Exception Handling

    - **Checked Exception**
        - Directly inherits Throwable class except RuntimeException and Error.
        - Checked at compile time.
        - Eg:- IOException, SQLException
    - **Unchecked Exception**
        - Classes which inherit from RuntimeException.
        - Checked at runtime.
        - Eg:- AirthmeticException, NullPointerException, etc
    - **Error**
        - Irrecoverable
        - Eg:- OutOfMemoryError, VirtualMachineError

- **Try**
    - Piece of code which is expected to throw an exception is written in this block.
    - Should be followed by either catch or finally or both.
- **Catch**
    - Handles the flow if exception occurs.
    - Only activated if an exception occurs.
    - Can have multiple catch blocks.
    - Should be in the order of most specific to most generic.
- **Finally**
    - Used to perform important actions such as closing connections, streams, etc.
    - Runs irrespective of exception.
    - Not necessary but can have only one finally block.

– ***throw keyword is used to explicitly throw a checked or an unchecked exception. It is mostly used to throw a custom exception. Eg:- throw new IOException("device not allowed");***

Exception Propagation

An exception is first thrown from the top of the stack and if it is not caught, it drops down the call stack to the previous method,If not caught there, the exception again drops down to the previous method, and so on until they are caught or until they reach the very bottom of the call stack.

- ***By default only unchecked exceptions are forwarded in calling chain.***

- **throws**
    - Used to declare an exception.
    - Checked exceptions can also be forwarded in call stack.
    - Should only declare checked exceptions as unchecked exception is under programmers control.
    - Eg:- **public static void main() throws IOException**

    - ***If you are calling a method that declares an exception, you must either caught or declare the exception***

    - ***Final is a keyword which is used to apply restrictions on class, method and variable. Final class can’t be inherited, final method cannot be overridden and final variable’s value can’t be changed.***

    - ***Finally is a block which is used to place important piece of code.***

    - ***Finalize is a method used to perform clean up processing before garbage collection.***

Exception With Method Overriding

- **If the superclass method does not declare an exception**

subclass overridden method cannot declare the checked exception but it can declare unchecked exception.

- **If the superclass method declares an exception**

subclass overridden method can declare same, subclass exception or no exception but cannot declare parent exception.

Custom Exception

Create a class which extends Exception and pass the String message to the super.

Eg:-

```
class CustomException extends Exception {
        CustomException(String message) {
                super(message);
        }
}

class TestException {
        pvsm() {
                throw new CustomException("testing..");
        }
}
```

——————X—————X—————