Creating and Modifying Text

Barb Ericson
Georgia Institute of Technology
Nov 2010

Learning Goals

- Manipulate strings
- Read and write files
- Handle exceptions
- Use a dynamic array: ArrayList
- Explain interfaces
- Create a class method
- Explain generics
- Write programs that manipulate programs

Why work with text?

- Text is everywhere
  - Web pages are text
  - Google works by reading the text
- Alice only allows us to work with text when we use say
  - Java lets us do much more with text
- Any type of media can be stored as text
  - Text is unimedia

Text as Unimedia

- Computers only understand 0 and 1
  - On and off of voltage
- But we can store anything with that
  - Text, Pictures, Sounds, Movies, HTML pages
- We can do the same with Text
  - Convert a picture to text
  - Convert a sound to text
- HTML is a textual language
  - That is used to create web pages

HTML

- Open a browser and go to any web page
  - Go to the View menu and click on Source
  - What you see is HTML
- HTML is HyperText Markup Language
  - Uses special tags to denote sections of a document
  - <title>This is the Title</title>
  - This tag starts a new paragraph
  - This tag means to show this in bold

java.lang.String

- Text in Java is stored as a String object
  - In Unicode format
    - 2 bytes per character (16 bits)
    - Matches ASCII for the first 128 characters
- A string literal is enclosed in double quotes
  - String message = "Hi There";
- To add a double quote to a string
  - Use \"

```
> String s = "She said, \"Hi there\"";
> System.out.println(s);
She said, "Hi there"
```

- Other special characters:
  - \n for new line
  - \t for tab

Strings are Sequences of Characters

- You can get the character at an index
  - Starting with index 0

  stringObj.charAt(index);
- Try this:
  > String test = "Hello";
  > System.out.println(test.charAt(0));
  > System.out.println(test.charAt(4));

➢How would you get the second character?

Looping through a string

- Use method length() to get the number of characters in the string
  - Not a public field like in arrays

```
> for (int i = 0; i < test.length(); i++)
    System.out.println(test.charAt(i));
H
e
l
l
o
```


Strings are immutable

- Strings don't change
  - Instead you get back a new string

```
> String test = "this is a test";
> String result = test.toUpperCase();
> System.out.println(test);
this is a test
> System.out.println(result);
THIS IS A TEST
```


Comparing Strings

- Don't use str1 == str2 to check if they contain the same characters
  - This returns true only if they refer to the same string object
- Use the method equals
  - str1.equals(str2)

```
> String str1 = new String("This is a test");
> String str2 = new String("This is a test");
> System.out.println(str1 == str2);
false
> System.out.println(str1.equals(str2));
true
```


Unicode

- International standard for character representation - 2 bytes / 65,536 chars
  - Characters from all the major world languages
    - Latin, Japanese, Chinese, etc
  - To see the decimal value for a Unicode character

```
int temp = 'a';
System.out.println(temp);
```

  - To create a character from a decimal value

```
char a = (char) 65;
```


String Methods

- Open the Java API http://java.sun.com/j2se/1.6.0/docs/api/index.htm
  - Click on the java.lang package
  - Click on the String class
- Look at the methods
  - Which will return part of a string?
  - Which will return the first index of a list of characters in the string?
  - While will return the last index of a list of characters?
  - Which will remove extra space before and after any other characters?
  - Which will return an array of String objects
    - By chopping the string up into substrings
    - Based on specified delimiters (like spaces or commas)

Exercise

- How would you put the following in a string in Java?
  - She said, "I will see you later".
- Create a short message and encode it using Unicode.
  - http://www.unicode.org/charts/PDF/U0000.pdf
    - The numbers under each character are in hexidecimal
  - Give it to another student to decode it
- How could you pull out “be” from the string “I will be back”?

