What is Exception?

An exception is an event, which occurs during the execution of program that disrupts the normal flow of the program's instructions.

Why Exception Occurs?

->A user has entered invalid data

->Network connectivity issues

->JVM ran out of memory

->user based exceptions

Types of Exceptions

1. Checked Exception Exceptional conditions that a well-written application should anticipate and recover from. For example - A user provide that name of a non existent file. exception occurs.

2. Unchecked Exception

Error Exceptional conditions that are external to the application, and the application usually cannot anticipate or recover from. For example- System/application is able to open the file but not ale to read it due to malfunctioning so it will throw an error.

Runtime Exception: These are exceptional conditions that are internal to the application, and the application usually cannot anticipate or recover from. For Example - If the file is read and logical error causes a null to be assed it will cause a exception.

How are Exceptions Handled?

->Throwing an exception When an error occurs within a method, the method creates an object and hands it off to the runtime system - exception object

->Call Stack The list of methods that had been called to get to method where the error occurred is known as the call stack. After a method throws an exception, the runtime system attempts to find something in the call stack to handle it.

->Catching the exception The runtime system passes the exception object to appropriate "exception handler- a method that contains a block of code that can handle the exception. The exception handler chosen is said to catch the exception.

**Blocks used in Exception handling:**

->Try-catch-finally: A method that specifies that it can throw the exception. The method must provide a throws clause.

->Try Block:

Contains set of statements that can throw exceptions

->Catch Block: Exception handler if the statements in the try block throws an exception

->Finally Block:

Optional block: It always executes when the try blocks exits. Finally block is executed even if an unexpected exception occurs.

try

{

//statements

}

catch(ExceptionType1 e1)

{

//catch block

}

catch(ExceptionType2 e2)

{

//catch block

}

finally

{

//it always executes

}

Static:

The static variable gets memory only once in class area at the time of class loading. It makes the program memory efficient For example: - static String company="CSS Corp"; A static method belongs to the class rather than object of a class. It does not require the need of creating an instance of a class.

Restrictions For Static Method

The static method cannot use non static data member to call non-static method directly(need object).

this and super cannot be used in static context.

Final

It is used to restrict the usage. The final keyword can be used with variable, method and class.

You cannot subclass a final class.

You cannot override a final method.

A final variable is a constant.

Enum:

An enum is a data type which contains fixed set of constants. Enums can be thought of as classes that have fixed set of constants. Benefits of Enum -enum improves type safety

enum can be easily used in switch

enum can be traversed

enum can have fields, constructors and methods.

enum may implement many interfaces but cannot extend any class because it internally extends Enum class.

Example: -

class Example

{

public enum Season

{Winter, Spring, Summer}

}

public static void main (String[] args)

{

for(Season s:Season.values())

System.out.println(s);

}

}

Value() Method The value method returns an array containing all the values of enum

Sample programs:

package exceptional\_handling;

public class WithEH {

public static void main (String[] args) {

int a=10;

for(int i=3;i>=0;i--)

try{

System.out.println(a/i);

}catch(ArithmeticException e){

System.out.println(e);

}

}

}

package exceptional\_handling;

public class WithoutEH {

public static void main (String[] args) {

int a=10;

for(int i=3;i>=0;i--)

try{

System.out.println(a/i);

}

}

}