CS 325 - Class 15

- Today
  - More about strings in Java
  - Input with Java: console and graphical
  - File I/O in Java
  - Using multiple classes in Java
- Announcements
  - Continue working on Project 3

More about Strings in Java

String Comparison

- Do NOT compare strings with ==
  - Use either **equals** or **equalsIgnoreCase**

```
if (command.equals("Hi")) …
if (s.equalsIgnoreCase("A")) …
```

- More information on strings
  - javafaq.nu/java-article440.html
  - javafaq.nu/java-article1072.html

Basic String Methods

- char charAt(int)
- boolean equals(object)
- int indexOf(char)
- int indexOf(String)
- String substring(int)
- String substring(int, int)

Class Exercises

- Write a Java program that takes one command line argument and determines whether or not it is a palindrome (reads the same forward and backward). Print "yes" or "no" depending on whether or not it is a palindrome.
- Example:
  - "civic"
- Example:
  - "redder"

- Write a Java program that takes two command line arguments and checks to see if all the characters in the first argument appear in the second argument. Print "yes" or "no" accordingly.
- Example:
  - Arg1 = "aeiou"
  - Arg2 = "thequickbrownfoxjumps overthelazyolddog"

Java Programming

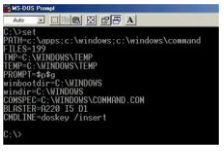
- Think about all the programs that we have written so far
  - Hello world
  - Print shapes
  - Print array backwards
  - Command line argument is a palindrome
  - Random numbers
- None of them used any input during execution

Input in Java

- In C++ we learned how to read from the console

  cin >> *vars;*

- Console input means console applications



- We prefer to build graphical applications, ***not*** console applications

- Will look at both in Java
  - Our focus is on graphical application input

Graphical input/output in Java

/fs3/rborie/325/samples/input1.java

```
import javax.swing.JOptionPane;
public class input1 {
   public static void main ( String args[ ] ) {
      String name, greeting;
      name = JOptionPane.showInputDialog("Enter Your Name:");
      greeting = "Hi " + name + ", Roll Tide!";
      JOptionPane.showMessageDialog(null, greeting,
                  "Better than console windows",
                  JOptionPane.INFORMATION_MESSAGE);
      System.exit(0);
   }
}
```

Using the Java Swing package for graphical I/O

JOptionPane methods

- **showInputDialog**
  - Prompts for an input
  - ***Always*** returns a string
- Always get a string
  - Can convert to int or double

```
int num1 = Integer.parseInt(string);
double num2 =
        Double.parseDouble(string);
```

```
public static String
   showInputDialog
       (Object message)
       throws
          HeadlessException
```

- **showMessageDialog**
  - Prints a message
  - Fields for title and message type
  - messageType includes:
    - ERROR_MESSAGE
    - INFORMATION_MESSAGE
    - WARNING_MESSAGE
    - QUESTION_MESSAGE
    - PLAIN_MESSAGE

```
public static void
  showMessageDialog
   (Component parentComponent,
    Object message,
    String title,
    int messageType)
       throws HeadlessException
```


Class Exercises

USING JAVA SWING

Need Java 1.5 or higher

- Write a Java program that reads in two names (your first name and then your last name) and prints out the entire name (concatenating first plus space plus last)
- Write a Java program that reads in a name and prints it out backwards
- Write a Java program that reads in two numbers and prints out the average of them
  - What happens when you enter input values that are not legal numbers?

```
$ java  -version
java version "1.5.0_12"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_12-b04)
Java HotSpot(TM) Client VM (build 1.5.0_12-b04, mixed mode)
```


Console input in Java

/fs3/rborie/325/samples/input2.java

```
import java.util.*;

public class input2 {
    public static void main ( String args[ ] ) {

        Scanner consoleInput = new Scanner(System.in);

        System.out.print("Please enter your name: ");
        String name = consoleInput.nextLine( );

        System.out.print("Please enter your age:  ");
        int age = consoleInput.nextInt( );

        System.out.print("Please enter your gpa:  ");
        double gpa = consoleInput.nextDouble( );

        System.out.println( );
        System.out.println(name + " is " + age + " years old with a gpa of " + gpa);
    }
}
```


Class Exercises (repeated)

USING CONSOLE INPUT

- Write a Java program that reads in two names (your first name and then your last name) and prints out the entire name (concatenating first plus space plus last)
- Write a Java program that reads in a name and prints it out backwards (reverses the name)
- Write a Java program that reads in two numbers and prints out the average of these two values
  - What happens when you enter input values that are not legal numbers?