Working with Delimited Strings

- Sometimes you get information about an object
  - In the form of a delimited string

  Jane Dorda :88, 92, 95, 87, 93, 85

  Mike Koziatek :75, 92, 83, 81, 91, 87

  Sharquita Edwards:91, 93, 95, 92, 94, 99
- Here the delimiters are a colon after the name and commas between the grades

Parsing a Delimited String

- Add another constructor to the Student class
  - That takes a delimited string
    - Name : grade1, grade2, grade3, grade4, grade5
- Use the split method to get an array of Strings
  - First based on the colon delimiter
  - Use trim to clear off any additional space from the name
    - The first element in the returned array
- Use the split method again to get the array of grades as strings
  - Use the comma as the delimiter
- Use Double.parseDouble to translate the grade string into a double value
  - For the grade array

Converting to a Number

- Strings are stored in Unicode format
  - Two bytes per character
- Integers are stored in 4 bytes (32 bits)
- You need to convert a number that is represented as a string into the number representation
- The wrapper classes have methods to do this
  - Integer.parseInt(numStr)

The string “1234” is stored in 8 bytes
With each character taking 2 bytes

00000000|00110001|00000000|00110010
00000000|00110011|00000000|00110011

The integer 1234 is stored in 4 bytes

00000000|00000000|00000100|11010010

Constructor that takes a Delimited String

```
public Student(String delimString,
               String nameDelim,
               String gradeDelim)
 {
   // split string based on name delimiter
   String[] splitArray = delimString.split(nameDelim);
   this.name = splitArray[0].trim();

   // get the grade string and break it and convert to double
   String grades = splitArray[1];
   String[] gradeStrArray = null;
```


Constructor - continued

```
   if (grades != null)
   {
     gradeStrArray = grades.split(gradeDelim);
     this.gradeArray = new
                  double[gradeStrArray.length];
     for (int i = 0; i < gradeStrArray.length; i++)
       this.gradeArray[i] =
          Double.parseDouble(gradeStrArray[i]);
   }
 }
```

Testing the Constructor

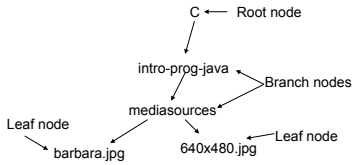
- Write a main method that will create a Student object and initialize the name and grade array
  - From a delimited string
- Run the main method from DrJava
- Use the Debugger to walk through the constructor

Files

- Files are named collections of bytes on your hard disk
  - Often have a base name and suffix
    - Like barbara.jpg
- Are grouped into directories
  - A directory can have other directories in it
  - There is often a root directory
    - Like the C: drive on Windows or \User on Macs
- A path is the list of all the directories from the root to the file
  - And includes the file's base name and suffix

Picture of a Path Tree

- Drawing a path yields an upside down tree
  - With the root at the top
  - And the leaves at the bottom
- C:\intro-prog-java\mediasources\640x480.jpg



Reading from a File

- When we read from a file
  - We copy data from disk into memory
- Things can go wrong
  - The file may not exist
  - The disk may go bad
  - The file may change while we are reading it
- In Java when things go wrong an java.lang.Exception object is created

Possible Exceptions

- What would happen if we try to read from a file that doesn't exist?
  - We would get a FileNotFoundException
- What would happen if we try to read past the end of the file?
  - IOException
- What would happen if the file changes while we are reading it?
  - IOException
- The code won't compile unless we
  - Either handle the exception with a try and catch
  - Or throw the exception

Generating Runtime Exceptions

- Try the following in the Interactions Pane
  - String test = null;
  - test.length();
    - What exception do you get?
- Try this
  - int sum = 95;
  - int num = 0;
  - System.out.println(sum/num);
    - What exception do you get?

The Call Stack

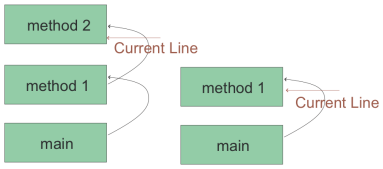
- Execution begins in the main method
  - That method creates objects and invokes methods on them
    - When execution jumps to another method an entry is added to the call stack
      - The current method
      - Where the call occurred in that method
    - When a method finishes executing
      - The entry is removed from the call stack
      - And execution returns to the next line in that method
      - Until the main method finishes

Example Call Stack

- Remove the check for gradeArray == null in the getAverage method
  - And run the main method
- This says a null pointer exception occurred
  - at line 109 in the method getAverage in the Student class
    - Which was called from method toString at line 120

```
java.lang.NullPointerException:
at Student.getAverage(Student.java:109)
at Student.toString(Student.java:120)
at java.lang.String.valueOf(String.java:2131)
at java.io.PrintStream.print(PrintStream.java:462)
at java.io.PrintStream.println(PrintStream.java:599)
at Student.main(Student.java:129)
```


Call Stack Picture



Turning on Line Numbers in DrJava

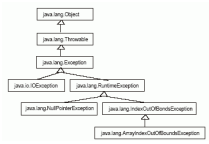
- To see the line numbers in DrJava click on
  - Edit then on
  - Preferences and then on
  - Display Options and
    - Check the Show All Line Numbers checkbox in the Preferences window.
    - Then click on OK.

