Writing Real Code

Variables
Gotta’ Do These Details

- This won’t be the most exciting part of the course.
  - But we need these details.
  - Will make your life *much* easier later.
  - Trust me.

- Promise cool stuff later. 😊
Variables and Types

- **Primitive types (depends on language)**
  - int: integer 5
  - double: floating point number 5.329
  - char: character ‘h’
  - boolean: true or false

- **Complex types**
  - string: collection of char “hello there”
  - date: collection of int 7/4/1976
  - airplane: collection of collections!
  - fifthLevelOgre: collection of collections!
More Data Types

Some languages have more/less data types

- Java
  - byte, short, int, long
  - float, double
  - char
  - boolean

- C
  - short, long, unsigned, int
  - float, double
  - char

- JavaScript
  - none!

Most commonly used are in black.
Declaring Variables (part 1)

- A variable is any name
  - firstLetterOfName, x, y, happy
  - speedLimit, numberOfApples, interestRate
  - person4, rat_population3, jsdhg45

- Can you guess the type?
  - A good name makes it easy

- If we didn’t tell the program what “jsdhg45” was, it would never be sure.
  - crash, burn, ugly program non-compile accident
  - except JavaScript and other “non-typed” languages
    - they don’t care!
Declaring Variables (part 2)

Syntax:
- datatype variable1, variable2, variable3;

Example:
- int numberOfApples;
- double interestRate, x;
- boolean happy, dead, passingCourse;

Warning: case sensitive!
Declaring Variables (part 3)

- **Generic name rules** *(differs with language)*
  - Case sensitive.
  - Can only use letters and numbers.
    - Numbers can’t go first
  - $ is ok, but discouraged *(has special use).*
  - _ is ok, but discouraged.

- **Convention**
  - Start with lower case letter
    - Other program “thingies” are upper case first *(classes)*
  - Use real words, run together.
    - Capitalize all but the first word.
    - E.g., numberOfCars, endProgramNow
Assigning Variables (part 1)

- Use "="
  - `int age = 21;`
  - `double gpa = 4.0;`
  - `boolean finishedWithClass = false;`
  - `char homeWorkGrade = 'A';`

Note that the end of a programming statement is finished with a ;

Note that chars are assigned with single quotes.

NOTE: If I don’t specify a language, then I’m talking about Java.
I’ll let you know if it’s another language.
Assigning Variables (part 2)

Can re-assign (change) later in code

```java
int age = 21; //original
age = 22; //new
```

```java
char homeWorkGrade; //just declared
homeWorkGrade = 'C'; //original assign
if(extraCreditGrade == 'A') //just a test!
{
    homeWorkGrade = 'B'; //re-assigned
}
```

```
“==” does not re-assign, just compares!
Only “=” assigns.
```

Comments hide behind //. Comments are not executed by the compiler. Comments are helpful!!!
Assigning Variables (part 3)

What if you do this?

```java
char homeWorkGrade;
homeWorkGrade = 'C';
if(extraCreditGrade = 'A') //assigns!
{
    homeWorkGrade = 'B';
}
```

In C, it assigns ‘A’ to extraCreditGrade.
- Then some C compilers will say, “yeah we assigned it ok”, so must be true.
- So always does stuff inside of brackets – *always* assigned a ‘B’

In Java, will not compile.
- Thank goodness…
public class CalculateGrade
{
    public static void main(String[] args)
    {
        int homeworkGrade, extraCreditGrade;

        System.out.println("Please enter your grade (0 to 100).");
        homeworkGrade = readIntegerValue();  //this won't work yet – just an illustration

        System.out.println("Enter extra credit grade (0 to 100).");
        extraCreditGrade = readIntegerValue();  //this won't work yet – just an illustration

        homeWorkGrade = homeworkGrade + extraCreditGrade;

        if(homeworkGrade < 60)
        {
            //nice instructor. Your grade can't be less than 60
            homeworkGrade = 60;
        }

        System.out.println("Your grade is "+ homeworkGrade );
    }
}
What’s In The Example?

