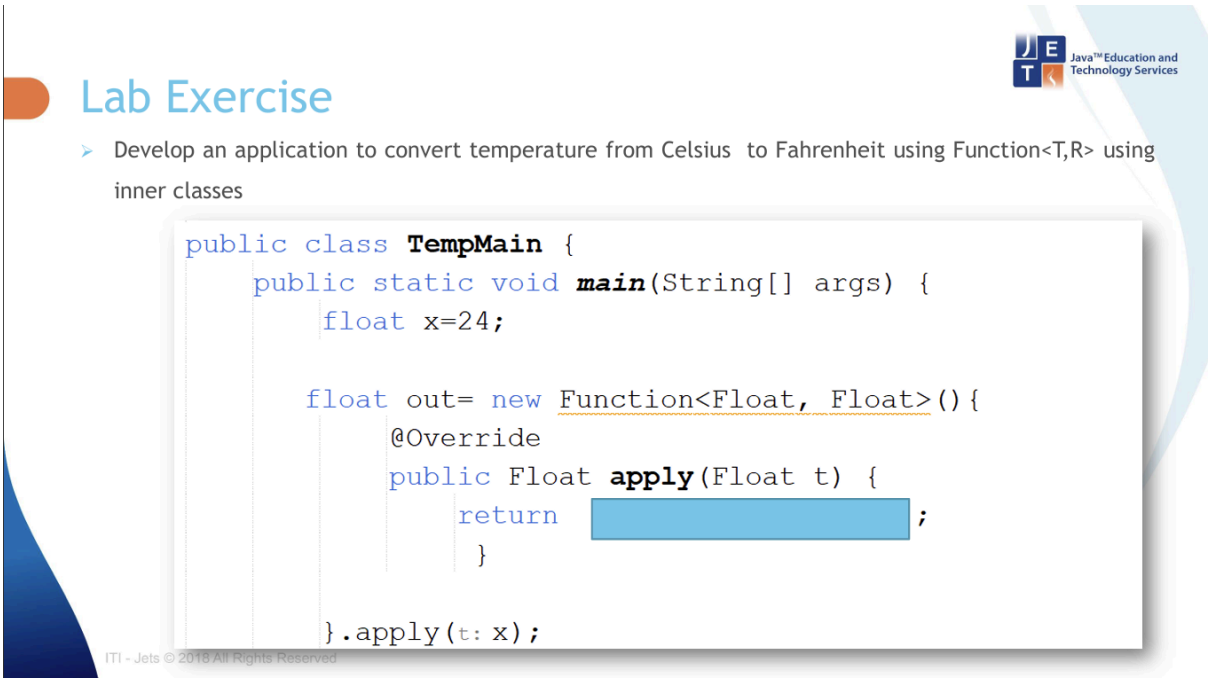
Java lab 4

Lab Exercises

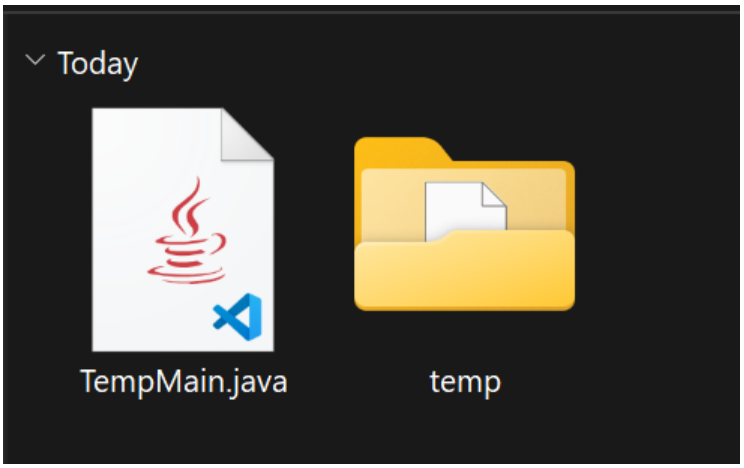
lab1:
Develop an application to convert temperature from Celsius to Fahrenheit using Function using inner classes



