

Scala Programming

Assignment

Name: Niranjana J
USN: 22BTRAD027
Branch: CSE- AI&DE

Question

In this assignment, students will create a custom type class in Scala and use it in an application. The type class should define a set of behaviours that can be applied to different types, such as numeric types, strings, or custom data types. The student should implement the type class using implicit conversions and demonstrate how it can be used in an application. The student should also write tests to ensure that the type class works correctly.

Code

```
class Change {  
  //converts integer to string  
  def change_str(x:Int): Unit = {  
    val result=x.toString;  
    println("The result is:"+result);  
    println("Type of the result is" +result.getClass());  
  }  
  //converts string to integer  
  def change_int(x:String): Unit = {  
    val result=x.toInt;  
    println("The result is:"+result);  
  }  
  
  //converts character to string  
  def change_char(x:Char): Unit = {  
    val result=x.toString;  
    println("The result is:"+result);  
  }  
  
  //converts character to ASCII value  
  def change_ascii(x:Char): Unit = {  
    val result=x.toInt;  
    println("The result is:"+result);  
  }  
}
```

```

    }
    object Main {
        def main(args: Array[String]): Unit = {

//creating new object
            val c1=new Change();

            val input1: Int=435;
            println("The input is "+input1);
            println("Type of the input is " +input1.getClass());
            c1.change_str(input1);

            val input2: String="98";
            println("The input is "+input2);
            println("Type of the input is " +input2.getClass());
            c1.change_int(input2);

            val input3: Char='a';
            println("The input is "+input3);
            println("Type of the input is " +input3.getClass());
            c1.change_char(input3);

            val input4: Char='k';
            println("The input is "+input4);
            println("Type of the input is " +input3.getClass());
            c1.change_ascii(input4) ;

        }
    }
}

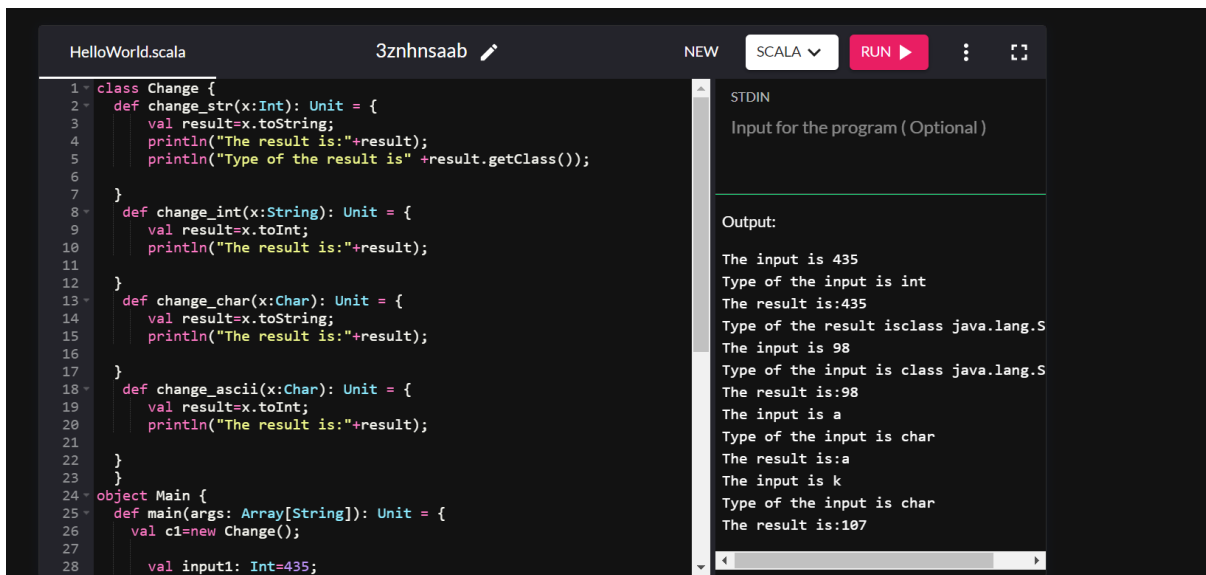
```

Output:

```

The input is 435
Type of the input is int
The result is:435
Type of the result isclass java.lang.String
The input is 98
Type of the input is class java.lang.String
The result is:98
The input is a
Type of the input is char
The result is:a
The input is k
Type of the input is char
The result is:107