1. Name is informative: CalculateGrade
2. Ignore “public class”, “public static…”
   - Required, but explained later
3. Declare variables
4. Read integer values
   - Won’t work yet!
   - Reading values -- we’ll do soon
5. Re-assign values
6. Informative comment
7. ; after each line of completed code.
   - Not after the “if” because stuff inside is part of statement
   - Think of {...} as part of the statement. Could put ; after {...}. 
Your Turn

- Write code that has the classSize for each of your classes.
  - i.e., declare separate variable for each class

- Assign the class size in the code
  - don’t try to read from keyboard

- Calculate the average class size
  - sum divided by number of classes
Casting (part 1)

What if…

- `int classSize = 'a';` //bad!
- `char yourGrade = 97;` //bad!
- `int runs = 2;` //ok
- `double rbi = 3.4;` //ok
- `runs = rbi;` //bad

- Bombs! Nastiness. Failure.

Assigning wrong types.
Casting (part 2)

Can fix this!

- int runs = 2; //ok
- double rbi = 3.4; //ok
- rbi = (double) runs; //ok

Casting!

Can you guess what value rbi will have? (Answer: 2.0)

Only works when assigning one variable to another
Casting (part 3)

How about this?

- int runs = 2;  //ok
- double rbi = 3.4;  //ok
- runs = (int) rbi;  //ok

Can you guess what value runs will have?
(Answer: 3) Truncation!
ASCII Detour

- All characters on your keyboard are assigned a number.
  - Called ASCII code
  - [ = 91, \ = 92, ] = 93, ^ = 94, _ = 95, ` = 96

- and oddly, ‘0’ = 48
  - ‘1’ = 49
  - ‘2’ = 50
  ...

Lousy, stupid, no good, @!$&! computer geeks…
Casting (part 4)

So can you guess how these will cast?

- char yourGrade;
  int yourScore = 97;

  yourGrade = (char) yourScore;

  Answer: ‘a’

- char yourGrade = ‘A’;
  int yourScore;

  yourScore = (int) yourGrade;

  Answer: 65 (ouch!)
Casting (part 5)

How about this?
- boolean happy = true;
  int status;
  status = (int) happy;

Bomb!
- Can really only cast between numbers.
- (chars are secretly numbers)
  - (ASCII, remember?)
Casting (part 6)

What about…

```java
int i = 3;
double d = i;
```

Aurgh, this works!
- Called implicit casting.
- C is all over this.
- In Java, works for smaller numbers assigned to bigger numbers.

- double ← float ← long ← int ← short ← byte

- `double` to `long`: 8 bytes to 4 bytes: `±2^{63}`
- `long` to `int`: 4 bytes to 2 bytes: `±2^{31}`
- `int` to `byte`: 2 bytes to 1 byte: `-128 to 127`

- DO NOT RECOMMEND. VERY CONFUSING!
public class RoundUp
{
    public static void main(String[] args)
    {
        int num1 = 3;
        double dNum = 2.3;
        char letter = ‘z’; //ascii is 122

        //what does this print?
        System.out.println("letter = "+ letter);

        //what does this print?
        int answer = num1 + (int) letter + (int) dNum;
        System.out.println("answer = " + answer);
    }
}
Strings

But what about entering text?

- Use **String**
  - not a primitive type
  - a collection of chars
  - called a “class”
  - the same as classes we’ve written
    - HelloProgram (this was a class!)
      - written to say hello
    - CalculateGrade (this was a class!)
      - written to calculate your grade
    - String
      - written to deal with text
String myFirstName = “David”;
String myLastName = “Bahr”;

System.out.println(“My first name is ” + myFirstName + “.”);

String fullName = myFirstName + myLastName;
System.out.println(“My full name is ” + fullName + “.”);

String space = “ ”;
fullName = myFirstName + space + myLastName;

System.out.println(“Actually, my full name is ” + fullName + “.”);
Cool String Manipulation

- String is a “class”.
- Classes contain “methods”.
- Methods manipulate stuff.
  - Use a method with a “.”
    - String flyGreeting = “Buzz off!”
      String loudGreeting = flyGreeting.toUpperCase();
      System.out.println(loudGreeting);

      Output: BUZZ OFF!

