OOPJ Notes Day-9 Session-1 Date: 06/05/2023

try with resource

- External resources like File, Database con and N/W con to be in try block before its use.
- Will be discussed in Java I/O and JDBC, Java Socket Programming

Custom exception and its need.

```java
class InvalidAgeException extends Exception
{
        public InvalidAgeException(String msg)
        {
                super(msg);
        }
}


public class CusExpDemo {


        public static void ValidateAge(int age) throws InvalidAgeException
        {
                if(age<18)
                {
                        throw new InvalidAgeException("Age is less than 18");
                }
                else
                {
                        System.out.println("Welcome To Vote");
                }
        }

        public static void main(String[] args) {


                int ag=13;
                try
                {
                        CusExpDemo.ValidateAge(ag);
                }
                catch(InvalidAgeException ex)
                {
                        System.out.println("Catched Exception:"+ex.getMessage());
                }



        }

}
```

Factory class and factory method

- We will discuss in MultiThreading and Serialization

Exception chaining and exception propagation

Generic programming using java.lang.Object class

- Need of generics

1. If we want to reduce developers effort then we should do generic programming.
2. If we want to write type-safe generic code then we should use generics.
3. Generics is a Java language feature which helps developer to write generic code by passing data type as a agrument.

- In Java, we can write generic code using:
    1. By Object Class
    2. Generics
- Why Generics
    1. Generics gives us stronger typechecking at compile time. In other words, using generics we can write type-safe generic code.
    2. It completly eliminates need of explict typecasting.
    3. It helps developer to define generic algorithms and data structures.
- Type argument & type parameter
- Commonly used type parameter names in Java
    1. T : Type
    2. N : Number
    3. K : Key
    4. V : Value
    5. E : Element
    6. S, U, R : Second type parameter names

- Bounded type parameters
    - If we want to put restriction on data type which is used as type argument then we should specify bounded type parameter.
    1. For Numbers value we give generic type as Number
- Wild card and its type
- Restrictions on generics : Kindly read them.
    - Reference: https://docs.oracle.com/javase/tutorial/java/generics/restrictions.html (https://docs.oracle.com/javase/tutorial/java/generics/restrictions.html)

Abstraction using interface

- Types of inheritance: Try by yourself
- Default method and static interface method: Try bye yourself
- Functional interface: to be discussed
- Abstract class versus interface: Kindly read and note this differnece
- Shallow copy and Deep copy: To be discussed
- Implementation of Cloneable interface: to be discussed

Working with collection

- Collection Framework Introduction
    - Collection Interface hierarchy
    - List Interface with hierarchy
    - ArrayList complete usage with non primitive instances
    - Use of equals, Comparable, Comparator in ArrayList
    - Vector
    - LinkedList
    - Traversing collection using Iterable & Iterator interface
    - Traversing using Enumeration, Iterator and ListIterator
    - Sorting instances using Comparable & Comparator interface
        - If we want to sort array/collection of non primitive type using Arrays.sort() method then non primitive type must implement Comparable interface.
        - Comparable is interface declared in java.lang package.
        - T -> the type of objects that this object may be compared to
        - Method:
            - int compareTo(T other)
                - Returns a negative integer(-1) : current/calling object is less than the specified object.
                - Returns zero(0): current/calling object is equal to the specified object.
                - Returns a positive integer(1) : current/calling object is greater than the specified object.
        - Comparable interface provides natural ordering(default ordering) of elements inside same class.
        - This interface imposes a total ordering on the objects of each class that implements it. This ordering is referred to as the class's natural ordering, and the class's compareTo method is referred to as its natural comparison method.

Nested, Inner and Anonymous classes: Will discuss on Monday (08-05-2023)

- Nested class and its type(static and non static nested class)
- Local class and its type(local inner class & anonymous inner class)
- Generic method
- Type erasure
- Bridge Method

- Fragile Base class problem