Exception Handling

The mechanism used to continue the flow of program, even though when the exception occur is known as Exception Handling.

Java provides, **try-catch** block to achieve exception handling.

try-catch block:

1. **try :**

```
try
{
    statements ;
}
```

Note:

1. in try block programmer should write the statements which could be responsible for exceptions.
2. When exception occur :
    1. execution of try block is paused
    2. Throwable type object is created depending on the Exception raised.
    3. try will throw the reference of the object created to the catch block.

1. **catch :**

```
catch (  declare variable of Throwable type )
{
    statements ;
}
```

Note :

1. If the catch block catches the Object thrown by try only then the exception is handled, catch block gets executed and the flow of program continues.
2. if the catch block doesn't catch the Object thrown by the try block, then Exception is not handled, catch block does not get executed and the program stops abruptly.

When the exception is caught ?

1. if the object thrown and variable declared is of same type.
   example refer : **workspace/Exceptions/src/excep2/E6.java**
2. if the variable declared is a super class type of the object thrown.
   example refer : **workspace/Exceptions/src/excep2/E7.java**

   Rules :
1. we cannot use only try block, we get Compile time error.
2. the variable declared in catch should only be Throwable type, else we get compile time error.

try block with multiple catch :

A try block can have more than one catch block.

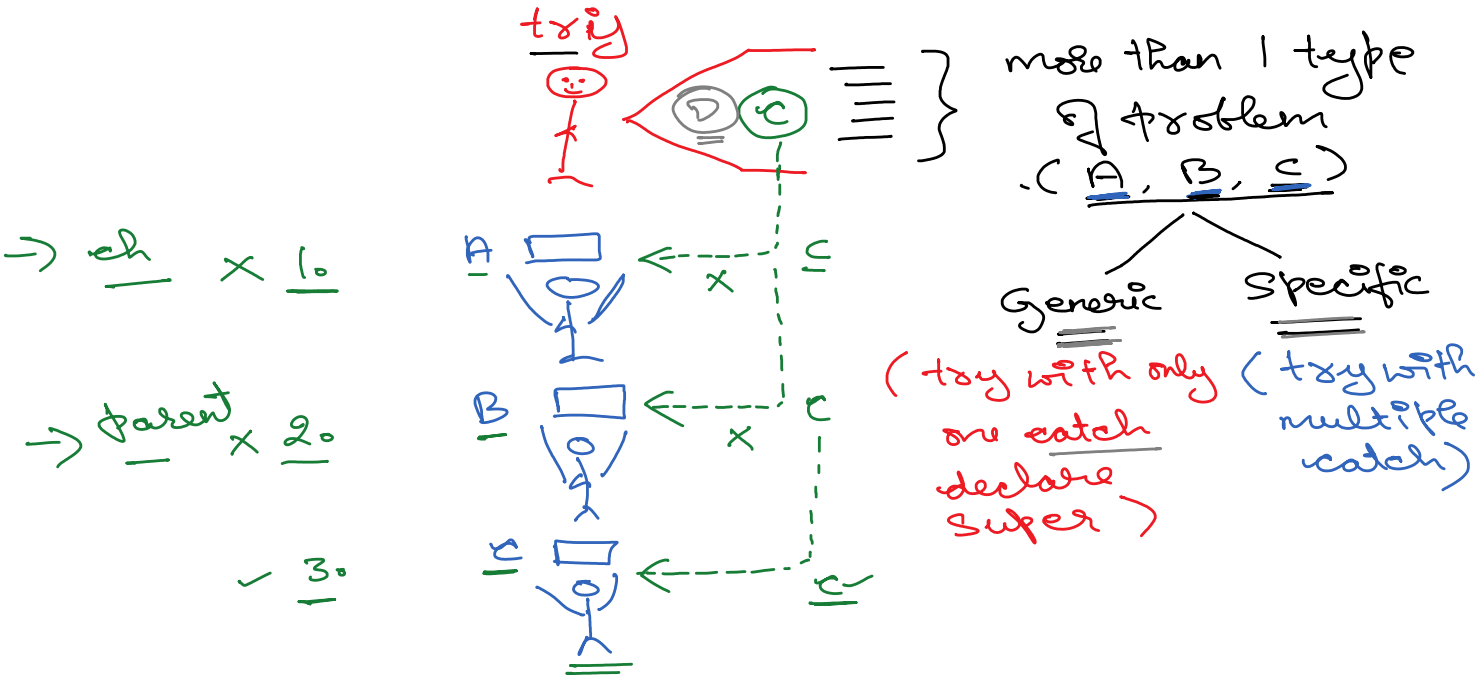
syntax:

```
try {

}
catch ( Throwable_type v1  )
{

}
    .
    .
    .
catch ( Throwable_type v1  )
{

}
```



