DAY-42

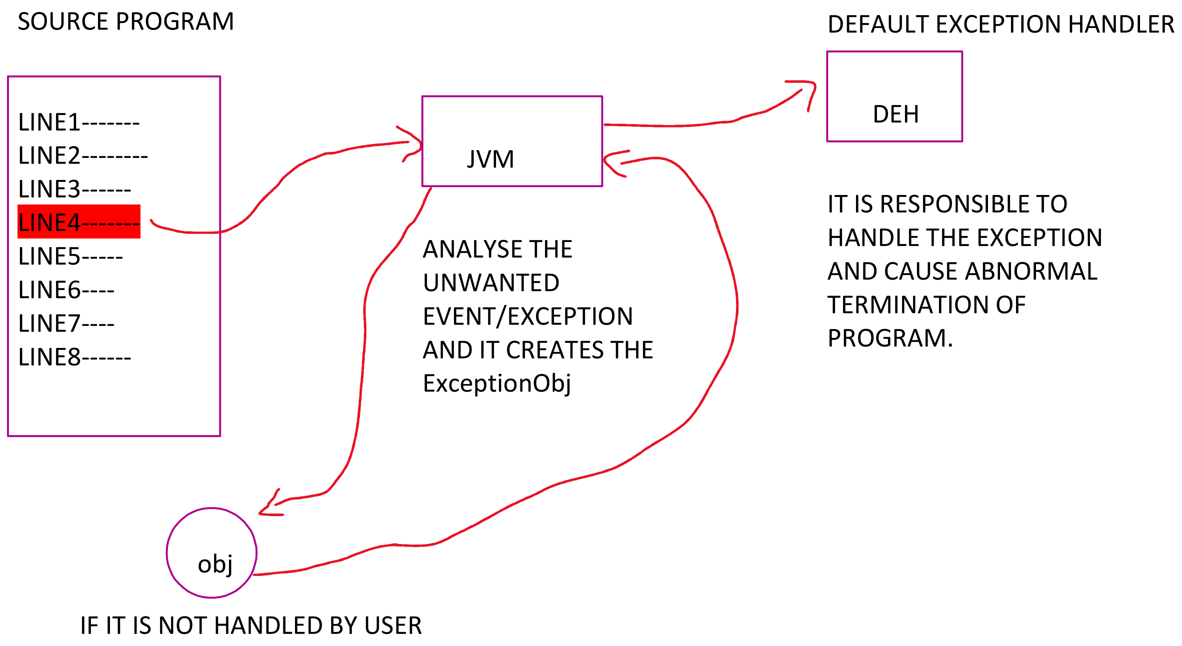
21 August 2023 15:38

ERROR HANDLING / EXCEPTION HANDLING

ERROR

ERRORS ARE SOMETHING UNEXPECTED WHICH OCCUR DURING COMPILE TIME OR RUN TIME

EX : SYNTAX ERROR , LOGICAL ERROR



SOURCE PROGRAM
DEFAULT EXCEPTION HANDLER
LINE1-------
LINE2--------
LINE3------
LINE4-------
LINE5-----
LINE6----
LINE7----
LINE8------
JVM
DEH
ANALYSE THE UNWANTED EVENT/EXCEPTION AND IT CREATES THE ExceptionObj
IT IS RESPONSIBLE TO HANDLE THE EXCEPTION AND CAUSE ABNORMAL TERMINATION OF PROGRAM.
obj
IF IT IS NOT HANDLED BY USER

IF AN EXECPTION OCCURS OR ERROR OCCURS IN THE PROGRAM THEN IT IS ANALYZED BY THE JVM AND IT CREATES ExceptionObj AND SEND IT BACK TO THE PROGRAM TO HANDLE IT. IF THE ExceptionObj IS NOT HANDLED IN THE PROGRAM THEN IT IS SENT BACK TO THE JVM AND JVM FORWARDS IT TO DEH FOR HANDLING IT. DEH HANDLES THE EXCEPTION BY TERMINATION THE PROGRAM ABNORMALLY AND BY GIVING THE MESSAGE ON THE SCREEN.

EXCEPTION

IT IS UNEXPECTED OR UNWATED EVEN WHICH LEADS TO ABNORMAL TERMINATION OF THE PROGRAM .