      System.out.println(loudGreeting.length());
      System.out.println(flyGreeting.replace(‘B’, ‘f’));

      Output: 9
      (includes space)
      Output: fuzz off!
Counting From 0 (part 1)

- System.out.println(flyGreeting.substring(0,3));
  - strings start counting from zero
  - buzz

  \[0 \quad 1 \quad 2 \quad 3\]

  - everything in Java indexed from zero
    - leftover from C and other languages

- substring starts at 0 (inclusive) and ends at 3 (exclusive)
  - answer: buz
  - ugh
So what’s the following return?

String statement = “Bart rules Springfield”;
int iPosition = statement.indexOf(‘t’);

Answer: 3

How about .indexOf(‘i’)?
Returns first instance. Spaces are counted. Answer: 14.

BTW, if the character doesn’t exist, returns −1.
More Cool String Methods

- See your text book.

- Better yet, look at the API
  - Application Program Interface
    - A list of all the “commands” in a language
      - classes (like String) and methods (like indexOf)
  - http://java.sun.com/j2se/1.5.0/docs/api/index.html
    - Find “String” in the list of “All Classes” on the left
The Java API For String

<table>
<thead>
<tr>
<th>Method Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>char</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>int</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>String</td>
</tr>
</tbody>
</table>

Java™ 2 Platform Std. Ed. v1.4.2

StreamPrintService
StreamResult
StreamSource
StreamTokenizer
Streamable
StreamableValue
StrictMath
String
StringBuffer
StringBufferInputStream
StringCharacterIterator
StringContent
Quote Failure

What would happen here?
String quote = "And then he said "yo" to me";

Fails! (won’t compile)
- Thinks the second quote is the end of the assignment.
  - expects a ; to come next.
    - String quote = "And then he said ";

There is no left and right quote distinction. Just the ".

Escape Characters

- Special characters preceded by a \
  - Used when you can’t use the character you want.

- e.g., \

  String quote = "And then he said "yo" to me";
  System.out.println(quote);

  - This outputs
    And then he said “yo” to me
More Escape Characters

Common escapes...

- \" double quote
- \' single quote
- \\ backslash (Irony! Can’t just use a \. Thinks it’s an escape.)
- \n new line
- \t tab

Example:

- System.out.println(“I’m done.\tUh, huh.\n Yeah \”done\”.’’);
- output:
  I’m done. Uh, huh
  Yeah “done”.

Note: no escape necessary for the single quote. Can you think of a case where it is necessary? Hint: char singleQuote = ’’; won’t work.
String Input

- We can read strings from keyboard.
  - Can do it with some simple graphics.

- Have to import special code (the graphics).
  - imports are always at beginning of program.
  - import javax.swing.JOptionPane;

Cool graphics code. Now we’re talkin’!
String Input Example

```java
/**
 * This imports a cool graphics class that we will need below.
 */
import javax.swing.JOptionPane;

/**
 * A simple class that reads in your name and then prints it out.
 * We use some nice (and simple!) graphics to make this look
 * good.
 */
public class CoolHello
{
    /**
     * Remember that a "main" method must always be
     * included in a Java application.
     */
    public static void main(String[] args)
    {
        // A JOptionPane is nothing more than a simple
        // graphical user interface. First we read
        // something in with an input "dialog box". Then
        // we use another kind of dialog box to print out
        // the greeting.
        //
        // Don't worry about the "null".
        // The showMessageDialog method requires
        // some input there, so to keep it happy,
        // we give it nothing (the null).
        String name = JOptionPane.showInputDialog("Please
        Enter Your Name");
        JOptionPane.showMessageDialog(null, "Hello "+name);

        // Ensures that graphics exit "nicely".
        // Otherwise have Java processes that don't quit.
        System.exit(0);
    }
}
```
Note the following

1. Comments can also be put in /**…*/
   • Start each line with *
   • **Use only to comment methods and classes**
   • Use // for everything else (not \\)

2. Must import the graphics

3. **ShowInputDialog** puts input into a String
   • Uses cool graphics, complete with a polite request.

