Object Oriented Programming with Java (Subject Code: BCS-403)

Unit 4

Lecture 36

Lecture 36

- Comparable Interface
- Comparator Interface
- Properties Class in Java

Comparable and Comparator Interfaces

Comparable Interface: Objects that implement the Comparable interface can be sorted based on their natural order. The compareTo() method defines the sorting criteria.

Comparator Interface: The Comparator interface allows you to define a custom sorting order for objects that don't implement Comparable or when you need to sort based on different criteria.

Sorting Methods

Collections.sort(List): Sorts the elements in the specified List using the natural order defined by the elements' Comparable implementation.

Collections.sort(List, Comparator): Sorts the elements in the specified List using the provided Comparator.

Arrays.sort(array): Sorts the specified array using the natural order defined by the elements' Comparable implementation.

Arrays.sort(array, Comparator): Sorts the specified array using the provided Comparator.

- Java Comparable interface is used to order the objects of the user-defined class.
- This interface is found in java.lang package and contains only one method named compareTo(Object).
- It provides a single sorting sequence only, i.e., you can sort the elements on the basis of single data member only.
- For example, it may be rollno, name, age or anything else.

compareTo(Object obj

public int compareTo(Object obj): It is used to compare the current object with the specified object. It returns

- positive integer, if the current object is greater than the specified object.
- negative integer, if the current object is less than the specified object.
- zero, if the current object is equal to the specified object.

```
class Student implements Comparable<Student>{
int rollno;
String name;
int age;
Student(int rollno,String name,int age){
this.rollno=rollno;
this.name=name;
this.age=age;
}

public int compareTo(Student st){
if(age==st.age)
return 0;
else if(age>st.age)
return 1;
else
return -1;
}
}
```

```
import java.util.*;
public class TestSort1{
public static void main(String args[]){
ArrayList<Student> al=new ArrayList<Student>();
al.add(new Student(101,"Vijay",23));
al.add(new Student(106,"Ajay",27));
al.add(new Student(105,"Jai",21));

Collections.sort(al);
for(Student st:al){
System.out.println(st.rollno+" "+st.name+" "+st.age);
}
}
}
```

```
105 Jai 21
101 Vijay 23
106 Ajay 27
```

Comparator interface

- **Java Comparator interface** is used to order the objects of a user-defined class.
- This interface is found in java.util package and contains 2 methods compare(Object obj1,Object obj2) and equals(Object element).
- It provides multiple sorting sequences, i.e., you can sort the elements on the basis of any data member, for example, rollno, name, age or anything else.

Properties Class in Java

- The **properties** object contains key and value pair both as a string.
- The java.util.Properties class is the subclass of Hashtable.
- It can be used to get property value based on the property key.
- The Properties class provides methods to get data from the properties file and store data into the properties file.
- Moreover, it can be used to get the properties of a system.

Features of Properties Class in Java

Inheritance: Properties extends the Hashtable class, inheriting its thread-safe behavior and the ability to store key-value pairs. However, both keys and values in a Properties object are treated as Strings.

Loading Properties: Properties objects can be loaded from various sources, including files, input streams, and readers. The load() method reads properties from an input stream in a simple line-oriented format (key=value).

Storing Properties: The store() method can write the contents of a Properties object to an output stream or writer in a format suitable for loading into a Properties object.

Constructors of Properties class

Method	Description
Properties()	It creates an empty property list with no default values.
Properties(Properties defaults)	It creates an empty property list with the specified defaults.

Methods of Properties class

Method	Description
public void load(Reader r)	It loads data from the Reader object.
public void load(InputStream is)	It loads data from the InputStream object
public void loadFromXML(InputStream in)	It is used to load all of the properties represented by the XML document on the specified input stream into this properties table.
public String getProperty(String key)	It returns value based on the key.
public String getProperty(String key, String defaultValue)	It searches for the property with the specified key.
public void setProperty(String key, String value)	It calls the put method of Hashtable.
public void list(PrintStream out)	It is used to print the property list out to the specified output stream.
public void list(PrintWriter out))	It is used to print the property list out to the specified output stream.

Features of Properties Class in Java

Property File Format: The default format for property files is a simple line-oriented format with key-value pairs separated by an equal sign (=). Comments can be included by starting a line with a hash (#) or an exclamation mark (!).

Retrieving Values: Values can be retrieved from a Properties object using the getProperty() method, which takes a key as an argument and returns the corresponding value as a String. If the key is not found, it returns null or a specified default value.

Accessing Properties: In addition to accessing properties directly, Java provides a utility class called System that allows access to system properties, which are accessible through the System.getProperties() method.

ABES Engineering College
College Code 032

java class to read the data from the properties file.

```
import java.util.*;
import java.io.*;
public class Test {
public static void main(String[] args)throws Exception{
    FileReader reader=new FileReader("db.properties");
    Properties p=new Properties();
    p.load(reader);
    System.out.println(p.getProperty("user"));
    System.out.println(p.getProperty("password"));
}
}
```

db.properties

```
user=system
password=oracle
```

Output:system
oracle

Example of Properties class to get all the system properties

```
import java.util.*;
import java.io.*;
public class Test {
public static void main(String[] args)throws Exception{
Properties p=System.getProperties();
Set set=p.entrySet();
Iterator itr=set.iterator();
while(itr.hasNext()){
Map.Entry entry=(Map.Entry)itr.next();
System.out.println(entry.getKey()+" = "+entry.getValue());
}
}
}
```