OOPJ Notes Revision Day-1 Date: 15-10-2024

Generics and Collection in Java-8

Generic Programming

- Generics
- Why Generics
- Generics Syntax
- Commonly used type parameter names in Java
    - T : Type
    - N : Number
    - K : Key
    - V : Value
    - E : Element
    - S, U, R : Second type parameter names
- Wild Card.
    - Types of wild card
        - Unbounded wild card :
        - Upper Bounded wild card
        - Lower Bounded wild card.
- Restrictions on Generics
- Ref: https://docs.oracle.com/javase/tutorial/java/generics/index.html

Collections Framework

- Collection Framework consist of various interfaces and classes to manage large quantity insatnces.
- Collection
- Ref: https://docs.oracle.com/javase/tutorial/collections/intro/index.html
- Collection Framework
    - Brief Introduction to:
        - Interfaces : Collection, List, Set, Map, Itarable,
        - Ref: https://docs.oracle.com/javase/tutorial/collections/interfaces/index.html
        - Classes : Collections, ArrayList, HashSet
- Ref: https://docs.oracle.com/javase/8/docs/technotes/guides/collections/overview.html
- Implementation of methods of ArrayList, HashSet and TreeSet using non-premitive type like String, Integer
- Difference b/w List and Set family
- Traversal of collection class objects using for-each, Itrator, ListItrator and forEach method
- Implementation of methods of ArrayList for user-defined classes like Student, Teacher and Employee
- Menu-Driving programs to Manage Employee Record by using ArrayList with options like
- Insert a Record, Update Record, Delete Record, View All record
- Menu-Driving programs to Book and Library Record by using ArrayList with options like
- Insert a Record, Update Record, Delete Record, View All record and Sorting by differnt parameter or Book like BookName, BookId, AutherName, BookPrice etc.

Brief introduction to Comparable and Comparator Interface

- Implementation of Comparable for TreeSet collection for storing user-defined class instance inside TreeSet
- Implementation of Comparator Interface

```
int compare(Student s1, Student s2)
{
    if(s1.RollNo>s2.RollNo)
    {
        return 1;
    }
    if(s1.RollNo<s2.RollNo)
    {
        return -1;
    }
    return 0;

}
```

- Using Class
- Using Anonymous class
- Using Lambda
- Introduction to Lambda and How to pass lambda to Functional Interface

Brief Introduction to Hasing, HashCode, HashFunction, Collision in HashMap

- Implementation of HashMap for classes like Integer and String
- Implementation of HashMap and HashTable for user defined classes like Employee and Address

Functional Programming in Java

- Functional Interfaces
- Impact of Functional programming upon Collection Framework
- Explore java.util.function package : Predicate, Map, Consumer, Supplier
- Lambda Expressions
- forEach method
- Introduction to Streams
- Streams vs. Collections
- java.util.stream.Stream API
- Types of Primitive Streams : IntStream, LongStream, DoubleStream & its API
- Different operations on streams : filter, map, reduce, sort, flatMap, anyMatch, count, boxing.