Exceptions

- Exceptions are objects of the class java.lang.Exception
  - Or are objects of classes that inherit from Exception
- There are two types of exceptions
  - Checked and Unchecked
    - Checked exceptions must be caught or thrown
      - IOException and FileNotFoundException
    - Unchecked exceptions do not have to be caught or thrown
      - NullPointerException, ArrayIndexOutOfBoundsException

Exception Inheritance Tree

- All classes inherit from Object
- All Exception classes inherit from Exception



Catching exceptions

- Use a try, catch, and finally block
- The try block encapsulates the code that can cause an exception
- The catch block is what to do if an exception happens
  - Catches all exceptions of the specified type and subclasses of that type
- The finally block will always be executed
  - Whether or not an exception occurs

Importing Classes To Read From Files

- To read from a file we will use classes in the java.io package
  - Which means that we will need to use import statements
    - Or use the full names of classes
      - package.Class
- Import statements go before the class declaration in the file
  - import package.Class;
    - Allows the short name to be used for just the mentioned class
  - import package.*;
    - Allows the short name to be used for any class in this package

Reading from a File

- To read from a character based file
  - Use a FileReader object
    - This class knows how to read character data from a file
  - With a BufferedReader object
    - To buffer the data as you read it from the disk
      - Into memory
    - Disks are much slower to read from than memory
      - So read a big chunk from disk into memory
        - » And then read from the chunk in memory as needed

Using Try, Catch, and Finally Blocks

- Wrap all code that can cause a checked exception in try, catch (and optionally finally) blocks

```
try {
  // code that can cause an exception
} catch (ExceptionClass ex) {
  // handle this exception
} catch (ExceptionClass ex) {
  // handle this exception
} finally {  // optional
  // do any required clean up
}
```


SimpleReader - Example Class

```
public class SimpleReader
{
  /**
   * Method to read a file and print out the contents
   * @param fileName the name of the file to read from
   */
  public void readAndPrintFile(String fileName)
  {
    String line = null;

    // try to do the following
    try {

      // create the buffered reader
      BufferedReader reader =
        new BufferedReader(new FileReader(fileName));
```


Simple Reader - Continued

```
// Loop while there is more data
while((line = reader.readLine()) != null)
{
  // print the current line
  System.out.println(line);
}

// close the reader
reader.close();
```

Simple Reader - Continued

```
  } catch(FileNotFoundException ex) {
    SimpleOutput.showError("Couldn't find " + fileName +
                           " please pick it.");
    fileName = FileChooser.pickAFile();
    readAndPrintFile(fileName);
  } catch(Exception ex) {
    SimpleOutput.showError("Error reading file " + fileName);
    ex.printStackTrace();
  }
 }

 public static void main(String[] args)
 {
   SimpleReader reader = new SimpleReader();
   reader.readAndPrintFile("test.txt");
 }
}
```


Key Points

- Notice that we put all 'normal' code in the try block
  - This handles the case when everything goes right
- We can catch more than one exception
  - Here we caught FileNotFoundException
    - And used the FileChooser to have the user pick the file
      - And then called the method again
  - Catching Exception will catch all children of Exception as well
    - So make it the last Exception you catch
- Finally blocks are not required
  - But they will always execute if there is an exception or not

java.util.ArrayList

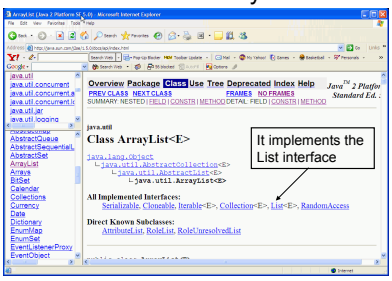
- An ArrayList object is an array that can grow or shrink as needed
  - Do we always know how many students can be in a class period?
    - If we create an array for more than we have, we waste space
    - If we try to add a student past the end of the array
      - We get an exception
- Use an ArrayList when you don't know how many of something you need

ArrayList Methods

- Look in the Java API for ArrayList
  - Open the class java.util
  - Click on the class ArrayList
  - What methods let you add an object to the ArrayList?
  - What method lets you get an object from the ArrayList?
  - What method tells you how many things are in the ArrayList?
  - What method lets you remove an object from an index in the ArrayList?

