**Utility Classes in Java**

Data Structure Classes

A variety of useful classes implementing standard computer science data structures: including BitSet, Dictionary, Hashtable, Stack and Vector. The java.util package also defines the Enumeration interface which provides a protocol for classes to count through a set of values.

Date

Use the Date class to create and manipulate calendar dates in a system-independent fashion.\_DNL StringTokenizer

This class converts a String of text into its tokens.

Properties

This class implements persistent properties. The properties table contains key/value pairs where both the key and the value are Strings. This class is used by the System class to implement System properties.

Observer and Observable

Classes that implement the Observer interface can "watch" Observable objects for state changes. When an Observable object changes it notifies all of its Observers of the change.

Random-Number Generator

The Random class implements a random-number generator.

Enumeration

The Enumeration interface defines a generic programming interface for iterating through a set of values.

**String Tokenizer**

The string tokenizer class allows an application to break a string into tokens. The StringTokenizer methods do not distinguish among identifiers, numbers, and quoted strings, nor do they recognize and skip comments.

The set of delimiters (the characters that separate tokens) may be specified either at creation time or on a per-token basis.

import java.util.StringTokenizer;

class StringTokenizerDemo

{

public static void main(String arg[])

{

StringTokenizer st1 = new StringTokenizer("Welcome to Java " +

" sessions."); // LINE A

while (st1.hasMoreTokens()) {

System.out.println(st1.nextToken());

}

System.out.print("---------\n");

StringTokenizer st2 = new StringTokenizer("Good morning,Today is," +

"Saturday.", ","); // LINE B

while (st2.hasMoreTokens()) {

System.out.println(st2.nextToken());

}

System.out.print("---------\n");

StringTokenizer st3 = new StringTokenizer("This-is-a-string-with hyphens.",

"-", true); // LINE C

while (st3.hasMoreTokens()) {

System.out.println(st3.nextToken());

}

}

}

**BitSet**

The BitSet class creates a special type of array that holds bit values. This array can increase in size as needed. This makes it similar to a vector of bits. The bits of a BitSet are indexed by nonnegative integers. Individual indexed bits can be examined, set, or cleared. One BitSet may be used to modify the contents of another BitSet through logical AND, logical inclusive OR, and logical exclusive OR operations.

lass BitSetDemo  
{  
    public static void main(String arg[])  
    {  
        BitSet bitSetA = new BitSet();  
        bitSetA.set(2);  
        bitSetA.set(3);  
        bitSetA.set(4);  
        bitSetA.set(5);  
        System.out.println("bitSetA = " + bitSetA);  
        BitSet bitSetB = new BitSet();  
        bitSetB.set(1, 5);  
        System.out.println("bitSetB = " + bitSetB);  
        // Clone of bitSetA  
        BitSet bitSetC = new BitSet();  
        bitSetC = (BitSet) bitSetA.clone();  
        System.out.println("bitSetC = " + bitSetC);  
        // AND bits  
        bitSetA.and(bitSetB);  
        System.out.println("bitSetA and bitSetB = " + bitSetA);  
        // OR bits  
        bitSetA.or(bitSetB);  
        System.out.println("bitSetA or bitSetB = " + bitSetA);  
        // AND NOT  
        bitSetA.andNot(bitSetB);  
        System.out.println("bitSetA and not bitSetB = " + bitSetA);  
        // XOR bits  
        bitSetC.xor(bitSetB);  
        System.out.println("bitSetC xor bitSetB = " + bitSetC);      
    }  
}

**Date class**

Date class is available in java.util package, it represents a specific instant of time, with millisecond precision. Date allows the interpretation of dates as year, month, day, hour, minute, and second values. It also allows the formatting and parsing of date strings.

import java.util.\*;  
  
class DateExampleTest  
{  
    public static void main(String arg[])  
    {  
        try  
        {  
            // Get current date and time  
            Date date = new Date(); // LINE A  
            System.out.println(date);  
            // Convert Date to String.  
            SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy hh:mm:ss"); // LINE B  
            String stringDate = dateFormat.format(date); // LINE C  
            System.out.println(stringDate);  
            // Convert String to Date.  
            SimpleDateFormat df = new SimpleDateFormat("dd/MM/yyyy HH:mm:ss");  
            String dateInString = "15/08/1947 02:25:56";  
            date = df.parse(dateInString); // LINE D  
            System.out.println(date);  
        } catch (Exception e)  
        {  
            e.printStackTrace();  
        }      
    }  
}

**Calendar Class**

Calender is an abstract class. It provides a set of methods to manipulate the date and time. The subclass of calender class provides the specific implementation to the abstract methods defined by calender to meet their own requirements.

import java.util.Calendar;  
  
class ExampleOfDateAndTime  
{  
    public static void main(String arg[])  
    {  
        String months[] = {"JAN", "FEB", "MAR", "APR", "MAY", "JUNE", "JULY", "AUG", "SEPT", "OCT", "NOV", "DEC"};  
          