OR

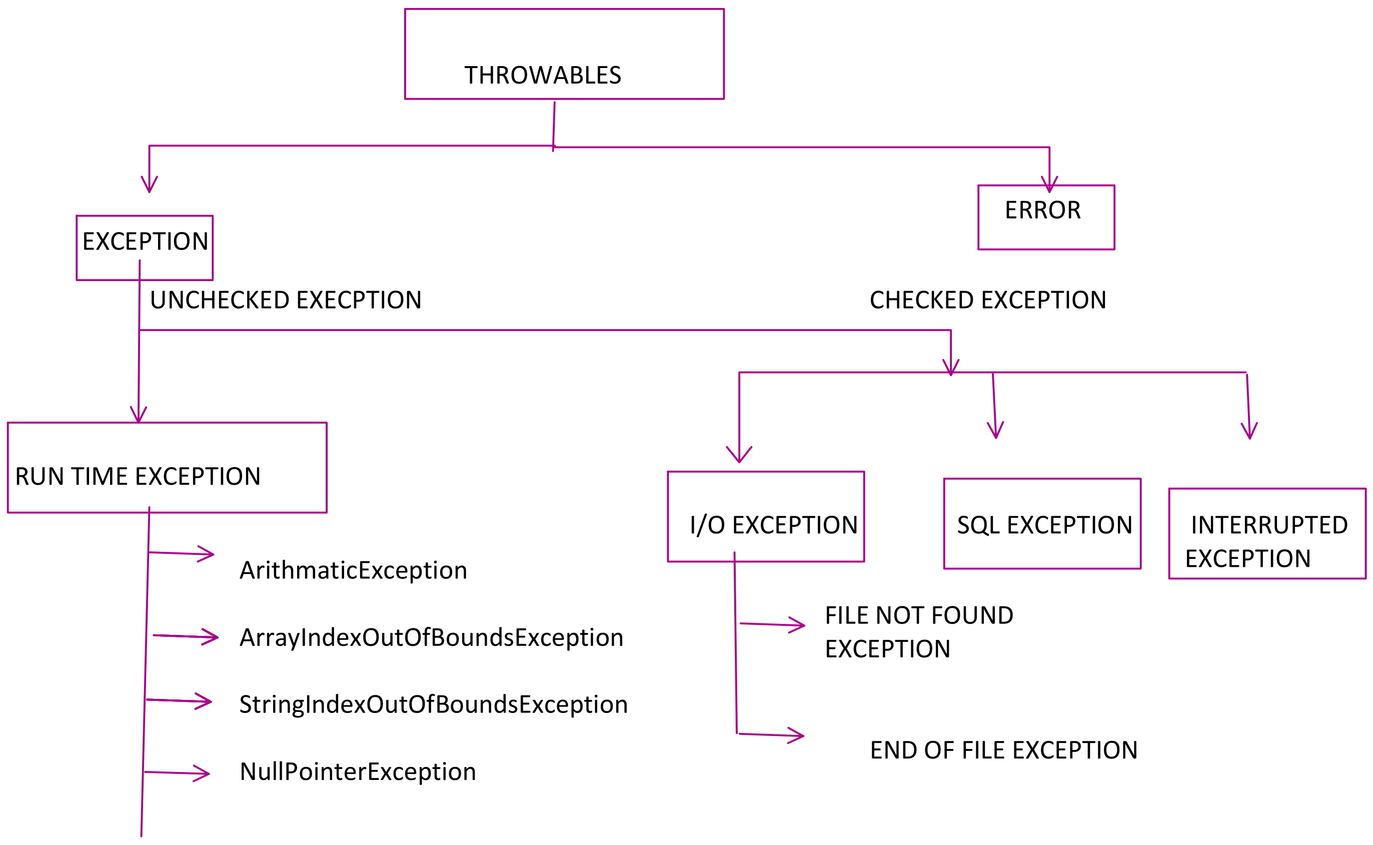
IT IS AN UNEXPECTED OR UNWANTED EVENT TRIGGERED/ENCOUNTERED BY JVM WHICH LEADS THE JVM TO FORCEFULLY TERMINATE THE PROGRAM.

DEFAULT EXCEPTION HANDLER

- WHENVER EXCEPTION OCCURS IN THE PROGRAM AND THE USER HAS NOT WRITTEN ANY CODE TO HANDLE

EXCEPTION THEN, THE JVM WILL HANDLE THIS EXCEPTION BY CALLING THE DEFAULT EXECPTION HANDLER.