An ArrayList is a List

- Look at the API for ArrayList



Interfaces

- An interface is an abstract class that only has public abstract methods and/or constants
  - An abstract method is a method without any code
    - like the compareTo method in Comparable

```
package java.lang;
public interface Comparable {
  public int compareTo(Object o);
}
```

  - Useful for saying what methods a class needs to have
    - The class that implements the interface must provide the code for the methods in the interface
    - The String class implements the Comparable interface

Declare variables as interface types

- When you create a variable that implements an interface you should declare it to be of the interface type
  - List studentList = new ArrayList();
- Many classes implement the same interface
  - So this gives you the flexibility to change the actual class with a minimum number of changes

Exercise

- Modify the Student class to implement the Comparable interface
  - public class Student implements Comparable
- Try to compile it
  - It won't compile till you add the compareTo method
    - And provide code for it
  - Create a compareTo method to compare the current student to a passed in one
    - You can compare the names
      - Since the String class implements Comparable

What the compareTo Method Returns

- The compareTo method returns an integer
  - Negative if the current object is less than the passed object
  - 0 if they are equal
  - Positive if the current object is greater than the passed object
- You need to cast from Object to Student
  - Before you can compare names
    - Student testStudent = (Student) o;

Final compareTo Method

- If o is null this will cause a NullPointerException
- If o isn't a Student it will cause a ClassCastException
- This will say two students with the same name are equal
  - Okay for sorting by name

```
public int compareTo(Object o)
{
  Student testStudent = (Student) o;
  return this.name.compareTo(testStudent.name);
}
```


Decoupling Classes

- One of the goals of Object-Oriented Programming
  - Is to decouple classes
    - Make class A not dependant on class B so that you can change out B for C
  - Interfaces let you do this
    - Variables can be declared to be the interface type
      - List studentList = null;
    - Then any class that implements this interface can be used
      - studentList = new ArrayList();

Other Interfaces

- LEGO bricks have a common interface
  - Makes it easy to connect two bricks
  - It doesn't matter what you connect as long as the interface is the same
- A USB interface
  - Allows you to connect different devices to your computer
    - USB drive, camera, etc
    - As long as they use the USB interface

ArrayList Exercise

- In the ClassPeriod class
  - Modify the studentArray to be a studentList
    List studentList = new ArrayList();
- Change all the methods that use an array to use a list
  - Cast back to Student when you pull the object out of the list

```
public Student getStudent(int index)
 {
   return (Student) this.studentList.get(index);
 }
```


Collections Store Objects

- Why do we need to cast the Student object back to Student when we pull it back out of a list?
  - A list is a collection of objects
    - We need to tell the compiler that it is really a Student object
- Or, if we are using Java 1.5 we can use generics
  - List<Student> studentList = new ArrayList();
    - Then we wouldn't need to cast to Student

Add a Constructor that takes a File Name

- Let's add a constructor to the ClassPeriod class that takes a file name to read the student information from

```
public ClassPeriod(String name, int num, String
    fileName)
{
   this.teacherName = name;
   this.periodNumber = num;
   loadStudentsFromFile(fileName);
}
```


Create Students from File Exercise

- Write the method loadStudentsFromFil e(fileName);
  - It will be similar to the readAndPrintFile method in SimpleReader
- It will loop reading from the specified file
  - Until the line that is returned from the reader is null
- It will use each line that it reads to create a Student object
  - Using the constructor that takes a delimited string
- It will add each new student to the studentList

Testing the Method

```
public static void main(String[] args)
 {
   ClassPeriod period =
     new ClassPeriod("Ms. Clark",5,"student.txt");

   // print info about the class period
   System.out.println(period);

   // print info for each student
   for (int i = 0; i < period.studentList.size(); i++)
     System.out.println("Student " + i + " is " +
                    period.getStudent(i));

 }
```


Writing to a File

- Very similar to reading from a file
  - But use FileWriter and BufferedWriter
  - Write out things with the method
    - write(string);
  - Force a new line with
    - newLine();
      - Different systems use different ways to end a line
        - » Macs versus Windows
      - This will write it out in away that works for the current system