        /\*Create a calendar initialized with the  
        current date and time in the default  
        locale and timezone.\*/  
        Calendar cal = Calendar.getInstance();  
          
        //Display current time and date information.  
        System.out.println("Date :");  
        System.out.print(months[cal.get(java.util.Calendar.MONTH)]);  
        System.out.print(" " + cal.get(Calendar.DATE) + " ");// LINE D  
        System.out.println(cal.get(Calendar.YEAR));  
          
        System.out.println("Current Time: "); // LINE A  
        System.out.print(cal.get(Calendar.HOUR) + ":");  
        System.out.print(cal.get(Calendar.MINUTE) + ":");  
        System.out.println(cal.get(Calendar.SECOND));  
          
        //Set the time and date information and display it.  
        cal.set(Calendar.HOUR, 10); // LINE C  
        cal.set(Calendar.MINUTE, 29); // LINE C  
        cal.set(Calendar.SECOND, 22); // LINE C  
          
        System.out.println("Updated time"); // LINE B  
        System.out.print(cal.get(Calendar.HOUR) + ":");  
        System.out.print(cal.get(Calendar.MINUTE) + ":");  
        System.out.println(cal.get(Calendar.SECOND));  
      
    }  
}

**Timezone**

import java.util.\*;  
  
class TimeZoneDemo  
{  
    public static void main(String arg[])  
    {  
        Calendar now = Calendar.getInstance(); // LINE A   
        TimeZone timeZone = TimeZone.getTimeZone("IST"); // LINE B  
        System.out.println(timeZone.getDisplayName()); // LINE C  
        timeZone = TimeZone.getTimeZone("GMT");  
        now.setTimeZone(timeZone); // LINE D  
        System.out.println(timeZone.getDisplayName());  
      
    }  
}

**Locale**

import java.util.Locale;  
  
class LocaleDemo  
{  
    public static void main(String[] args)  
    {  
        Locale l = Locale.getDefault(); // LINE A  
        System.out.println(l);   
        System.out.println(l.getDisplayCountry()); // LINE B  
        System.out.println(l.getCountry()); // LINE C  
        System.out.println(l.getDisplayLanguage()); // LINE D  
        System.out.println(l.getLanguage()); // LINE E  
    }  
}

**Random**

import java.util.Random;  
  
class RandomTest  
{  
    public static void main(String arg[])  
    {  
        Random r = new Random();  
          
        System.out.println("The next Integer value : " + r.nextInt()); // LINE A  
        System.out.println("The next Boolean value : " + r.nextBoolean());  
          
        byte[] b = new byte[30];  
        r.nextBytes(b); // Puts the next byte in array  
        System.out.println("Value of Byte array : " + b);  
          
        System.out.println("The next Double value : " + r.nextDouble());  
        System.out.println("The next Float value : " + r.nextFloat());  
        System.out.println("The next Gaussian value : " + r.nextGaussian());  
        System.out.println("The next Integer  value between 0 to 4 : " + r.nextInt(5));  
        System.out.println("The next Long value : " + r.nextLong());  
        r.setSeed(20);  
        System.out.println("The set Seed value : " + r.nextInt());  
      
    }  
}

**Currency**

import java.util.\*;  
  
class CurrencyDemo  
{  
    public static void main(String arg[])  
    {  
        Currency currency = Currency.getInstance(Locale.US);  
        System.out.println( "Currency : " + currency.getDisplayName());  
        System.out.println("Symbol : " + currency.getSymbol());      
    }  
}

**Formatter**

import java.util.\*;  
  
class BasicFormatterDemo  
{  
    public static void main(String arg[])  
    {  
        Formatter formatter = new Formatter();  
          
        formatter.format("Learn Java with %s in %d days and score %f", "Clear concepts", 10, 97);  
          
        System.out.println(formatter);  
        formatter.close();      
    }  
}

**Scanner Class**

The Scanner class is a class in java.util package, which allows a user to read values of various types.

The Scanner looks for tokens in the input. A token is a series of characters that ends with whitespace. A whitespace character can be a blank, a tab character, a carriage return, or the end of the file.

For example, if we read a line that has a series of numbers separated by blanks, the scanner will take each number as a separate token.

**package** day\_19;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.util.Scanner;

**public** **class** ScannerConstructorDemo1 {

**public** **static** **void** main(String arg[]) **throws** FileNotFoundException

{

// Read from a file rather than the keyboard

Scanner sc = **new** Scanner(**new** File("/Users/sabniss/Desktop/java-training/java-training/src/day\_19/ScannerConstructorDemo1.java")); // LINE A

// Check if sc has another token in the file

**while**(sc.hasNext())

System.***out***.println(sc.next());

// Close the scanner

sc.close();

}

}