**Encapsulation:**

Hiding the information and controlling the access of the data

Ex:bank transactions,passwords etc…

DTO -data transfer object-🡪properties should be private,must have an no arg constructor,must have setter and get method,toString method,equals method.

EX: <https://github.com/deepthins08/Core_Java/tree/main/Intellij/Access%20Specifier>

**Abstraction:**

Knowing the functionality but don’t know the implementation.

Incomplete idea

Ex: we have a thoughts to buy an apartment but physically do not exist. planning to have.

Abstract 🡪 it is a keyword,it means incomplete idea. Which is used in declaring class and methods.

When a class have an abstract method that class should be a abstract class.

Ex:Number,httpservlet,generalservlet. These are the exist abstract class

The abstract class can only get by inheriting it.

Ex: public abstract class demo{

Public abstract void display();

}

Public class child extends demo{

@override

Public void display(){

Sout(“displaying”);

}

EX: <https://github.com/deepthins08/Core_Java/tree/main/Intellij/Product/src/com/xworkz/product>

**Exception**:

Exception is a exceptional event which is a scenario that will distrubs the normal or regular flow.

Exception implements the Throwable interface.

Throwable is a interface is used handle the exceptions.

Throwable has two sub interfaces--🡪 Error and Exception.

Throwable

Runtime Exception

Exception

Error

Program issue

System issue

Data issue

Logical issue

Here Errors are used for the system issue.

Exception is used for the Program issue but in Exception it will deal with only data issue for logical issue RuntimeException will handle it. In a class if Exception or its sub interfaces are created the compiler will force to handle the event or exception is known as checked or compiletimeException.

RuntimeException is sub interface of Exception it has also so may sub interfaces. When RuntimeException or it sub interface are occurred the complier not force to handle it it also called uncheckedException or RuntimeException.

There are 5 keywords are there

* Throw🡪 Create an event
* Throws🡪delegate the event
* Try🡪absorb the event
* Cath🡪handle the event
* Finally🡪handle the resources

Throw: Throw keyword is used to create an event. If know the possible exception the we can create the event to tell that jvm to stop the execution.

When throw keyword is encountered the execution will stop and look for the handler if handler was not found then execution will be terminated.

Throws: Throws keyword is used for delegate the event means if we don’t know how to handle the event the we can tell that jvm to handle it own or to do some one.

Try: try block is a special block which absorbs the event means when we create an event inside the try block it will stop the execution and look for the handler if handler is not there it comes out of the try block only. It will not terminate the whole program.

Try can used with the diff handlers.

* Try with catch

Example of try

<https://github.com/deepthins08/Core_Java/tree/main/Intellij/Product/src/com/xworkz/product/exception/blocks>

* Try with multiple catch
* Try with multi catch
* Try with resources
* Try with finally
* Try with multiple catch and finally
* Try with multi catch and finally
* Try with catch and finally

Catch: it is used to handle the event or catch the event.

Syntax is

catch (subinterfaceException e) {

e.message();

}

Finally: it is used to handle the resources means used to close the events. Explicitly we call close() in finally.

When we used try with resources it will have automatic close method in it.

**Collections: Collection API (Application programming interface):** which is used to connect to did classes.

Collection is set of interfaces, classes, abstract classes, enum classes, annotations

Collections are grouping elements it is overcome the draw back of arrays. Array is fixed size we can not modify the length or size of the array. But in collection we can add n number of elements to the array.

Collection is in the package of java.util

List and set are sub interfaces of collection

Iterator

TreeSet

HashSet

LinkedHashSet

LinkedList

ArrayList

Set

List

Collection

Dotted lines are representing classes that are implemented those interfaces.

Example for colletion:

<https://github.com/deepthins08/Core_Java/tree/main/Intellij/collection/src/com/xworkz/collection>

**Comparator:**

**Comparator** which is used in collections for sorting the data in ascending and descending order. The comparator interface had a method compare method which will do the all the logic to sort the data. In order to sort the data in ascending and descending order we have to override the method. Using comparator we can sort all the properties by ascending and descending order.

Ex: <https://github.com/deepthins08/Core_Java/tree/main/Intellij/customDataTypeCollection/src/com/xworkz/customtype>

Ex 2: <https://github.com/deepthins08/Core_Java/tree/main/Intellij/collection/src/com/xworkz/collection>

**Comparable:**

**Comparable** which is used in collection in order to sorting the data. Using comparable we can set default property to sort in ascending or descending that cant be changed by overriding the **compareTo** method.

Ex: <https://github.com/deepthins08/Core_Java/tree/main/Intellij/comparable/src/com/xworkz/comparable>

**Lambda Expression:**

**Lambda expression** is used to sort the data in collection by reducing the code. If try to sort the properties in comparator or comparable we have to write separate class for to override the method for each property. but if we use lambda expression we don’t have to write another class we can sort directly using lambda expression.

Syntax: Comparator<> comp=(parameters) -> {

Statement;

}

Comparable<> comp1=(param)-> {

}

Ex: <https://github.com/deepthins08/Core_Java/tree/main/Intellij/lamdaExpression/src/com/xworkz/lamdaExpression>