DAILY ASSIGNMENT – 10/08/2021

JAVA EXCEPTIONS

When executing Java code, different errors can occur: coding errors made by the programmer, errors due to wrong input, or other unforeseeable things. When an error occurs, Java will normally stop and generate an error message. The technical term for this is: Java will throw an exception (throw an error).

JAVA TRY AND CATCH

The try statement allows you to define a block of code to be tested for errors while it is being executed.

The catch statement allows you to define a block of code to be executed, if an error occurs in the try block.

SYNTAX:-

```
try {
      //  Block of code to try
    }
    catch(Exception e) {
      //  Block of code to handle errors

    }
```

EXAMPLE :-

```
public class Main {
      public static void main(String[ ] args) {
        try {
          int[] myNumbers = {1, 2, 3};
          System.out.println(myNumbers[10]);
        } catch (Exception e) {
          System.out.println("Something went
wrong.");
        }
      }
    }
```

FINALLY

The finally statement lets you execute code, after try...catch, regardless of the result:

SYNTAX:-

```
try {
      //  Block of code to try
    }
    catch(Exception e) {
      //  Block of code to handle errors
    } finally

      // Block of code
```

EXAMPLE :-

```
import java.io.File;
import java.io.FileWriter;

public class Test{

    public static void main(String [] args) throws
Exception {
        File f = new File("myFile.txt");

        try (FileWriter fw = new FileWriter(f);) {
            fw.write("bye");
        }

        System.out.println("yes this will print");

        try {
            int b = 10, c = 0;
            int a = b / c;
        } catch (Exception e) {
            System.out.println("Cannot divide by 
zero");
        } finally {
            System.out.println("used for closing and 
releasing objects");
        }
    }
}
```

JAVA THREADS

Threads allows a program to operate more efficiently by doing multiple things at the same time. Threads can be used to perform complicated tasks in the background without interrupting the main program.

There are two ways to create a thread:

1. By extending Thread class
2. By implementing Runnable interface.

Thread class provide constructors and methods to create and perform operations on a thread. It can be created by extending the Thread class and overriding its run() method

SYNTAX:-

```
public class Main extends Thread {
          public void run() {
            System.out.println("This code is running in a thread");
          }
        }
```

Another way to create a thread is to implement the Runnable interface

SYNTAX:-

```
public class Main implements Runnable {
          public void run() {
            System.out.println("This code is running in a thread");
          }
        }
```

RUNNING THREADS

If the class extends the Thread class, the thread can be run by creating an instance of the class and call its start() method

SYNTAX :-

```
public class Main extends Thread {
          public static void main(String[] args) {
            Main thread = new Main();
            thread.start();
            System.out.println("This code is outside of the thread");
          }
          public void run() {
            System.out.println("This code is running in a thread");
          }
        }
```

If the class implements the Runnable interface, the thread can be run by passing an instance of the class to a Thread object's constructor and then calling the thread's start() method

SYNTAX :-

```
public class Main implements Runnable {
        public static void main(String[] args) {
          Main obj = new Main();
          Thread thread = new Thread(obj);
          thread.start();
          System.out.println("This code is outside of the thread");
        }
        public void run() {
          System.out.println("This code is running in a thread");
        }
      }
```

EXAMPLE :-

```
public class Test {
    //synchronous - one after another - process
    public static void main(String[] args) throws
Exception {
        A a = new A();
        a.start();

        B b = new B();
        Thread th = new Thread(b);
        th.start();
    }

}

//multi level inheritance
```

```java
class A extends Thread {
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println("----a---" + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {

            }
        }
    }
}

class B implements Runnable{
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.err.println("----b---" + i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {

            }
        }
    }
}

class C{

}
```