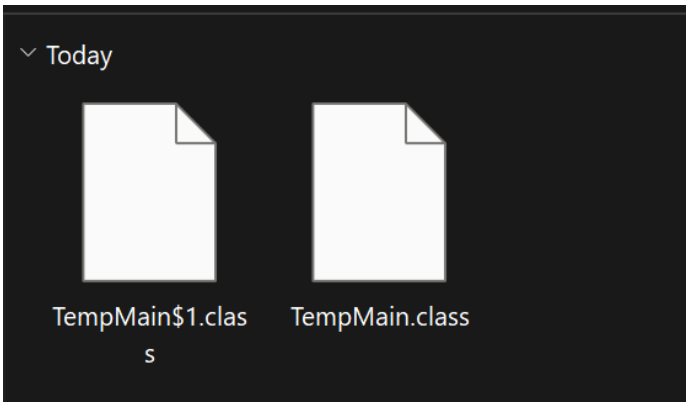
Open notepad write this
in TempMain.java:

```
package temp;
import java.util.function.Function;
public class TempMain {
        public static void main(String[] args){
                float x = 98.6f;
                float result = new Function<Float,Float>(){
                        @Override
                        public Float apply (Float t){
                                return ((t - 32) * (5.0f/9));
                        }
                }.apply(x);
                System.out.println("Temp is "+x+" F or "+ result +" F");
```

}}



And Inside the temp package:




```
C:\Users\nadam\Downloads\open source\Java\day4\lab\lab4.1>javac -d . TempMain.java

C:\Users\nadam\Downloads\open source\Java\day4\lab\lab4.1>java temp.TempMain
Temp is 98.6 F or 37.0 F
```

lab2:

- **Create your own exception class.**
- **Write down two other classes:**
- **the first will contain three methods throwing your newly created exception class**
- **The second class will be calling the methods that throws exceptions using the try-catch-finally block.**

Open notepad write this
in HandleException**.java:**

```
public class HandleException {

	public static void main(String[] args){

		ThrowingClass ex = new ThrowingClass();

		//fact
		try {
			ex.fact(-5);
	} catch (MyOwnException e) {
			System.out.println("Caught exception: " +
e.getMessage());
	} finally {
			System.out.println("Finally executed for fact");
	}

		//checkString
		try {
			ex.checkString("Hi");
	} catch (MyOwnException e) {
			System.out.println("Caught exception: " +
e.getMessage());
	} finally {
			System.out.println("Finally executed for checkString");
	}
```

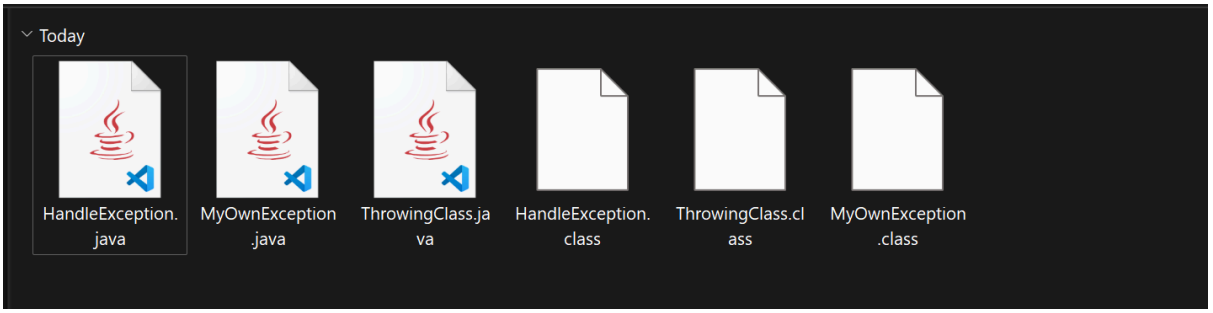
```
		//checkNonZero
		try {
			ex.checkNonZero(0);
	} catch (MyOwnException e) {
			System.out.println("Caught exception: " +
e.getMessage());
	} finally {
			System.out.println("Finally executed for
checkNonZero");
	}
	}
}
```

