02-07-2024

-> package sruja;

• Trace . java

public class Demo {

    public static void main (String arC])
    {
        try
        {
            int a = Integer. parseInt (arCo]);
            int b = Integer . parseInt (arC1]);
            int c = a/b;
            System . out . println (c);
        }

        catch (Exception e)
        {
            e . print stack trace ();
        }

        system . out . println ("Program continued");

    }   finally
        {
            System . out . println (" Finally blocked ");
        }

    }
}

Arguments

40   2.5

Run As =>
Run Configuration

ARGUMENTS
40   2.5

OUTPUT:

Finally blocked

FINALLY BLOCK :-

It gives guarentee that this block statements are executed.

SYNTAX :-

```
finally
{
    - - -
}
```

→ package sruja

```
public class Demo {

    public static void main (String ar [ ])
    {
        try
        {
            int a = Integer. parseInt (ar[0]);
            int b = Integer. parseInt (ar[1]);
            int c = a/ b;

            System. out. println (c);
        }
        catch ( Exception e)
        {
            e. print Stack Trace();
        }
        finally
        {
            System. out. println ("Finally blocked ");
        }

        System. out. println ( "Program continued ");
    }
}
```

Finally - Block . java

o/p :-
Finally blocked
Program continued

THROWS :- It is a keyword which declares to the method that which exception to be thrown.

SYNTAX :- throws

THROW :- It is a keyword of JAVA which throws exceptional object by explicit

MULTI TASKING / MULTI PROCESSOR :- Utilizing ideal time of M-Processor is called multi tasking (OR) multi processor.

THREAD :- A thread is an independent process. A thread can be created in JAVA by 2 ways

1. By extending Thread class

2. By implementing runnable interface

New -> Class -> Thread2 -> Browse -> Thread class -> Finish

Browse -> thread java.lang package while creating a class

RIGHT CLICK -> @override
public void run () {

, run Method

RIGHT CLICK -> SOURCE -> OVERRIDE / IMPLEMENT METHODS ->

SEARCH RUN

Thread 1. java

→
```
package sreeja;

public class Thread1 extends Thread
{
    public void run()
    {
        for (int i=1; i<=10; i+=2)
        {
            System.out.println(i);
            try {
                Thread.sleep(1000);
            }
            catch (InterruptedException e)
            {
                e.printStackTrace();
            }
        }
    }
}
```

Thread 2. java

→
```
package sreeja;

public class Thread2 extends Thread
{
    public void run()
    {
        for (int i=
        {
            System.out.println(i);
```

```java
try {
    Thread.sleep(1000);
}

    catch (InterruptedException e)
    {
        e.printStackTrace();
    }
        }
    }
}
```

Thread~~Demo~~.java

→ package sreeja;

public class Demo
{
    public static void main(String arc [])
    {
        Thread1 p = new Thread1();
        Thread2 q = new Thread2();
        p.start(); q.start();
    }
}