Unit – II
Java Programming Concepts

Text Book - Java Complete Reference by Herbert Schidlt

Rahul R.Pitale
Department of Computer Engineering
PCCOE

Reflections in Java

- Reflection is a very useful approach to deal with the java class at **runtime**
- It is used to load the java class,call its method at **runtime**
- It can inspects and dynamically call classes, methods,attributes at **runtime**

Reflection api classes

- java.lang.reflect – pkg
- Classes
  - java.lang.reflect.field
  - java.lang.reflect.method
  - java.lang.reflect.constructor
  - java.lang.reflect.modifier

Reflection API

- For Reflection the base concept is creating class Object to the perticular class
- How to Create class Object
  - **Class c = class.forName(“Teacher”)**
  - **Teacher t = new Teacher();**

    Class c = t.getClass();
  - **Class c = Teacher.class**

Boxing and Unboxing

- The automatic conversion of primitive data types into its equivalent Wrapper type is called Boxing
- Opposite operation is called Unboxing
- Advatages : No need of conversion between primitives and Wrappers manually so less coding is requires

Serialization and Deserialization

- Serialization:
  - It is a process of writing a state of object to a file
  - It is a process of converting an object from java supported form to file supported form
  - FileOutputStream and ObjectOutputStream classes we can used to achieve Serialization

- Deserialization
  - It is a process of reading state of an object from file
  - It is a process of converting an object from File into java supported form
  - FileInputStream and ObjectInputStream classes we can used to achieve desearialization

Collection Framework

- Collection
  - It is a group of individual objects as a single entity
  - Ex : Collection of Book
- Collection Framework
  - To represent a group of individual object as a single entity several classes and interfaces are required
  - Collection Framework defines several classes and interfaces to represent collection

Difference Between Array & Collections

ARRAY	COLLECTIONS
1.Fixed in Size	1.Growable in size
2.w.r.to memory arrays are not remmended to use	2. w.r.to memory collections are recommended
3.w.r.to Performance to good	3. w.r.to Performace not recommended
4.It holds only homogenous data elements	4. It Holds Homogenous and Heterogenous
5. Underlying Data Structure is not available i.e readymade methods is not avaliable	5. Based on some DS i.e for every collection readymade method is available
6. It hold primitive int[] as well as Object Integer[]	6.Only Object can hold

Interfaces of Collection Framework

1. Collection
2. List
3. Set
4. SortedSet
5. Navigable Set
6. Map
7. Queue

- Collection Interface
  - It defines most common methods applicable for any Object
  - It is root interface of collection framework

- List
  - List is child interface of collection
  - Where duplicates are allowed and insertion order preserved
  - **ArrayList**
  - **LinkedList**
  - **Vector**
  - **Stack**

- Set
  - It is child interface of collection
  - If you dont want duplicates and insertion order then you should go for Set
  - **HashSet**
  - **LinkedHashSet**
- Sorted Set
  - Duplicates are not allowed but if you want to represent group of individual object according to sorting order

- NavigableSet
  - It is a child interface of sorted set
  - **TreeSet**
- Map
  - It is not child interface of collection
  - Collection and map are two different things
  - It represent the object with key value pair
  - Key – duplicates are not allowed
  - Value – values will be duplicate
  - **HashMap – LinkedHashmap**
  - **WeakHashMap**
  - **IdentityHashMap**
  - **HashTable**

- Queue
  - FIFO
  - **PriorityQueue**
  - **BlokingQueue**
    - **LinkedBlockingQueue**
    - **PriorityBlockingQueue**

ArrayList

- Underlying DS – Resizable Array or growable array
- Duplicates are allowed
- Insertion order is preserved
- Homogenous as well as Heterogenous objects
- Null insertion is possible

Constructors in ArrayList

1. ArrayList l = new ArrayList();
2. ArrayList l = new ArrayList(int initalCapacity);
3. ArrayList l = new ArrayList(Collection c)

ArrayList Impements Interfaces

- Serializable
- Cloneable
- **RandomAccess (Only ArrayList and Vector)**

- ArrayList is best :
  - If frequent operation is **Retrieval**
- ArrayList is worst :
  - If operation is **insertion or deletion** of element in middle

LinkedList

- For Insertion and deletion in middle LinkedList is the best choice
- Underlying DS – Doubly LinkedList
- Insertion order is preserved
- Duplicates are allowed
- Homogenous and Heterogenous
- Null insertion is possible

Constructors in LinkedList

1. LinkedList l = new LinkedList();
2. LinkedList l = newLinkedList(Collection c);

Difference Between ArrayList and LinkedList

ArrayList	LinkedList
1. Best ; Retrieval	1.Best: Insertion and Delection
2. Worst: Insertion and Delection	2.Worst: Retrieval
3. Underlying DS : Resizable Array	3. Underlying DS: Double LinkedList
4. RandomAccess Interface	4. No RandomAccess Interface