File I/O in Java

- Very similar to what you have seen before with C++
- Example at the right displays a file on the console
  - Checks to make sure the user invoked the program correctly
  - Prints the name of the file and then a row of "=" signs as an underline
  - Within a try block
    - Opens the file
    - Reads (and prints) lines in the file as long as more exist
  - Has a block that catches errors in case of problems

```
import java.io.*;
import java.util.*;
public class file {
   public static void main( String [ ] args ) {
      // print error if no filename specified
      if (args.length == 0) {
            System.out.println("usage: java file FILENAME");
            return;
      }
      // print out the file that we are going to display
      System.out.println("Displaying contents of " + args[0] );
      for (int a=0; a< 27+args[0].length(); ++a)
            System.out.print("=");
      System.out.println();
      // need a try block since we are doing I/O
      try {
            // allocate a scanner for our file
            Scanner fileIN = new Scanner( new File (args[0]) );
            // while lines exist in the file
            while ( fileIN.hasNext() ) {
                  // get the next line and print it to the console
                  String str = fileIN.nextLine();
                  System.out.println( str );
            }
      }
      // print a message if things go wrong
      catch ( IOException e)
            { System.out.println("File problems"); }
   }
}
```


Class Exercises

- This program exists at **/fs3/rborie/325/samples/file.java**
- Modify this program so that the program prompts the user for a filename (rather than read it from the command line)

Scaling up in Java

- Writing a Java program
  - Actually defining a class

```
public class XXXX {
  ...
}
```

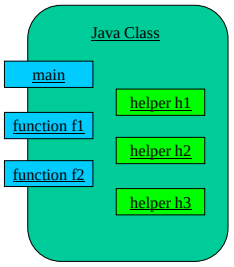
  - Execution starts at the "main" within the class

```
public static
  void main
  (String [ ] args)
      {
          ...
      }
```

- Can add other methods within the class. Why?
  - Modular programming
  - Stepwise refinement
  - Decomposition
- Most programs will have
  - Main method
  - Several helper methods

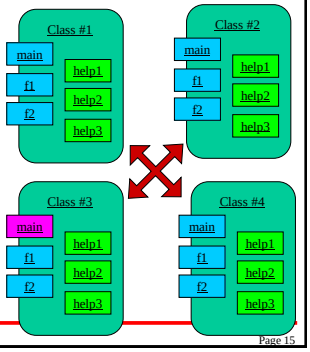
Scaling up in Java (slide #2)

- For a given class
  - Define private methods to assist in the tasks demanded of the class
    - String getHostname
    - String getFilename
  - Define public methods visible to clients
    - Think of a List class
      - Insert
      - Remove
      - Find
      - Length



Scaling up in Java (slide #3)

- Multiple classes that interact with each other
  - Each class contains a "main" routine
    - Used for "unit testing" of that class
    - Invoking that class' main just tests that class
  - One class contains a main routine that invokes the overall application



Class Exercises

- The directory **/fs3/rborie/325/samples** has 3 java files
  - Location.java
  - City.java (which uses the class Location)
  - app.java (which uses the class City & Location)
- The "main" method in Location.java and City.java is simply for debugging
- The "main" in app.java runs the program

- Move these into a subdirectory on your account and compile them. Questions:
  - How do you run the application?
  - How could you test each other class?
  - What happens if you compile app.java first?

A few more remarks about Java

- Method signatures can start with:
  - public static void …
  - static String …
  - static int …
  - public int …
  - public void …
- Public methods
  - Visible outside the class
  - Other classes can invoke this method on an object of that type
- Static methods
  - Usually at least one such method exists for each class (**main**)
  - If not static, then called an "instance method"
    - Can access instance variables directly

Object-Oriented Prog. with Java

- No functions can be declared outside all classes
  - The only functions in Java are methods (both instance methods and static methods)
- All instance methods are automatically virtual
  - It's not required (or permitted) to declare them as "virtual"
- All object variables are automatically polymorphic
  - Always uses implicit pointers
  - No dereferencing (Java has no –> or unary * operators)
- Instance method calls are always resolved using late method binding

OOP with Java: Example

```
class A {
   public int m( ) { return 1; }
};
class B extends A {          // use “extends” to derive from a base class
   public int m( ) { return 2; }
};
class Test {
   public static void main (String[ ] args) {
            A x;                          // the initial value of object x = null
            x = new A( );
            System.out.println (x.m( ));      // output is 1
            x = new B( );
            System.out.println (x.m( ));      // output is 2
   }
}
```

Page 19