package Generics;  
  
import com.sun.jdi.PathSearchingVirtualMachine;  
  
public class BoundedTypeParameterDemo {  
 public static void main(String[] args) {  
 Boxed<Integer> asd=new Boxed(12);  
 Integer k= asd.getValue();  
 System.*out*.println(k);  
}  
}  
  
class Boxed <N extends Number>{  
 private N value;  
  
 Boxed(N value){  
 this.value=value;  
 }  
  
 public N getValue(){  
 return this.value;  
 }  
}

**Note: Parameterized type can be only classes that extends Number class (Like Integer, Double, Long, Float) but cannot be String, char etc**

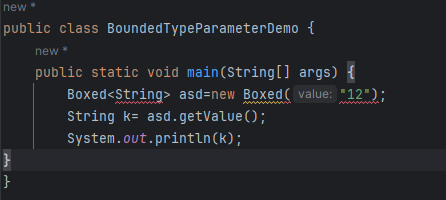
**Output**

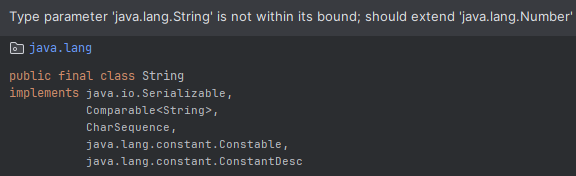
**C:\Users\Roystan\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.3\lib\idea\_rt.jar=60123:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.3\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath C:\Users\Roystan\IdeaProjects\JavaWorkspace\out\production\JavaWorkspace Generics.BoundedTypeParameterDemo**

**12**

**Process finished with exit code 0**

**Compile time error when string or char is used**

****

****

**Example 2**

package Generics;  
  
public class BoundedTypeParameterDemo2 {  
 public static void main(String[] args) {  
 actions jack=new actions();  
 jack.value= 13.0367675121241124124;  
 Boxx<actions> asd=new Boxx(jack);  
 asd.print();  
 }  
}  
  
interface Runnable{  
 void run();  
}  
interface Printable{  
 void print();  
}  
  
class actions extends Number implements Printable,Runnable{  
 public Number value;  
 @Override  
 public void print() {  
 System.*out*.println(doubleValue()+" ");  
 }  
  
 @Override  
 public int intValue() {  
 Integer intConvertedValue=(Integer) value;  
 return intConvertedValue.intValue();  
 }  
  
 @Override  
 public long longValue() {  
 Long longConvertedValue=(Long) value;  
 return longConvertedValue.longValue();  
 }  
  
 @Override  
 public float floatValue() {  
 Float floatConvertedValue=(Float) value;  
 return floatConvertedValue.floatValue();  
 }  
  
 @Override  
 public double doubleValue() {  
 Double doubleConvertedValue=(Double) value;  
 return doubleConvertedValue.doubleValue();  
 }  
  
 @Override  
 public void run() {  
 System.*out*.println("Runnable");  
 }  
}  
  
class Boxx <T extends Number & Printable & Runnable>{  
 private T item;  
  
 Boxx(T item){  
 this.item=item;  
 }  
  
 public T getItem(){  
 return this.item=item;  
 }  
  
 public void print(){  
 item.print();  
 }  
  
}

**C:\Users\Roystan\.jdks\openjdk-21.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.3\lib\idea\_rt.jar=60291:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2023.3.3\bin" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8 -Dsun.stderr.encoding=UTF-8 -classpath C:\Users\Roystan\IdeaProjects\JavaWorkspace\out\production\JavaWorkspace Generics.BoundedTypeParameterDemo2**

**13.036767512124113**

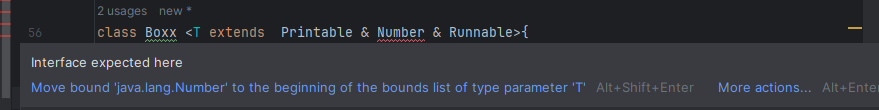
**Process finished with exit code 0**

**Errors from above example if not implemented**

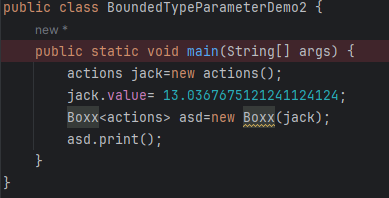
1. **class Boxx <T extends Number & Printable & Runnable> should always follow below rule**

**<T extends ClassName & Interface 1& Interface 2>**

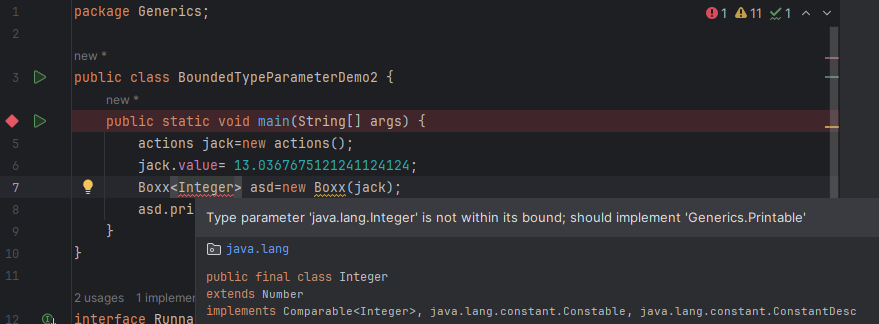
**Error when not implemented as per the rule**

****

1. **class satisfying above condition should only passed as arguments**

****

**Error when not implemented as per the rule**

****