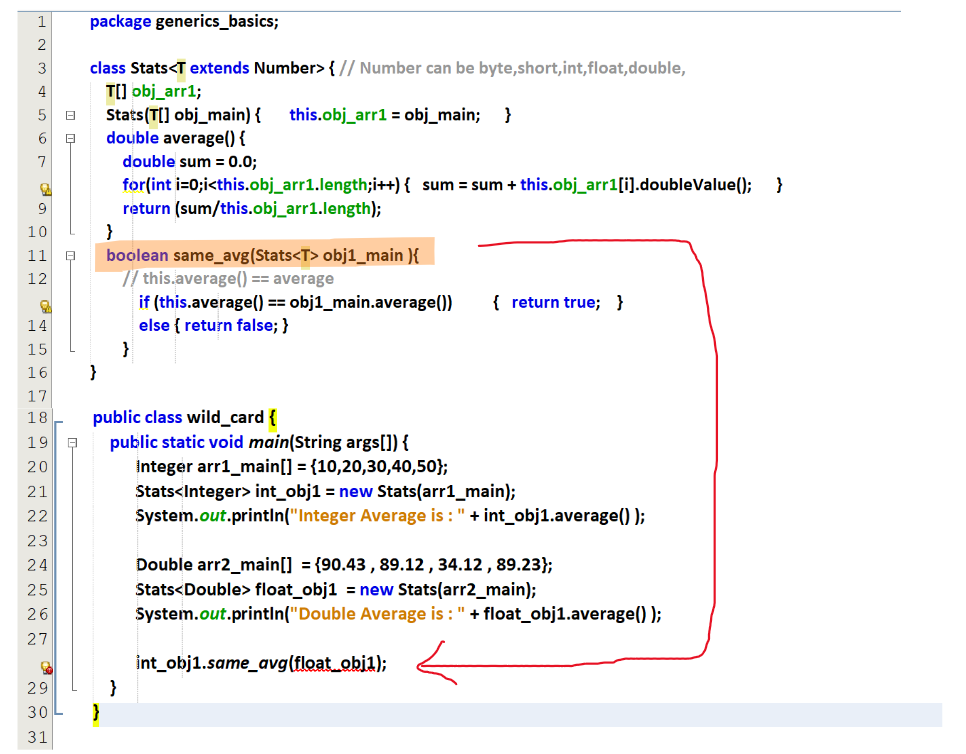
### **Refer bound\_types\_2.java**





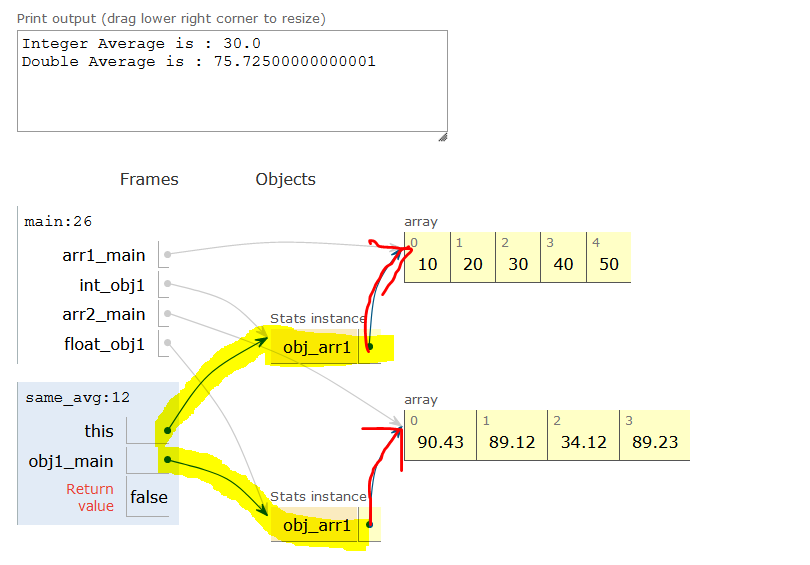
Forcefully we are making the generic class to be of Number

# **Need for wild card**



After putting ?





# **Generic Methods**

