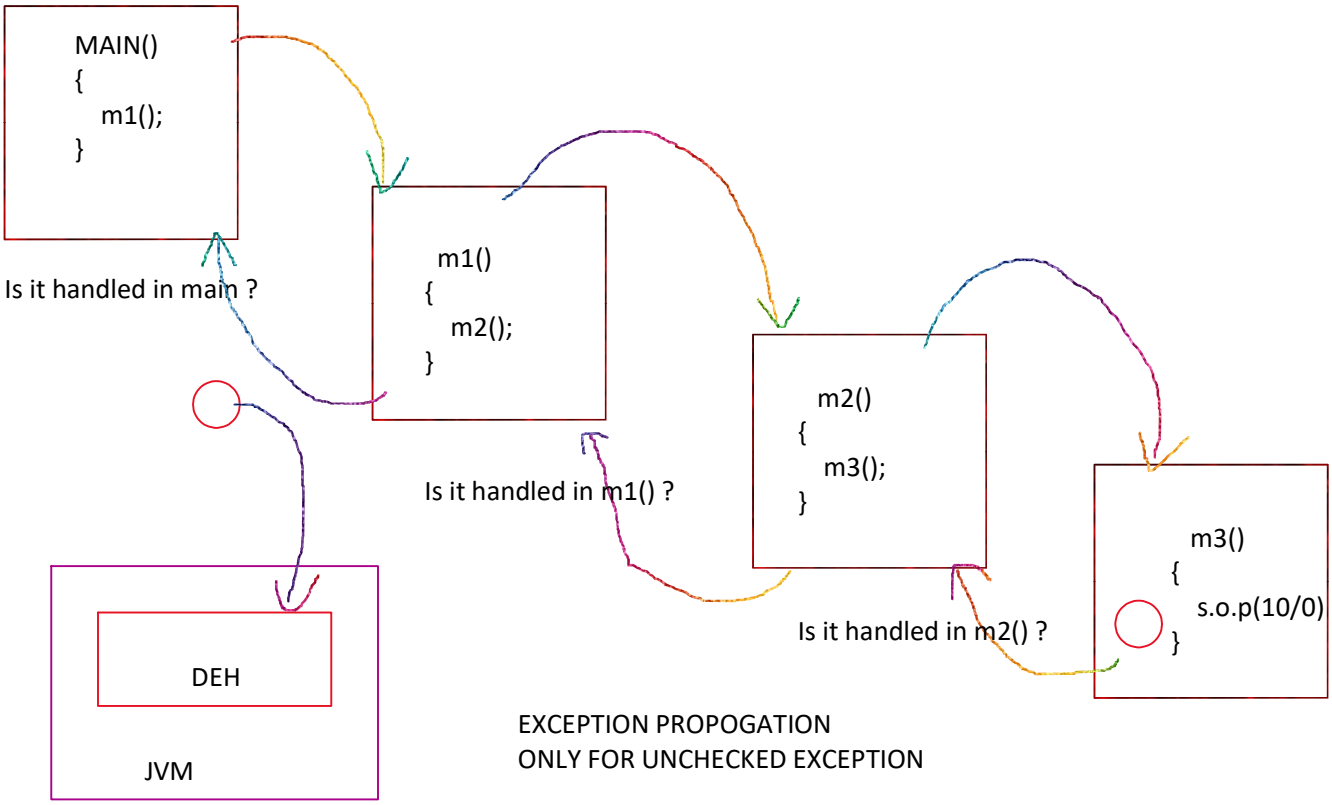
DAY-44

23 August 2023 15:44

EXCEPTION PROPAGATION

- THE ABILITY OF AN EXCEPTION OBJECT TO PROPOGATE BACK TO ITS CALLER IS CALLED AS EXCEPTION PROPOGATION
- WHENEVER, EXCEPTION OCCURS, JVM WILL CREATE RESPECTIVE TYPE OF EXCEPTION OBEJCT AND THROWS IT BACK TO PROGRAM.
- IF USER HAS NOT WRITTEN ANY CODE TO HANDLE THE EXCEPTION OBJECT THEN , THIS OBJECT WILL PROPOGATE BACK TO CALLER AND FINALLY TO JVM WHICH WILL HANDLE IT TO DEFAULT EXCEPTION HANDLER
- THIS AUTOMATIC PROPAGATION CAPABILITY IS THERE ONLY FOR UNCHECKED EXCEPTION.