SimpleWriter

```
public class SimpleWriter
{
  /**
   * Method to write a silly file
   */
  public void writeSillyFile()
  {
    try {
      // try to open the buffered writer
      BufferedWriter writer =
        new BufferedWriter(new FileWriter("silly.txt"));

      // write out the file
      writer.write("Here is some text.");
      writer.newLine();
      writer.write("Here is some more.");
      writer.newLine();
```


Simple Writer - Continued

```
      writer.write("And now we're done.");
      writer.newLine();
      writer.newLine();
      writer.write("THE END");
      writer.close();
    } catch (Exception ex) {
      System.out.println("Error during write of silly.txt");
    }
  }

  public static void main(String[] args)
  {
    SimpleWriter writer = new SimpleWriter();
    writer.writeSillyFile();
  }

}
```


Generating a Form Letter

- You can use a method to personalize a form letter
  - By passing in the title, last name, city and eye color
  - And writing out the letter with these items inserted at the appropriate places

Form Letter Generator Class

```
import java.io.*;
/**
 * Class used to generate form letters
 * @author Barbara Ericson
 */
public class FormLetterGenerator
{

  /**
   * Method to generate a form letter
   * @param title the person's title (Mr., Mrs., Dr.)
   * @param lastName the last name for the recipient
   * @param city the name of the city for the recipient
   * @param eyeColor the eye color of the recipient
   */
  public void writeLetter(String title, String lastName,
                          String city, String eyeColor)
  {
```


Form Letter Generator Class - Cont

```
String fileName = lastName + "Letter.txt";

// try to open the file and write to it
try {

  // create the buffered writer to use to write the file
  BufferedWriter writer =
    new BufferedWriter(new FileWriter(fileName));

  // write the beginning of the letter
  writer.write("Dear " + title + " " + lastName + ", ");
  writer.newLine();
  writer.newLine();
```


Form Letter Generator Class - Cont

```
    // write the body of the letter
    writer.write("I am writing to remind you of the offer");
    writer.newLine();
    writer.write("that we sent to you last week.  Everyone in");
    writer.newLine();
    writer.write(city +
                 " knows what an exceptional offer this is!");
    writer.newLine();
    writer.write("(Especially those with lovely eyes of " +
                 eyeColor + "!)");
    writer.newLine();
    writer.write("We hope to hear from you soon.");
    writer.newLine();
    writer.newLine();
```

Form Letter Generator Class - Cont

```
    // write the ending
    writer.write("Sincerely,");
    writer.newLine();
    writer.write("I. M. Acrook");

    // close the file
    writer.close();
   } catch (Exception ex) {
    System.out.println("Error writing to " + fileName);
   }
  }

  public static void main(String[] args)
  {
   FormLetterGenerator formGenerator = new FormLetterGenerator();
   formGenerator.writeLetter("Mr.","Guzdial","Decatur","brown");
  }

}
```


Write a File Exercise

- Create another method to write a form letter
  - Have it take the high temp, low temp and, chance of rain
  - Have it print out the following:
  - Todays high will be (high temp) and the low will be (low temp). There is a (chance of rain) % chance of rain

Modifying a Program

- You can read the source code from a file
  - And change it in some way
    - And write it back out
- Just read each line and look for a string that you want to change
  - If the current line doesn't have the string to change then just add it to a list of lines
  - If the current line has the string to change then change it and add it to a list of lines
- When you reach the end of the file
  - Write out the lines in the list

Modifying the Cartoon Class

- We will create a method to change the text passed to the addWordBalloon method
  - In the main method
  - Use indexOf to look for the text

FileModifier Class

```
import java.util.*;
import java.io.*;

/**
 * Class to demonstrate using a program to modify another program
 * @author Barb Ericson
 */
public class FileModifier
{
  /**
   * Method to modfiy the first string in a method to
   * be the passed changed text
   * @param fileName the file name for the class to modify
   * @param textToChange the text to change
   * @param changedText the new text to use for the text to
   * change
   */
  public void modifyFile(String fileName,
                         String textToChange,
                         String changedText)
```


File Modifier - Cont

```
{
   List lineList = new ArrayList();
   String line = null;
   int pos = 0;

   // try the following
   try {

     // open the file to read from
     BufferedReader reader =
       new BufferedReader(new FileReader(fileName));

     /* loop while there are more lines in the file
      * and we haven't found the text to change yet
      */
     while((line = reader.readLine()) != null &&
           line.indexOf(textToChange) < 0)
       lineList.add(line);
```