```



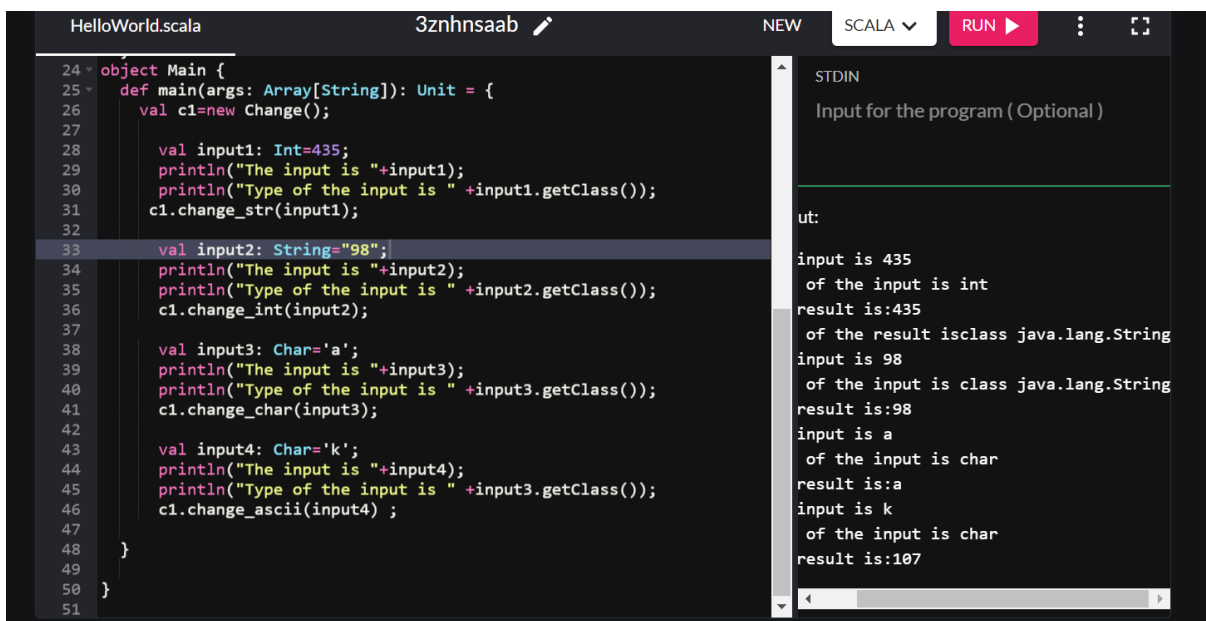
```
1 class Change {
2   def change_str(x:Int): Unit = {
3     val result=x.toString;
4     println("The result is:"+result);
5     println("Type of the result is" +result.getClass());
6   }
7
8   def change_int(x:String): Unit = {
9     val result=x.toInt;
10    println("The result is:"+result);
11  }
12
13  def change_char(x:Char): Unit = {
14    val result=x.toString;
15    println("The result is:"+result);
16  }
17
18  def change_ascii(x:Char): Unit = {
19    val result=x.toInt;
20    println("The result is:"+result);
21  }
22 }
23
24 object Main {
25   def main(args: Array[String]): Unit = {
26     val c1=new Change();
27
28     val input1: Int=435;
```

STDIN

Input for the program (Optional)

Output:

The input is 435
Type of the input is int
The result is:435
Type of the result isclass java.lang.S
The input is 98
Type of the input is class java.lang.S
The result is:98
The input is a
Type of the input is char
The result is:a
The input is k
Type of the input is char
The result is:107



```
24 object Main {
25   def main(args: Array[String]): Unit = {
26     val c1=new Change();
27
28     val input1: Int=435;
29     println("The input is "+input1);
30     println("Type of the input is " +input1.getClass());
31     c1.change_str(input1);
32
33     val input2: String="98";
34     println("The input is "+input2);
35     println("Type of the input is " +input2.getClass());
36     c1.change_int(input2);
37
38     val input3: Char='a';
39     println("The input is "+input3);
40     println("Type of the input is " +input3.getClass());
41     c1.change_char(input3);
42
43     val input4: Char='k';
44     println("The input is "+input4);
45     println("Type of the input is " +input3.getClass());
46     c1.change_ascii(input4) ;
47   }
48 }
49
50
51
```

