CS 1301 – Ch 8, Handout 3

This section discusses the StringBuilder and StringBuffer classes, the File and PrintWriter classes to write text files, as well as the Scanner class to read text files.

The StringBuilder and StringBuffer Classes

1. The StringBuilder and StringBuffer classes are more flexible ways of representing Strings. They are more efficient when you need to change a string; however, the String class is better if you are not going to change a string.The StringBuffer class is *synchronized*, meaning that multiple *threads* can access it safely. Other than that, it is about the same as the StringBuilder class.

2. Some of the methods in StringBuilder: append, delete, deleteCharAt, insert, replace, reverse, setCharAt, toString, capacity, charAt, length, substring, trimToSize. Explore the API:

   http://java.sun.com/javase/6/docs/api/java/lang/StringBuilder.html

3. Example: Write a method that returns the hex representation of a decimal number (integer).

```
public static void main(String[] args)
{
   long num = 444;
   System.out.println( " Decimal: " + num + ", Hex: " + decimalToHex( num ) );
}

public static String decimalToHex( long num )
{
   StringBuilder hexStrBld = new StringBuilder();
   int digit;

   while( num > 0 )
   {
      digit = (int)(num % 16);
      hexStrBld.append( charDigit( digit ) );
      num /= 16;
      System.out.println( digit );
   }
   hexStrBld.reverse();
   return hexStrBld.toString();
}

public static char charDigit( int digit )
{
   switch( digit )
   {
      case 10: return 'A';
      case 11: return 'B';
      case 12: return 'C';
      case 13: return 'D';
      case 14: return 'E';
      case 15: return 'F';
      default: return String.valueOf( digit ).toCharArray()[0];

   }
}
```

4. Example: Write a method to reverse the characters in a string array. We saw this example in the Chapter 8A notes.

```
public class ReverseStringExample
{
   public static void main(String[] args)
  {
      System.out.println( reverseString("Welcome Home") );
   }

  public static String reverseString( String string )
   {

      StringBuilder stringSB = new StringBuilder( string );

      StringBuilder reverseSB = stringSB.reverse();

      String revS = reverseSB.toString();

      return revS;

//    return new StringBuilder(string).reverse().toString();
   }

}
```

The output is: emoH emocleW

5. Example: Illustrate some of the StringBuilder methods.

```
StringBuilder sb = new StringBuilder( "Welcome Home" );

System.out.println( sb.toString() );  // Welcome Home

StringBuilder sb2 = new StringBuilder( "Chief" );

sb.append( sb2 );

System.out.println( sb.toString() );  // Welcome HomeChief

int index = sb.indexOf( "Chief" );

sb.insert( index, ' ' );

System.out.println( sb.toString() );  // Welcome Home Chief

sb.append( " forof Police" );

System.out.println( sb.toString() );  // Welcome Home Chief forof Police
```

```
int beg = sb.indexOf("for");

sb.delete( beg, beg+3 );

System.out.println( sb.toString() );  // Welcome Home Chief of Police

for( int i=0; i<sb.length(); i++ )
{
   if( Character.isUpperCase( sb.charAt(i) ) )
      sb.setCharAt( i, Character.toLowerCase( sb.charAt(i) ) );
   else
      sb.setCharAt( i, Character.toUpperCase( sb.charAt(i) ) );
}

System.out.println( sb.toString() );  // wELCOME hOME cHIEF OF pOLICE

sb = new StringBuilder( "Welcome Home Chief of Police" );

System.out.println( sb.toString() );  // Welcome Home Chief of Police

beg = sb.indexOf( "Welcome Home" );
int end = "Welcome Home".length();

sb.replace( beg, end, "Howdy" );

System.out.println( sb.toString() );  // Howdy Chief of Police

beg = sb.indexOf("Chief");
end = beg + "Chief".length();
String s = sb.substring( beg, end );

sb.append( " " + s + "!" );

System.out.println( sb.toString() );  // Howdy Chief of Police Chief!

System.out.println( sb.capacity() );  // 44
System.out.println( sb.length() );    // 28
sb.trimToSize();
System.out.println( sb.capacity() );  // 28
System.out.println( sb.length() );    // 28
```