File Modifier - Cont

```
/* If we get there we either ran out of lines or we
 * found the text to change
 */
if (line != null)
{
  // get the position of the text to change
  pos = line.indexOf(textToChange);

  // modify the string
  lineList.add(line.substring(0,pos) +
            changedText +  line.substring(pos + textToChange.length()));

  // loop till the end of the file adding the rest
  while ((line = reader.readLine()) != null)
  {
    lineList.add(line);
  }
}
```


File Modifier - Cont

```
// now close the file
reader.close();

// create a writer to write out the file
BufferedWriter writer =
  new BufferedWriter(new FileWriter(fileName));

// loop writing out the lines
for (int i = 0; i < lineList.size(); i++)
{
  writer.write((String) lineList.get(i));
  writer.newLine();
}
```


File Modifier - Cont

```
    // close the writer
    writer.close();
  } catch (FileNotFoundException ex) {
    SimpleOutput.showError("Couln't find file " + fileName);
    fileName = FileChooser.pickAFile();
    modifyFile(fileName,textToChange,changedText);
  } catch (Exception ex) {
    SimpleOutput.showError("Error during read or write");
    ex.printStackTrace();
  }
 }
```


File Modifier – Main Method

```
// Main method to run
public static void main(String[] args)
{
  FileModifier fileMod = new FileModifier();
  String file =
    "C:\\intro-prog-java\\bookClassesFinal\\Cartoon.java";
  fileMod.modifyFile(file,
                     "Just Horsing Around!",
                     "What's up, Wilbur?");
}
}
```


Searching for data in a file

- Sequences of nucleotides are in a file
  - Name of a parasite followed by nucleotide sequence (DNA)
  - Search for a sequence with > starting data for a parasite

```
>Schisto unique AA825099
gcttagatgtcagattgagcacgatgatcgattgaccgtgagatcgacga
gatgcgcagatcgagatctgcatacagatgatgaccatagtgtacg
>Schisto unique mancons0736
ttctcgctcacactagaagcaagacaatttacactattattattattatt
accattattattattattactattattattattattactattattta
ctacgtcgctttttcactccctttattctcaaattgtgtatccttccttt
```


Algorithm for search

- Open the file
- Loop reading the file a line at a time until we find the sequence or reach the end of the file
  - Append the new line to a string with “\n” to show the end of a line
- Once we find the sequence look backwards for the start of the sequence
  - Find the name between > and “\n”

Class SequenceSearcher

```
import java.io.*;
/**
 * Class that searches a file for a given sequence and reports
 * on the name where that sequence was found
 * @author Barb Ericson
 */
public class SequenceSearcher
{
  /**
   * Method to search for a given sequence and then
   * report on the name
   */
  public String getNameForSequence(String fileName, String seq)
  {
```


SequenceSearcher - cont

```
    String info = "";
    String line = null;
    String name = null;

    // try the following
    try {
      // read from the file
      BufferedReader reader =
        new BufferedReader(new FileReader(fileName));

      // loop till end of file or find sequence
      while ((line = reader.readLine()) != null &&
          line.indexOf(seq) < 0)
      {
```


SequenceSearcher - cont

```
        // add to string with new line character
        info = info + line + "\n";
      }

      // if get here either end of line or we found the sequence
      if (line != null)
      {
        // look backward for the last >
        int firstIndex = info.lastIndexOf('>');

        // look forward from the > for the new line character
        int secondIndex = info.indexOf('\n',firstIndex);
```


SequenceSearcher - cont

```
        // get the name between the > and new line
        name = info.substring(firstIndex+1,secondIndex);
      }
    } catch (FileNotFoundException ex) {
      SimpleOutput.showError("Couldn't find file " + fileName);
      fileName = FileChooser.pickAFile();
      getNameForSequence(fileName,seq);
    } catch (Exception ex) {
      SimpleOutput.showError("Error during read or write");
      ex.printStackTrace();
    }
    return name;
  }
```


Main method for testing

```
public static void main(String[] args)
  {
    SequenceSearcher searcher = new SequenceSearcher();
    String fileName = FileChooser.getMediaPath("parasites.txt");
    String seq = "ttgtgta";
    String name = searcher.getNameForSequence(fileName,seq);
    if (name == null)
      System.out.println("The sequence " + seq +
                         " wasn't found in " + fileName);
    else
      System.out.println("The sequence " + seq +
                         " was found in " + name);
  }
}
```