- WHENEVER EXECPTION OCCURS IN THE PROGRAM, THE JVM WILL ANALYZE IT CREATE REPECTIVE TYPE OF EXCEPTION OBJECT.
- ONCE, THE EXECPTION OBJECT IS CREATED BY JVM, IT WILL THROW THE OBJECT BACK TO PROGRAM EXPECTING THE USER WOULD HAVE WRITTEN SOME CODE TO HANDLE THE EXCEPTION OBEJCT, IF THE USER HAS NOT WRITTEN ANY CODE TO HANDLE IT, THEN THE JVM WILL TAKE BACK THAT EXCEPTION OBJECT, FROM PROGRAM AND GIVE IT TO DEFAULT EXECPTION HANDLER.
- THE DEFAULT EXECEPTION HANDLER WILL FIRST TERMINATE THE PROGRAM AND THEN DISPLAY THE MESSAGE OR INFORMATION WITH RESPECT TO EXECPTION.
- WHENEVER THE DEH WILL HANDLE EXCEPTION , THE PROGRAM WILL BE TERMINATED i.e, THE LINES OF CODE WRITTEN AFTER THE EXCEPTION LINE WILL NEVER GET EXECUTED.



```
package exc;

public class ArrayEx {

	public static void main(String[] args)
	{
		int x = 1;
		int y = 0;

		System.out.println("Good Morning");
		System.out.println(x/y); //causing exception
		System.out.println("hi");
		System.out.println("Good Evening");
		System.out.println("BitsQ");
```

```
        }

}
```

- TO OVERCOME THIS, IF THE USER WANTS TO TERMINATE THE PROGRAM NORMALLY i.e, ALL THE STATEMENTS AFTER THE EXCEPTION LINE MUST BE EXECUTED, TEN THE USER MUST HANDLE THE EXCEPTION OBEJCT IN TE PROGRAM EXPLICITLY. IT CAN BE DONE BY HELP OF.

    i. TRY
    ii. CATCH
    iii. TROW
    iv. TROWS
    v. FINALLY

<u>TRY AND CATCH</u>

- TRY BLOCK CONTAINS SUSPECTED LINES OF CODE
- ONCE EXECPTION OCCURS ONLY THEN CORRSPONDING CATCH BLOCK WILL EXECUTE , ELSE CATCH WILL NOT EXECUTE.
- JUST BECAUSE YOU HAVE WRITTEN TRY BLOCK DOESN'T MEAN EXCEPTION HAS TO OCCURS.
- ONCE CONTROL JUMPS **TRY** TO CATCH IT WILL NEVER GO BACK TO **TRY** AGAIN, HENCE ALL THE LINES WRITTEN IN **TRY** AFTER EXECPTION LINE WON'T BE EXECUTED.

```
TRY
{
-------------- 1
__-----------2      CAUSING EXCEPTION
-------------3

}

CATCH ()
{
-----------4
-----------5
-----------6

}
```

- TRY CATCH BLOCK : USUALLY WE WRITE THOSE LINES OF CODE WHICH MIGHT THROW EXCEPTION INSIDE TRY BLOCK.
- ONCE EXCEPTION OCCURS IN THE TRY BLOCK , THE CORRESPNDING EXCEPTION OBJECT WILL BE CREATED AND IT WILL BE THROWN OUT OF TRY BLOCK.
- THE USER SHOULD DEFINE THE CORRSPONDING CATCH BLOCK TO CATCH THE EXCEPTION OBJECT WHICH IS THROWN FROM THE TRY BLOCK
- WE WRITE CODE TO HANDLE EXECPTION INSIDE THE CATCH BLOCK.

THROWABLE

Method
getMessage();

EXCEPTION

overridding

RUN TIME EXCEPTION

1. ArithmeticException
2. ArrayIndexOutOfBoundsException
3. StringIndexOutOfBoundsException
4. NullPoitnerException

```
package exc;

public class ArrayEx {

	public static void main(String[] args)
	{
		int x = 1;
		int y = 0;

		System.out.println("Good Morning");

		try
		{
			System.out.println("entering try block");
			System.out.println(x/y); //causing exception
			System.out.println("exiting try block");
		}
		catch(ArithmeticException e)
		{
			System.out.println("entering catch block");
			System.out.println(e.getMessage());
			System.out.println("exiting catch block");
		}
		System.out.println("hi");
		System.out.println("Good Evening");
		System.out.println("BitsQ");

	}

}
```