The File Class

1. The File class is used to obtain information about and access to files stored on media. Some of the useful methods are: exists, canRead, canWrite, isDirectory, isFile, isHidden, getAbsolutePath, getName, getPath, getParent, lastModified, length, listFile, delete, renameTo. Explore the API:

   http://java.sun.com/javase/6/docs/api/java/io/File.html

The PrintWriter Class

1. The PrintWriter class is used to write text files. Some of the important methods are: print, println, printf.

2. http://java.sun.com/javase/6/docs/api/java/io/PrintWriter.html

Example: Writing a Text File

```
import java.io.PrintWriter;
import java.io.File;

public class WriteFile
{
   public static void main(String[] args) throws Exception
   {
      // Declare output file.
      File outFile;

      // If file name not specified on command line, use default.
      if( args.length == 0 )
         outFile = new File( "isdd.txt" );
      else
      // otherwise, use file name on command line.
         outFile = new File( args[0] );

      // See if file exists already.
      if( outFile.exists() )
     {
         // If it does, the exit the program so as not to
         // overwrite this file.
         System.out.println( "File already exists" );
         System.exit(0);
      }
      // We will write these numbers to a file, 3 per line,
      // comma delimited.
      int[] nums = {33, 44, 55, 66, 12, 33, 55, 66, 77, 22};

      // Declare and create a PrintWriter.
      PrintWriter writer = new PrintWriter( outFile );

      // Loop over the array of numbers.
      for( int i=0; i<nums.length; i++ )
      {
         // Write a number.
         writer.print( nums[i] );

         if( (i+1)%3 == 0 )
            // If third number of line, start a new line.
            writer.println();
         else
            if ( i < nums.length-1 )
               // If not the last number, then print a delimiter.
               writer.print( ", " );
      }
      // Close the PrintWriter.
      writer.close();
   }
}
```

The Scanner Class

1. The Scanner class is used to read text files. Some of the important methods are: close, hasNext, next, nextLine, nextInt, nextDouble.

2. http://java.sun.com/javase/6/docs/api/java/util/Scanner.html

Example: Reading a Text File

1. In this case, we want to read a text file (employees.txt) that looks like this:

```
Dave
44.55
Custodian
Sue
33.44
Accountant
Sherry
55.34
Programmer
Mike
23.45
Intern
Nicole
76.34
President
```

And we want to create *Employee* objects from this data. Each three lines of the text file correspond to an employee specifying their name, salary, and title. The *Employee* class:

```
public class Employee
{
    private String name;
    private double salary;
    private String title;

    public Employee( String name, double salary, String title )
    {
        this.name = name;
        this.salary = salary;
        this.title = title;
    }

    public String getName() { return name; }
    public double getSalary() { return salary; };
    public String getTitle() { return title; }
}
```

The driver that does the reading and creating:

```
import java.util.Scanner;
import java.io.File;

public class ReadFile
{
   public static void main(String[] args) throws Exception
   {
      String name, title, strSalary;
      double salary;
      Employee[] employees = new Employee[100];
      int numEmps = 0;

      // Declare input file.
      File inFile;

      // If file name not specified on command line, use default.
      if( args.length == 0 )
         inFile = new File( "employees.txt" );
      else
      // otherwise, use file name on command line.
         inFile = new File( args[0] );

      if( !inFile.exists() )
      {
         System.out.println( "Can't find file" );
         System.exit(0);
      }
      // Declare and create Scanner.
      Scanner input = new Scanner( inFile );

      // While there are more things to read...
      while( input.hasNext() )
      {
         // Read name, salary, and title.
         name = input.next();
         salary = input.nextDouble();
         title = input.next();

         // Create Employee and put into array.
         employees[numEmps++] = new Employee( name, salary, title );
      }
      // Close the Scanner.
      input.close();

      // Display the array of Employees that were read.
      for( int i=0; i<numEmps; i++ )
         System.out.printf( "name: %s, salary: %.2f, title: %s\n",
                            employees[i].getName(),
                            employees[i].getSalary(),
                            employees[i].getTitle() );
   }
}
```

