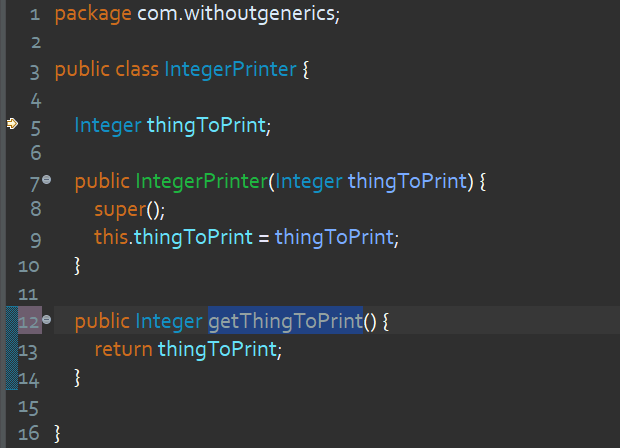
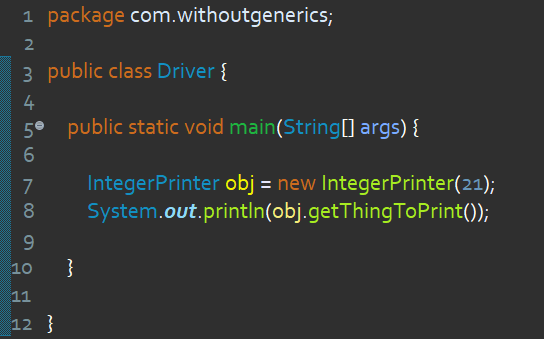
**What kind of problem java developer were running into before Generics exists ?**

Observe the below code,





Now what will happen, if I pass a double value while creating IntegerPrinter object ?



Yes, it is an error !

In the same way, if I pass a String value

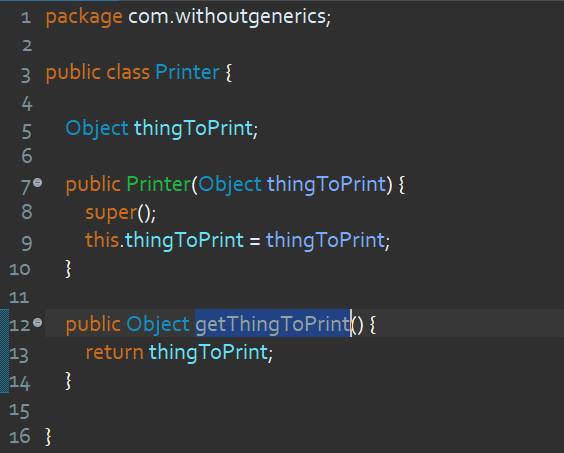


This is also an error.

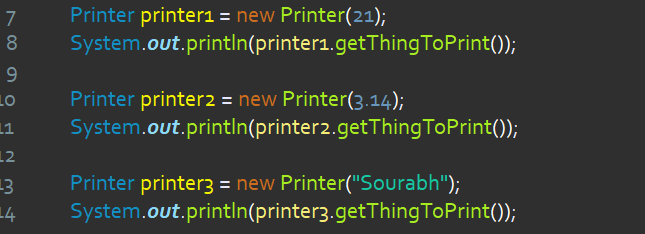
Conclusion is this class is not going to accept any type of value excluding Integer.

Now my task is to create a **Printer** class which is going to accept any type of value whether it is double, int, String etc.

A quick solution can come in our mind is that to create a class like below !



As Object class is the parent of every class in java, so whatever value you will pass it will definitely accepted now.



You can see Integer, Double, String all three different value has accepted by Printer class and we can print them as well.

So we have achieved what we wanted.

But still there is an problem in this approach !

**What is the problem ?**

Now suppose I want to perform few operations over the value I am getting from the method **getThingToPrint()**.

For an example, i want to add some number with whatever integer value **getThingToPrint()** returns.



As you can see, it is raising an error. But why ?

It is because our **getThingToPrint()** method returns a Object value and we are trying to add the Integer value with Object value which is not possible.

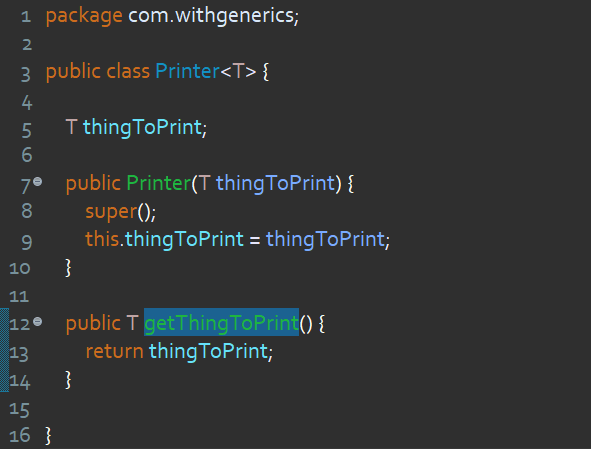
To resolve this issue we have to down cast the returned value into Integer.



Now the code is error free.

So whenever we use this method and perform some operation we have to downcast it every time. That is very hectic.

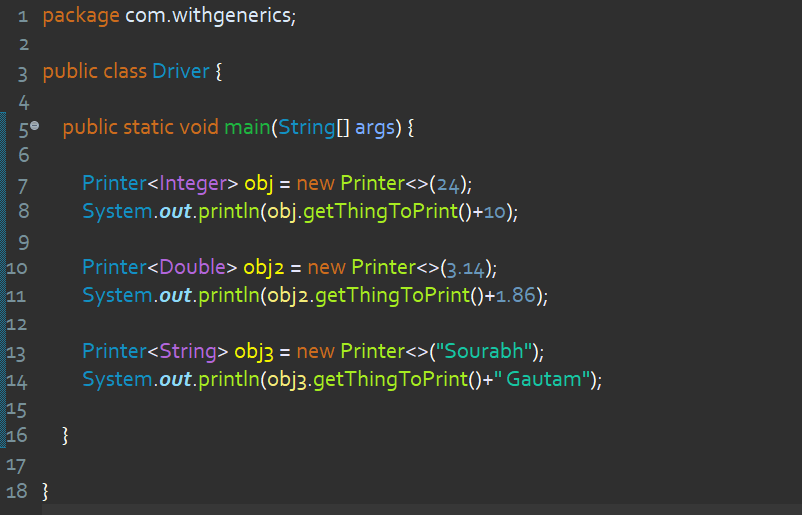
**This is where the generics come into picture. They can resolve this issue as well as they provides some more feature to generalize the classes.**



As you can see, here I am creating a class called Printer and to make it generic type the syntax we follow :

**class Printer<T> { }**

Where, T can be any identifier but conventions recommend T and it is stand for **Type**.



Now whenever we will create of Printer class object java will allow us to define what type of object is going to be insert into T so that java can replace all those T from the passing Type.



In this case, java will replace the T from Integer. This is also the solution of above two problems.

With generic we have to never type cast our object and we perform any operation related to the returned object.

**What if we don’t mention the generic type ?**

****

The T will then act as a Object type. So this is as similar as the previous solution where we were using Object class to refer any type of object to make our class generalize. So we should never go with this.

Also IDE will warn us that **Printer** is generic type, we should pass the Type.