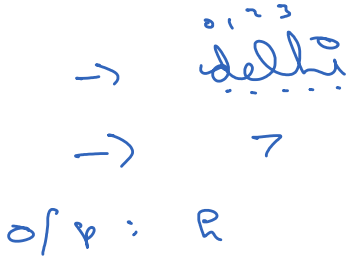
Rule to be followed for try with multiple catch :

The order of catch blocks should be maintained such that, the subclass type should be written first and then the superclass type.
Example refer : **workspace/exceptions/src/excep2/E9.java**

if superclass type is used first and then the subclass type is used then we get a compile time error.
Example refer : **workspace/exceptions/src/excep2/E10.java**

Task1 :

1. WAJP to read name of the city from the user, ask the user to enter a single digit lucky number. The program should print the character present in the city name present in that index position.
   (handle the exceptions if any required)



finally :

1. finally is a block in java, it can be used along with try or try-catch block only.
2. The statements present in finally block gets executed for all the scenarios of the try :

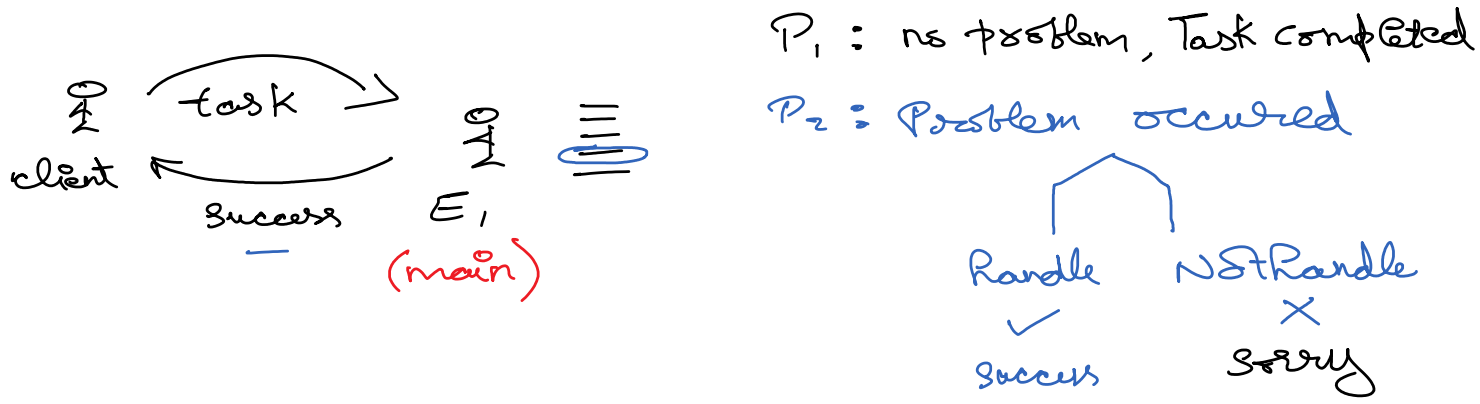
   i. Exception not occurred finally gets executed.
   ii. Exception occurred but not handled finally gets executed
   iii. Exception occurred and handled finally gets executed.

