RunTimeStackMechansim

For every thread in java language, jvm create a seperate stack at the time of Thread creation.
All method calls performed by this thread will be stored in the stack. Every entry in the stack
is called "StackFrame/Activation Record".

```
     main() => doStuff() => doMoreStuff()
```

eg::

```
class Demo{
      public static void main(String[] args){
            doStuff();
      }
      public static void doStuff(){
            doMoreStuff();
      }
      public static void doMoreStuff(){
            System.out.println("hello");
      }
}
```

output:: Hello

Default Exception handling

```
class Demo{
      public static void main(String[] args){
            System.out.println("Entering main");
            doStuff();
            System.out.println("Exiting main");

      }
      public static void doStuff(){
            System.out.println("Entering doStuff");
            doMoreStuff();
                System.out.println("Exiting doStuff");
      }
      public static void doMoreStuff(){
            System.out.println("Entering doMoreStuff");
            System.out.println(10/0);
            System.out.println("Exiting doMoreStuff");
      }
}
```

Output::

```
Entering main
Entering doStuff
Entering doMoreStuff
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at TestApp.doMoreStuff(TestApp.java:14)
        at TestApp.doStuff(TestApp.java:9)
        at TestApp.main(TestApp.java:4)
```

As noticed in the above example in the method called doMoreStuff(), exception is raised.
  =>When exception is raised inside any method, that method is responsible for creating the
    Exception object will the following details
         Name of the exception::java.lang.ArithmeticException

```
         Description of exception::/ by zero
         location/stacktrace::
```

1. This Exception object will be handed over to jvm, now jvm will check whether the method has
   the handling code or not,if it is not available then that method will be abnormally terminated.
   since it is a method, it will propogate the exception object to caller method.

2. Now jvm will check whether the caller method is having the code of caller method or not
   if it is not available, then that method will be abnormally terminated.

3. Simiar way if the exception object is propogated to main(), jvm will check whether the main()
   is having a code for handling or not, if not then the exception object will be propogated to
   JVM by terminating the main().

4. JVM now will handover the exception object to "Default exception handler", the duty of
   "default exception handler" is to just print the exception object details in the following way

```
    Exception in thread "main" java.lang.ArithmeticException:/ by zero
                   at TestApp.doMoreStuff
                   at TestApp.doStuff
                   at TestApp.main
```

```
class Demo{
      public static void main(String[] args){
            System.out.println("Entering main");
            doStuff();
            System.out.println("Exiting main");

      }
      public static void doStuff(){
            System.out.println("Entering doStuff");
            doMoreStuff();
            System.out.println(10/0");
                System.out.println("Exiting doStuff");
      }
      public static void doMoreStuff(){
            System.out.println("Entering doMoreStuff");
            System.out.println("hello");
            System.out.println("Exiting doMoreStuff");
      }
}
Output::
Entering main
Entering doStuff
Entering doMoreStuff
Hello
Exiting doMoreStuff
Exception in thread "main" java.lang.ArithmeticException : / by zero
               at TestApp.doStuff()
                  TestApp.main()
```

eg#3

```
class TestApp{
      public static void main(String[] args){
            System.out.println("Entering main");
            doStuff();
            System.out.println(10/0);
            System.out.println("Exiting main");

      }
      public static void doStuff(){
            System.out.println("Entering doStuff");
            doMoreStuff();
            System.out.println("hiee");
                System.out.println("Exiting doStuff");
      }
      public static void doMoreStuff(){
            System.out.println("Entering doMoreStuff");
            System.out.println("hello");
            System.out.println("Exiting doMoreStuff");
      }
}
```

Output

```
Entering main
Entering doStuff
Entering doMoreStuff
hello
Exiting doMoreStuff
hiee
Exiting doStuff
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Test.main(Test.java:9)
```