Vector

- Underlying DS – Resizable Array
- Duplicates are allowed
- Insertion order is perserved
- Null insertion possible
- Homogenous and Heterogenous
- Implements Serializable,Clonable and RandomAccess Interface
- Methods are Synchronized hence it is threadsafe
- For Retrieval vector is best choice

Vector Methods

- For adding object
  - add(Object o) ------[from Collection]
  - add(int index,Object o) -----[from List]
  - addElement(Object o)-----[from Vector]
- For Removing Object
  - Remove(Object o)------[from Collection]
  - removeElement(Object o)----[from Vector]
  - remove(int index)----[from List]
  - RemoveElementAt(int index)-----[from Vecor]
  - Clear -----[from collection]
  - RemoveAllElement() [from Vector]

- For Retrieving Objects
  - Object get(int index) ---[from Collection]
  - Object elementAt(int index)-----[from Vector]
  - Object firstElement() ---[from vector]
  - Object lastElement()----[from vector]
- Other Methods
  - int size()
  - int capacity()

Constructors

1. Vector v = new Vector();
2. Vector v = new Vector(int initalcapacity);
3. Vector v = new Vector(int initalcapacity,int incrementalcapacity);
4. Vector v = new Vector(Collection c);

Stack

- It is child class of vector
- It is specailly for LIFO order
- Constructors
  - Stack s = new Stack()
- Methods
  - Push(Object o)
  - Pop()
  - Peek()
  - empty()
  - search()

HashSet

- Undelying DS – HashTable
- Duplicates are not allowed
- Insertion order not perserved
- Heterogenous and Homogenous
- Null is possible
- Implements serializable and clonable

Constructors

1. HashSet h = new HashSet();
2. HashSet h = new HashSet(int initialCapacity);
3. HashSet h = new HashSet(int initialCapacity,float loadfactor);
4. HashSet h = new Hashset(Collection c)

TreeSet

- Underlying DS – Balanced Tree
- Duplicates are not allowed
- Insertion order is not applicable
- Sorting order is applicable
- Heterogenous are not allowed
- Null acceptance **(Only once)**

Constructors

1. TreeSet t = new TreeSet();
2. TreeSet t = new TreeSet(Comparator c);
3. TreeSet t = new TreeSet(Collection c);
4. TreeSet t = new TreeSet(SortedSet s);

Enumeration

- It is use to retrive objects one by one from collections
- Using Element method we can get enumeration object
- Ex: Enumeration e = v.element()
- Methods
  - Public boolean hasMoreElements();
  - Public object nextElement();

Limitation of Enumeration

- It is only for legacy class not for all classes like linked list,arraylist etc..
- Only read operation perform using enumeration

Iterator

- It is applicable for any collection class
- Read as well as remove operation
- How to create object?
- Ex: Iterator itr = c.iterator();
- Methods
  - Public boolean hasNext()
  - Public Object next()
  - Public void remove()

Limitation of Iterator

- It is single direction
- Addition of new object not possible

List Iterator

- It is Bidirectional cursor
- Replacement and addition of new object is possible
- How to create Object?
- Ex:ListIterator itr = l.listIterator();
- Methods
  - public boolean hasNext()
  - public object next()
  - Public int nextIndex()
  - Public boolean hasPrevious()
  - Public object previous()
  - Public int previousIndex()
  - Public void remove()
  - Public void set (it is for replacement)
  - Public void add()

StringTokenizers

- java.util.StringTokenizers class allows an application to break a string into tokens
- Constructors in String Tokenizers
  - StringTokenizers(String str)
  - StringTokenizers(String str,String delim)
  - StringTokenizers(String str,String delim,boolean returnDelims)

- Methods
  - Int countTokens()
  - Boolean hasMoreElements()
  - Boolean hasMoreTokens()
  - Object nextElement()
  - String nextToken()
  - String nextToken(String delim)

Dictionary

- Dictionary is an abstract class that represents a key/value storage repository and operates much like Map.
- Given a key and value, you can store the value in a Dictionary object. Once the value is stored, you can retrieve it by using its key. Thus, like a map, a dictionary can be thought of as a list of key/value pairs.

Methods

- Enumeration elements( )
- Object get(Object key)
- boolean isEmpty( )
- Enumeration keys( )
- Object put(Object key, Object value)
- Object remove(Object key)
- int size( )

  The Dictionary class is obsolete.
- You should implement the Map interface to obtain key/value storage functionality.

Observable

- The java.util.Observable class represents an observable object, or "data" in the model-view paradigm.
- Following are the important points about Observable:
  - The class can be subclassed to represent an object that the application wants to have observed.
  - An observable object can have one or more observers.

- Class declaration
  - public class Observable extends Object
- Class constructors
  - Observable()
  - This constructs an Observable with zero Observers.

- Class methods
  - void addObserver(Observer o)
  - protected void clearChanged()
  - int countObservers()
  - void deleteObserver(Observer o)
  - void deleteObservers()
  - boolean hasChanged()
  - void notifyObservers()
  - void notifyObservers(Object arg)
  - protected void setChanged()