4. **ShowMessageDialog** fancy alternative to System.out.println
   • Prints a String.
JOptionPane Appearance

- Compile and run

- It asks for input

- Then it displays the output
Number Input

Not happy just inputting Strings?

1. Just input a String anyway!
2. Now *convert* String to a number.
Number Input Option 1

- An easy way to convert String to int.
  - String sNumber = “71”;
    int iNumber = Integer.parseInt(sNumber);

- An easy way to convert String to double.
  - String sNumber = “34.5”;
    double dNumber = Double.parseDouble(sNumber);
import javax.swing.JOptionPane;

public class ReadNumbers {
    public static void main(String[] args) {
        String sNumber = JOptionPane.showInputDialog( "Please enter an integer number." );
        int iNumber = Integer.parseInt(sNumber);

        sNumber = JOptionPane.showInputDialog( "Please enter a double number." );
        double dNumber = Double.parseDouble();

        JOptionPane.showMessageDialog(null, "The integer number is " + iNumber);
        JOptionPane.showMessageDialog(null, "The double number is " + dNumber);
        System.exit(0);
    }
}
Number Input Appearance

- Enter any integer.
- Gets converted to an int.
Converting “boolean” and “char”

- Handy, but *doesn’t work for boolean or char.*

- If converting a String to a boolean.
  ```java
  String stringBoolean = “false”;
  boolean newBoolean = true;
  if(stringBoolean.equals(“false”))
  {
      newBoolean = false;
  }
  ```

- If converting a String to a char
  ```java
  String s = “z”;
  char c = s.charAt(0); //grabs character at position 0
  ```
Number Input Option 2: Scanner

- Import special code.
  - import java.util.Scanner;

- For String to int
  ```java
  String stringNumber = "123";
  Scanner scanner = new Scanner(stringNumber);
  int iNumber = scanner.nextInt();
  ```

- For String to double
  ```java
  String stringNumber = "17.45";
  Scanner scanner = new Scanner(stringNumber);
  double dNumber = scanner.nextDouble();
  ```

- Warning! Fails if try to give non-integer or non-double.
  - but implicit casting works for int to double
  - i.e., can give nextDouble an int

**Lingo:** nextInt is just a method in the class Scanner. nextDouble is a method in the class Scanner.
import javax.swing.JOptionPane;
import java.util.Scanner;

public class ReadNumbers {
    public static void main(String[] args) {
        String sNumber = JOptionPane.showInputDialog("Please enter an integer number.");
        Scanner myScanner = new Scanner(sNumber);
        int iNumber = myScanner.nextInt();

        sNumber = JOptionPane.showInputDialog("Please enter an integer number.");
        myScanner = new Scanner(sNumber);
        double dNumber = myScanner.nextDouble();

        JOptionPane.showMessageDialog(null, "The integer number is " + iNumber);
        JOptionPane.showMessageDialog(null, "The double number is " + dNumber);
        System.exit(0);
    }
}

Note: We don’t re-declare sNumber and myScanner. We just re-assign!
Converting “boolean” and “char” With Scanner

If converting a String to a boolean.

String stringBoolean = “true”;
Scanner scanner = new Scanner(stringBoolean);
boolean newBoolean = scanner.nextBoolean();

If converting a String to a char

● Aurgh! It’s different.

String s = “z”;
char c = s.charAt(0); //grabs character at position 0
Scanner FYI

- Scanner doesn’t need the JOptionPane. It can read directly for the keyboard.

  Scanner myScanner = new Scanner(System.in);
  String yourName = myScanner.nextLine();

- Reads whatever you type – just hit return when done typing.

- Read about Scanner in your text and API! Has lots of cool features.
  - http://java.sun.com/j2se/1.5.0/docs/api/index.html