Challenge

- Find some interesting data on a web page
  - Like the current temperature
  - View the source and save the source as a text file
  - Write a class that reads the desired data from the file

Adding text to all files in a directory

- Professional photographers add text to all their photos on the web
  - To make it harder for people to "steal" them
- You can write a class to do this!

```
> import java.io.File;
> File dir = new File("C:\\intro-prog-java\\mediasources\\");
> String[] pathArray = dir.list();
> for (int i=0; i < 5; i++) System.out.println(pathArray[i]);
swan.jpg
MattScotland.jpg
twoSwans.jpg
kidsTree.jpg
redDoor.jpg
```


Class DirectoryWorker

```
import java.io.*;
/**
 * Class to work with files in a directory
 */
public class DirectoryWorker
{
 /**
  * Method to add a string to every picture in directory
  * @param dir the name of the directory
  * @param text the text of the string to add
  */
 public void addStringToPictures(String dir, String text)
 {
```


DirectoryWorker - cont

```
  String name = null;

  // create the object that represents the directory
  File file = new File(dir);

  // Get the array of names in the directory
  String[] nameArray = file.list();

  // loop through the names
  for (int i = 0; i < nameArray.length; i++)
  {
   name = nameArray[i];
```


DirectoryWorker - cont

```
    // if this is a picture file
    if (name.indexOf(".jpg") >= 0)
    {
     // create the picture object
     Picture p = new Picture(dir + name);

     // add the text to the picture
     p.drawString(text, 5,
              p.getHeight() - 50);

     // save the changed picture to a file
     p.write(dir + "titled-" + name);
    }
   }
 }
```


Testing DirectoryWorker

```
 public static void main(String[] args)
  {
    DirectoryWorker worker = new DirectoryWorker();
    worker.addStringToPictures(
      "c:\\intro-prog-java\\mediasources\\",
      "Copyright 2009");
  }
}
```


Challenge

- Add another method to the DirectoryWorker class to turn all pictures in a directory into grayscale
- Write them out with –gray appended to the original name

Randomly Generated Text

- Some magazines titles are very strange
  - Elvis runs a restaurant.
- You can randomly generate combinations of nouns, verbs and phrases to make your own silly sentences.
- Using the class java.util.Random
  - To create a random number generator
  - And methods nextDouble and nextInt to get random numbers

Try Out the Random Number Generator

- In the interactions pane
  - Create a random number generator
  - Random randNumGen = new Random();
- Get a random double between 0 and 1
  - double num = randNumGen.nextDouble();
- Get a random integer between 0 and the (passed number – 1)
  - int steps = randNumGen.nextInt(11);
  - Will randomly get from 0 to 10

SentenceGenerator Class

```
import java.util.Random;

/**
 * Class to generate sentences
 * @author Barb Ericson
 */
public class SentenceGenerator
{
  /////////// fields /////////////
  private String[] nounArray = {"Mark", "Adam", "Angela",
    "Larry", "Jose", "Matt", "Jim"};
```


SentenceGenerator Class - Cont

```
  private String[] verbArray = {"runs", "skips", "sings",
    "leaps", "jumps", "climbs", "argues", "giggles"};
  private String[] phraseArray = {"in a tree", "over a log",
    "very loudly", "around the bush",
    "while reading the newspaper",
    "very badly", "while skipping",
    "instead of grading"};
  private Random randGen = new Random();
```


SentenceGenerator Class - Cont

```
  ////////////// methods ////////////////////////////////
  /**
   * Method to generate a random sentence
   * @return a random sentence
   */
  public String generateRandomSentence()
  {
    String sentence =
      nounArray[randGen.nextInt(nounArray.length)] + " " +
      verbArray[randGen.nextInt(verbArray.length)] + " " +
      phraseArray[randGen.nextInt(phraseArray.length)] + ".";

    return sentence;
  }
```


SentenceGenerator Class - Cont

```
public static void main(String[] args)
  {
    SentenceGenerator sentenceGen =
      new SentenceGenerator();
    for (int i = 0; i < 5; i++)
      System.out.println(
        sentenceGen.generateRandomSentence());
  }
}
```

Read Field Data Exercise

- Create another constructor for the SentenceGenerator class
  - That takes 3 files names
    - One for the nouns
    - One for the verbs
    - One for the phrases
  - And reads each file into an ArrayList
    - Read an item on each line
  - And then uses the toArray method of ArrayList to convert from an ArrayList to an array