- IF EXCEPTION IS NOT HANDLED

```
package exc;

public class A {
```

```
	public static void main(String[] args)
	{
		System.out.println("main starts");
		m1();
		System.out.println("main ends");

	}
	public static void m1()
	{
		System.out.println("m1 starts");
		m2();
		System.out.println("m1 ends");
	}

	public static void m2()
	{
		System.out.println("m2 starts");
		m3();
		System.out.println("m2 ends");
	}

	public static void m3()
	{
		System.out.println("m3 starts");
		System.out.println(10/0);
		System.out.println("m3 ends");
	}


}
```

- IF EXCEPTION IS HANDLED IN M3()

```
package exc;

public class A {

	public static void main(String[] args)
	{
		System.out.println("main starts");
		m1();
		System.out.println("main ends");

	}
	public static void m1()
	{
		System.out.println("m1 starts");
		m2();
		System.out.println("m1 ends");
	}

	public static void m2()
	{
```

```
        System.out.println("m2 starts");
        m3();
        System.out.println("m2 ends");
    }

    public static void m3()
    {
        System.out.println("m3 starts");

        try
        {
            System.out.println(10/0);
        }
        catch (Exception e)
        {
            System.out.println(e.getMessage());
        }

        System.out.println("m3 ends");
    }


}
```

- IF EXCEPTION IS HANDLED IN M2()

```
package exc;

public class A {

    public static void main(String[] args)
    {
        System.out.println("main starts");
        m1();
        System.out.println("main ends");

    }
    public static void m1()
    {
        System.out.println("m1 starts");
        m2();
        System.out.println("m1 ends");
    }

    public static void m2()
    {
        System.out.println("m2 starts");

        try
        {
            m3();
        }
        catch (Exception e)
        {
```

```
                System.out.println(e.getMessage());
            }

            System.out.println("m2 ends");
        }

        public static void m3()
        {
            System.out.println("m3 starts");
            System.out.println(10/0);
            System.out.println("m3 ends");
        }


    }
```

THROWS KEYWORD

- IT IS USED TO HANDLE THE CHECKED EXCEPTION
- THROWS KEYWORD USED TO INFORM THE CALLER ABOUT THE CHECKED EXCEPTION.
- SINCE, THE CHEKED EXECPTION DOESN'T HAVE THE CAPABILITY TO PROPAGATE BY THEMSELVES THAT IS WHY THEY MUST BE EITHER HANDLES THERE ITSELF OR THE USER HAS TO INFORM THE CALLER USING THROWS KEYWORD.
- USING THE THROWS KEYWORD, WE CAN INFORM THE CALLER ABOUT MULTIPLE EXCEPTIONS AT A TIME AND THE TROWS KEYWORD WILL ALWAYS BE USED IN METHOD DECLARATION.
    - EG : public static void m1() throws SQLException
- USING THROWS KEYOWRD , WE CAN INFORM CALLER ABOUT MULTIPLE CHECKED EXCEPTIONS AT A TIME.

```
Public static void main(String[] args) throws Exception
{
   m1();
}

Public static void m1() throws Exception
{
   m2();
}


Public static void m2() throws Exception
{
   m3();
}


Public static void m3() throws Exception
{
   throw new IOException();
}
```

*****WHAT IS THE DIFFERENCE BETWEEN THROW AND THROWS

THROW	THROWS
IT IS USED TO THROW BOTH CHECKED AND UNCHECKED EXCEPTIONS	IT IS USED TO INFORM THE CALLER ABOUT THE CEHCKED EXCEPTIONS AND ALSO HELPS CHECKED EXCEPTION OBJECT TO PROPOGATE BACK TO CALLER.
IT IS USED INSIDE METHOD DEFINITION	IT IS USED IN METHOD DECLARATION
USING THROW, WE CAN THROW ONLY ONE EXCEPTION OBJECT AT A TIME	USING THROWS, WE CAN INFORM CALLER ABOUT MULTIPLE EXCEPTION AT A TIME

*****DIFFERENCE BETWEEN CHECKED AND UNCHEKCED EXCEPTION

UNCHECKED EXECPTION	CHECKED EXCEPTION
UNCHECKED EXCEPTIONS ARE KNOWN AT RUNTIME AND ALSO OCCUR AT RUNTIME	CHECKED EXCEPTIONS ARE KNOWN AT COMPILE TIME AND OCCUR AT RUN TIME
UNCHECKED EXCEPTION HAVE AUTOMATIC PROPOGATION CAPABILITY	CHECKED EXCEPTIONS DO NOT HAVE AUTOMATIC PROGATION CAPABILITY
THROWS KEYWORD NOT REQUIRED	THROWS KEYWORD REQUIRED TO INFORM THE CALLER ABOUT CHECKED EXCEPTIONS.