STDIN

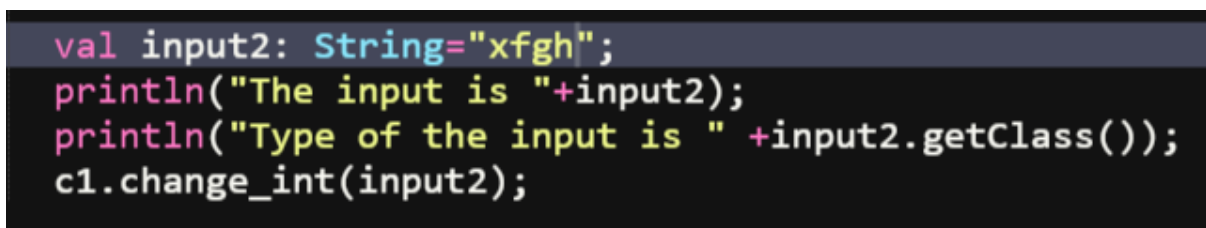
Input for the program (Optional)

ut:

input is 435
of the input is int
result is:435
of the result isclass java.lang.String
input is 98
of the input is class java.lang.String
result is:98
input is a
of the input is char
result is:a
input is k
of the input is char
result is:107

In this program, different functions are defined to convert one data type to another in class Change. The input is stored in a variable within the program and an appropriate function is called. These functions return the converted value as well as its type.

In the output section, the input value, its type, the converted value and its data type are visible.



```
val input2: String="xfgh";
println("The input is "+input2);
println("Type of the input is " +input2.getClass());
c1.change_int(input2);
```

If we try to pass a non-integer value to the string to int function, the program throws the following exception:

The input is xfgh

Type of the input is class java.lang.String

```
java.lang.NumberFormatException: For input string: "xfgh"
    at
    java.base/java.lang.NumberFormatException.forInputString(NumberFormatException.java:67)
        at java.base/java.lang.Integer.parseInt(Integer.java:668)
        at java.base/java.lang.Integer.parseInt(Integer.java:786)
        at scala.collection.StringOps$.toInt$extension(StringOps.scala:915)
        at Change.change_int(HelloWorld.scala:9)
        at Main$.main(HelloWorld.scala:36)
        at Main.main(HelloWorld.scala)
        at java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at
    java.base/jdk.internal.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)
        at
    java.base/jdk.internal.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.base/java.lang.reflect.Method.invoke(Method.java:568)
        at
    scala.reflect.internal.util.RichClassLoader$.anonfun$run$extension$1(ScalaClassLoader.scala:101)
        at scala.reflect.internal.util.RichClassLoader$.run$extension(ScalaClassLoader.scala:36)
        at scala.tools.nsc.CommonRunner.run(ObjectRunner.scala:30)
        at scala.tools.nsc.CommonRunner.run$(ObjectRunner.scala:28)
        at scala.tools.nsc.ObjectRunner$.run(ObjectRunner.scala:45)
        at scala.tools.nsc.CommonRunner.runAndCatch(ObjectRunner.scala:37)
        at scala.tools.nsc.CommonRunner.runAndCatch$(ObjectRunner.scala:36)
        at scala.tools.nsc.AbstractScriptRunner.runCompiled(ScriptRunner.scala:168)
        at scala.tools.nsc.AbstractScriptRunner$.anonfun$runScript$1(ScriptRunner.scala:177)
        at
    scala.tools.nsc.AbstractScriptRunner$.anonfun$withCompiledScript$9(ScriptRunner.scala:154)
        at scala.tools.nsc.util.package$.waitForThreads(package.scala:55)
        at scala.tools.nsc.AbstractScriptRunner.runScript(ScriptRunner.scala:151)
        at scala.tools.nsc.MainGenericRunner.runTarget$1(MainGenericRunner.scala:74)
        at scala.tools.nsc.MainGenericRunner.run$1(MainGenericRunner.scala:91)
        at scala.tools.nsc.MainGenericRunner.process(MainGenericRunner.scala:103)
        at scala.tools.nsc.MainGenericRunner$.main(MainGenericRunner.scala:108)
        at scala.tools.nsc.MainGenericRunner.main(MainGenericRunner.scala)
```

As long as we pass suitable inputs, the program gives correct output.

Github link:

<https://github.com/niranjana628/Scala-Programming>