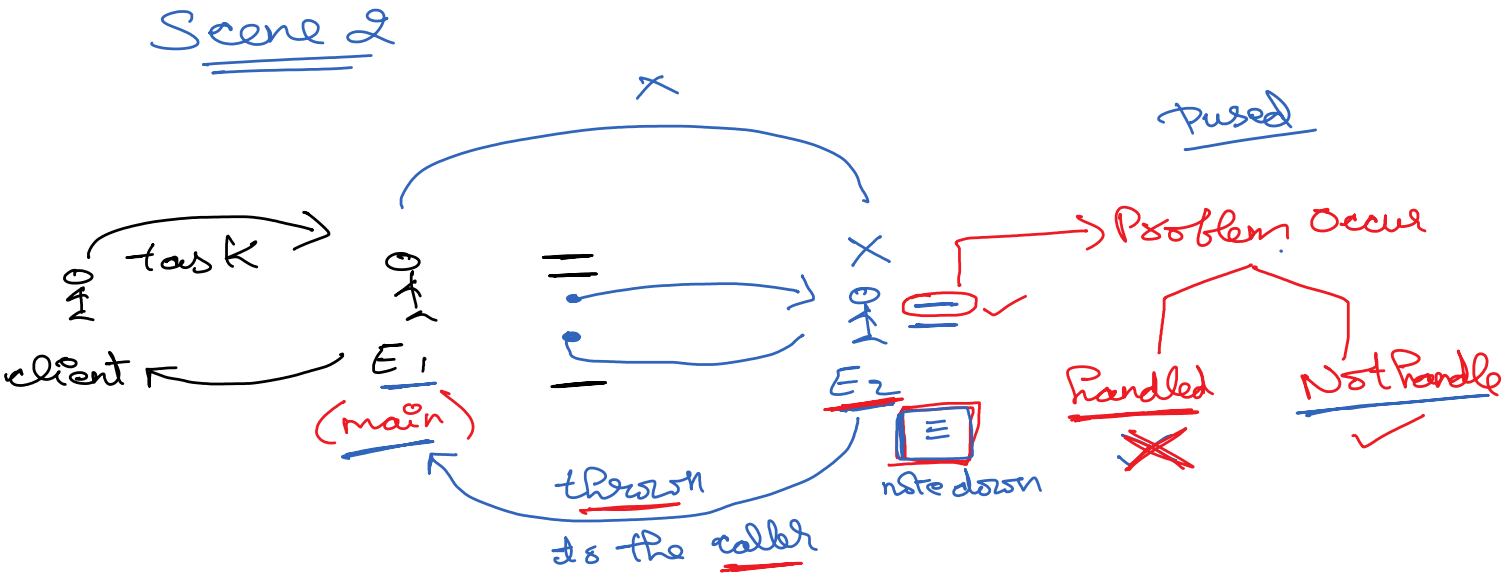
Note : When to use a finally block :

case1 : if we have any set of instructions to be executed at any cost whatever happens, at such situations we make use of finally block (example closing a file).

case2 : the programmer is not ready to handle the exception, but wants to execute a set of instructions before the program is stopped, at situation the programmer can use try-finally block.

example refer : **workspace/exceptions/src/excep2/E13.java**





Exception Object Propagation :

The movement of Exception object from called method to the calling method, when the exception is occurred and not handled is known as Exception object propagation.

Or

When an exception is occurred in a method and not handled, the Exception object is thrown to its caller, this process is known as Exception Object Propagation.

Stack Trace :

It gives the flow of Exception object, from top of the stack to the bottom of the stack.

Note :

1. Exception object propagation happens implicitly when the Exception is occurred and not handled for the **Unchecked Exceptions**.

2. For checked exception to achieve exception object propagation we have to declare the exception using throws keyword.

throws:

1. it is a keyword.
2. it is used to declare an exception to be thrown by the method.

syntax: (it is always used in method declaration statement)

[modifiers] return_type method_name(arguments,..) throws exception1 , …

Note:

1. for checked exceptions, if the exception is not handled then it is mandatory to declare using throws.

2. **If a method declares an exception using throws, then the caller of the method must either declare or handle the exception.**

example:

```
package excep4;

import java.io.FileNotFoundException;
import java.io.FileOutputStream;

public class File1 {

    public static FileOutputStream createFile(String path) throws FileNotFoundException
    {
        return new FileOutputStream(path) ;
    }
```

```
}
```

```
package com.excep1;

import java.io.FileNotFoundException;

public class File1driver1 {

	public static void main(String[] args) throws FileNotFoundException {

		String path = "D://Demo//X2.txt" ;

		File1.createFile(path) ;
	}
}
```

==========

throw :

1. it is a keyword.
2. it is used to raise an exception.
3. it creates a Throwable type object, pauses the normal execution of the program and will throw the exception object created.

   syntax :

   ```
   throw  new constructor()  ;
   ```

   Note:

   1. the object should be throwable type only, if not we get CTE.

Example refer : workspace/Exceptions/src/excep5/E1.java to E3.java

Assignment :

What is the difference between throw and throws ?

Custom Exception :

A programmer can create his own exception class, it is known as custom exception.

steps to create :

1. create a new class.
2. inherit any class from Throwable Hierarchy.

Note :

1. if it inherits checked exception, new exception class also behaves like a checked exception.
2. if it inherits unchecked exception, new exception also behaves like a unchecked exception.

Example :

refer :
workspace/exceptions/src/excep5/Employee.java, EmpDriver1.java