in ThrowingClass **.java:**

```
public class ThrowingClass {
	//num negative
	public int fact(int num) throws MyOwnException{
		if(num < 0) throw new MyOwnException("we can't Cal
factorial of Negative number:(");
		if(num == 0 || num == 1 ) return 1;
		return num*fact(num-1);
	}
	//string empty
	public void checkString(String str) throws MyOwnException{
		if(str.isEmpty()) throw new MyOwnException("we shouldn't
have empty string");
		System.out.println("Our String is : " + str);
	}
	//num = 0
	public void checkNonZero(int num) throws MyOwnException{
		if(num == 0) throw new MyOwnException("we shouldn't have
zero number");
		System.out.println("Our number is : " + num);
	}
}
```

in MyOwnException **.java:**

```
public class MyOwnException extends Exception {
	public MyOwnException(String ex)
	{
		super(ex);
	}
}
```




```
C:\Users\nadam\Downloads\open source\Java\day4\lab\lab4.2>javac HandleException.java

C:\Users\nadam\Downloads\open source\Java\day4\lab\lab4.2>java HandleException
Caught exception: we can't Cal factorial of Negative number:(
Finally executed for fact
Our String is : Hi
Finally executed for checkString
Caught exception: we shouldn't have zero number
Finally executed for checkNonZero

C:\Users\nadam\Downloads\open source\Java\day4\lab\lab4.2>
```

Lab3 :-

• Create a base class named Shape that contains one abstract method draw().

• Create two concrete classes (Rectangle and Circle) that extend Shape

• Create a test class that defines a method that accepts a list that contains only child classes of shape

• Test your method by creating two ArrayList of Rectangle and shapes and pass them to the generic method

In TestMain.java

```
import java.util.ArrayList;

public class TestMain{

	public static void drawShapes(ArrayList<? extends Shape> shapes) {
		for (Shape shape : shapes) {
		shape.draw();
		}
	}

	public static void main(String[] args)
	{
		ArrayList<Rect> rects = new ArrayList<>();
		rects.add(new Rect());
	rects.add(new Rect());

		ArrayList<Shape> shapes = new ArrayList<>();
	shapes.add(new Rect());
	shapes.add(new Circle());

		System.out.println("Drawing Rectangles:");
	drawShapes(rects);

	System.out.println("\nDrawing Shapes:");
	drawShapes(shapes);
	}

}
```

In Circle **.java**

```
public class Circle extends Shape{
	public void draw(){
		System.out.println("----------------------------------------");
```

```
                System.out.println("Drawing Circle......");
                System.out.println("----------------------------------------");
        }
}
```

In Rect **.java**

```
public class Rect extends Shape{
        public void draw(){
                System.out.println("----------------------------------------");
                System.out.println("Drawing Rectangle......");
                System.out.println("----------------------------------------");
        }
}
```

In Shape**.java**

```
abstract class Shape{
        public abstract void draw();
}
```

```
C:\Users\nadam\Downloads\open source\Java\day4\lab\lab4.3>javac -d . TestMain.java

C:\Users\nadam\Downloads\open source\Java\day4\lab\lab4.3>java TestMain
Drawing Rectangles:
----------------------------------------
Drawing Rectangle......
----------------------------------------
----------------------------------------
Drawing Rectangle......
----------------------------------------

Drawing Shapes:
----------------------------------------
Drawing Rectangle......
----------------------------------------
----------------------------------------
Drawing Circle......
----------------------------------------

C:\Users\nadam\Downloads\open source\Java\day4\lab\lab4.3>
```

Lab4 :-

- **Create a generic interface that could be used to represent complex numbers**
- **Create some generic methods that represent basic arithmetic operation on complex**

```
C:\Users\nadam\Downloads\open source\Java\day4\lab\lab4.4>java ComplexMain
c1: 3.04 + 4.06i
c2: 1.88 + 2.09i
Addition: 4.92 + 6.15i
Subtraction: 1.16 + 1.97i
Multiplication: -2.76 + 13.98i
Division: 1.80 + 0.16i

C:\Users\nadam\Downloads\open source\Java\day4\lab\lab4.4>
```