Computer Networks

- Computers communicate with each other over networks
- Networks use agreements about how to communicate
  - How to address computers?
  - How data is represented?
  - How data will be transferred?
  - What protocol will the computers use to pass the data?
    - Rules for doing the activity

Internet Agreements

- How to address the computers
  - Use Internet Protocol (IP) addresses
  - x.x.x.x where 0 <= x <= 255
  - Domain name servers translate domain names to addresses
    - www.cnn.com is http://64.236.24.20
- Data will be passed in packets
  - Like an envelope with from and to IP addresses and the number of bytes in the packet
- How packets are routed
  - Allow for some machines to be down
    - In case of a nuclear attack
- Protocols
  - FTP (File Transfer Protocol)
  - POP and SMTP (Mail transfer protocols)

Web Agreements

- Way to specify a location of a resource on the web
  - URL - Uniform Resource Locator
    - Has protocol to use, domain name of the server, and path to resource on the server
    - http://www.cc.gatech.edu/index.html
- How to serve documents
  - HTTP – HyperText Transfer Protocol
  - HTTPS – More secure protocol
- How the documents are formatted
  - HTML – HyperText Markup Language
    - Used for Web pages

Reading a Web Page

- Some programs gather information from several web pages
  - They pull out the information they want
    - And then display it in a new format
    - Like google's news page
- You can do this too! We can write a method to find the current temperature in a web page
  - And display it to the user

Reading from the Web

- We need to know where the web page is
  - URL (Uniform Resource Locator)
  - Gives protocol
- We need something that can read from a URL
  - And give us the bits
  - And we need it to be character
  - And we want to buffer it for more efficient reading

Reading from the Web

- To represent a URL
  - We will use java.net.URL
- To get something that can read from a URL
  - We will use the openStream method to get an InputStream from a URL
- To convert the InputStream object into something that works with characters
  - We will create an InputStreamReader
- To buffer the data in memory as we read it
  - We will use a BufferedReader

Each Class has a Responsibility

- In Object-oriented programming each object should be responsible for one major thing
  - Like representing a URL
- You create objects of different classes to work together to accomplish a task
- In a well designed program
  - No one object does all the work
  - Classes are easy to reuse

getTempFromNetwork Method

```
public String getTempFromNetwork(String urlStr)
{
  String temp = null;
  String line = null;
  String seq = "&ordm";

  try {

    // create a url
    URL url = new URL(urlStr);

    // open a buffered reader on the url
    InputStream inStr = url.openStream();
    BufferedReader reader =
      new BufferedReader(new InputStreamReader(inStr));
```


getTempFromNetwork Method - Cont

```
    // loop till end of file or find sequence
    while ((line = reader.readLine()) != null &&
          line.indexOf(seq) < 0)
    {}

    // if there is a current line
    if (line != null)
    {
      // find the temperature
      int degreeIndex = line.indexOf(seq);
      int startIndex = line.lastIndexOf('>',degreeIndex);
      temp = line.substring(startIndex + 1, degreeIndex);
    }
```


getTempFromNetwork Method - Cont

```
    } catch (FileNotFoundException ex) {
      SimpleOutput.showError(
              "Couldn't connect to " + urlStr);
    } catch (Exception ex) {
      SimpleOutput.showError
            "Error during read or write");
      ex.printStackTrace();
    }
    return temp;
  }
```


How it Works

- First we create some variables that we will need
  - And set temp to null
- This method reads a line at a time from the network address while
  - the line isn't null and
  - doesn't have the sequence we are looking for in it
- After this we check if the loop stopped because the line was null
  - If not we get the starting index of the sequence we were looking for
  - And we look backwards from there for the previous >
  - Then we get the temperature from between the > and the sequence
- Finally we return the value in temp

Read from Web Page Exercise

- Find a web page with some interesting data on it
- Use the view source button to see what the HTML looks like for the page
- Find some way to identify the data that you want
- Write a method to read from that URL and return the desired data
- Write a main method to test it

Summary

- All media can be saved as text
- Text in Java is stored in String objects
  - java.lang.String
- Exceptions are exceptional events
  - Checked exceptions must be caught or thrown
- Classes in the java.io package can be used with files
- Classes in the java.net package can be used to work with networks
- While loops will loop till a condition is false