Note, this program will also work if the text file (e2.txt) is specified like this:

Dave 44.55 Custodian
Sue 33.44 Accountant
Sherry 55.34 Programmer
Mike 23.45 Intern
Nicole 76.34 President

Note, this program will fail if a *name* has two words, *e.g.* Sue Ellen.

2. Suppose that we have a variable number of items on a line. For instance, suppose that we want to add up all the values on a line, for each line. Here an input file:

3 4 2 10 12
3 6
5 7 8

Here, we'll have to read an entire line as a string, using the *nextLine()* method. Then, we'll have to parse the line using a regular expression:

```
import java.util.Scanner;
import java.io.File;

public class ReadFile2
{
   public static void main(String[] args) throws Exception
   {
      String line;
     int sum = 0, val;
      String[] tokens;

      File inFile;

      if( args.length == 0 )
         inFile = new File( "nums.txt" );
      else
         inFile = new File( args[0] );

      if( !inFile.exists() )
      {
         System.out.println( "Can't find file" );
         System.exit(0);
      }

      Scanner input = new Scanner( inFile );
```

```
// While there are more lines to read...
      while( input.hasNext() )
      {
         // Read a line
         line = input.nextLine();

         // Break line into tokens.
//          tokens = line.split(" +");
         tokens = line.split("\\s+");
         // Initialize sum to 0.
         sum = 0;

         // Loop through the tokens.
         for( String t : tokens )
         {
            // Convert token to integer and add it.
            val = Integer.parseInt( t );
            sum += val;
         }
         // Display the sum on a line.
         System.out.println( sum );
      }
  }
}
```

3. Suppose that we want to read some numbers and a string that may contain multiple words. For instance, we may want to read: age, salary, title. And, "title" can have multiple words in it. This is simple if we require that the multiword string be at the end of each line. For example:

    33 44.55 Custodian
    23 55.34 Programmer Super Max
    56 23.45 Intern
    32 99.23 Wholesale Marketer
    19 76.34 President

   Here, we'll have to read an entire line as a string, using the *nextLine()* method. Then, we'll have to parse the line using a regular expression:

```
Scanner input = new Scanner( inFile );

while( input.hasNext() )
{
   age = input.nextInt();
   salary = input.nextDouble();
   title = input.nextLine();

   System.out.printf( "age: %d, salary: %.2f, title: %s\n",
                      age, salary, title);
}
```

4. Same as the previous example, except suppose that the title is at the beginning:

Custodian 33 44.55
Accountant 44 33.44
Programmer Super Max 23 55.34
Intern 56 23.45
Wholesale Marketer 32 99.23
President 19 76.34

The solution is to basically, *work backwards.* When we split the line, we know that the last two tokens have the age and salary. Then, we can *mark* where the title ends, at index .length-3. Then, we can loop from index 0 to *end* and get each word in the title.

```
while( input.hasNext() )
{
  String line = input.nextLine();

   // Break line into tokens.
   tokens = line.split("\\s+");

   // Know last two are numbers:

   salary = Double.parseDouble( tokens[tokens.length-1] );
   age = Integer.parseInt( tokens[tokens.length-2] );

   int end = tokens.length-3;

   StringBuilder sb = new StringBuilder();

   for( int i=0; i<=end-1; i++ )
   {
      sb.append( tokens[i] + " " );
   }
   sb.append( tokens[end] );

   title = sb.toString();

   System.out.printf( "title: %s, age: %d, salary: %.2f\n",
                      title, age, salary );
}
```