2: Basics of Java Programming

CSC 1051 – Algorithms and Data Structures I

Dr. Mary-Angela Papalaskari
Department of Computing Sciences
Villanova University

Course website:
www.csc.villanova.edu/~map/1051/

Some slides in this presentation are adapted from the slides accompanying Java Software Solutions by Lewis & loftus and from Daniel Joyce’s slides for CSC1051.
Last week – Introduction to course

• Computer Science overview

• Hardware & Software

• Ideas that led to the development of computers—birth of the field of Computer Science
Last week – Lab 1:

• Learn about jGrasp - the programming environment that we will be using in this class
  – Compile and run a java program

• Understand the relationship between a Java class name and the name of the .java file where the class is defined

• Practice using basic Java output statements and adding comments

• Learn about variables, string literals, concatenation. *E.g.*,

  ```java
  int x = 42, count = 100;
  String name = "Kripke";
  System.out.println ("Howdy " + name);
  System.out.println ("The answer is " + x);
  System.out.println ("And counting up: " + (count + 1));
  ```

• Explore Java syntax
• Experience some errors!

CSC 1051 M.A. Papalaskari, Villanova University
Next:

- Printing statements
- Variables
- Assignment operator
- Primitive data types
- Arithmetic operators
- Problem solving: Create a program that will help us calculate a grade point average (GPA) given the number of quality points (QP) and the number of credits.
Basics of Java programming

Topic thread

• 2.1 Character Strings
• 2.2 Variables, Assignment
• 2.3 Data Types, in particular int, double
• 2.4 Expressions (simple)
• 2.5 Data Conversion
• 2.6 Interactive Programs
• 5.1 Boolean Expressions
• 5.2 The if Statement
• 5.4 The while Statement
Character Strings

- A *string literal* is represented by putting double quotes around the text

- Examples:
  "This is a string literal."
  "123 Main Street"
  "X"

- Every character string is an object in Java, defined by the `String` class
The println Method

• In the *Lincoln* program we invoked the `println` method to print a character string

• The `System.out` object represents a destination (the monitor screen) to which we can send output

```
System.out.println ("Whatever you are, be a good one.");
```
The print Method

• The `System.out` object has another method:

• `print` is similar to the `println` except that it does not advance to the next line

• Example: `Countdown.java`
public class Countdown {
    // Prints two lines of output representing a rocket countdown.
    public static void main (String[] args) {
        System.out.print("Three... ");
        System.out.print("Two... ");
        System.out.print("One... ");
        System.out.print("Zero... ");
        System.out.println("Liftoff!"); // appears on first output line
        System.out.println("Houston, we have a problem.");
    }
}
Output

Three... Two... One... Zero... Liftoff!
Houston, we have a problem.

public class Countdown
{
    // Prints two lines of output representing a rocket countdown.
    public static void main (String[] args)
    {
        System.out.print ("Three... ");
        System.out.print ("Two... ");
        System.out.print ("One... ");
        System.out.print ("Zero... ");
        System.out.println ("Liftoff!"); // appears on first output line
        System.out.println ("Houston, we have a problem.");
    }
}
String Concatenation

• The *string concatenation operator* (+) is used to append one string to the end of another
  
  "And one more " + "thing"

• It can also be used to append a number to a string

• A string literal cannot be broken across two lines in a program

• See [Facts.java](#)
public class Facts{
    public static void main (String[] args)
    {
        System.out.println("We present the following facts for your "+"extracurricular edification: ");
        System.out.println ( );
        System.out.println ( );
        System.out.println ("Letters in the Hawaiian alphabet: 12");
    }
}
// A numeric value can be concatenated to a string
System.out.println ("Dialing code for Antarctica: " + 672);

System.out.println ("Year in which Leonardo da Vinci invented " + "the parachute: " + 1515);

System.out.println ("Speed of ketchup: " + 40 + " km per year");
} 

We present the following facts for your extracurricular edification:

Letters in the Hawaiian alphabet: 12
Dialing code for Antarctica: 672
Year in which Leonardo da Vinci invented the parachute: 1515
Speed of ketchup: 40 km per year

```java
System.out.println("Speed of ketchup: " + 40 + " km per year");
```

Escape Sequences

• What if we wanted to print the quote character?
• Let’s try something like this…

    System.out.println ("I said "Hello" to you.");

• An escape sequence is a series of characters that represents a special character
• An escape sequence begins with a backslash character (\)

    System.out.println ("I said \"Hello\" to you.");
Escape Sequences

• Some Java escape sequences:

<table>
<thead>
<tr>
<th>Escape Sequence</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\b</td>
<td>backspace</td>
</tr>
<tr>
<td>\t</td>
<td>tab</td>
</tr>
<tr>
<td>\n</td>
<td>newline</td>
</tr>
<tr>
<td>\r</td>
<td>carriage return</td>
</tr>
<tr>
<td>&quot;</td>
<td>double quote</td>
</tr>
<tr>
<td>'</td>
<td>single quote</td>
</tr>
<tr>
<td>\</td>
<td>backslash</td>
</tr>
</tbody>
</table>

• See [Roses.java](#)
public class Roses {
    // Prints a poem (of sorts) on multiple lines.
    public static void main (String[] args) {
        System.out.println("Roses are red,\nViolets are blue,\n" +
                        "Sugar is sweet,\nBut I have " +
                        "commitment issues",\n" +
                        "So I'd rather just be friends\nAt this point in our " +
                        "relationship.");
    }
}
public class Roses {
    public static void main(String[] args) {
        System.out.println("Roses are red,
	Violets are blue,
Sugar is sweet,
But I have "commitment issues",
So I'd rather just be friends
At this point in our relationship.");
    }
}
Quick Check

Write a single `println` statement that produces the following output:

"Thank you all for coming to my home tonight," he said mysteriously.
Example: Problem Solving

- Problem from textbook:

  PP 2.3 (page 109) Write an application that prompts for and reads a person's name, age, college and pet's name. Then print a paragraph inserting the appropriate data. For example:

  Hello, my name is name and I am age years old. I'm enjoying my time at college, though I miss my pet petname very much!

- Variables?
- Algorithm?
Topic Thread

- 2.1 Character Strings
- 2.2 Variables, Assignment
- 2.3 Data Types, in particular int, double
- 2.4 Expressions (simple)
- 2.5 Data Conversion
- 2.6 Interactive Programs
- 5.1 Boolean Expressions
- 5.2 The if Statement
- 5.4 The while Statement
Variables

- A variable is a name for a location in memory
- A variable must be declared by specifying the variable's name and the type of information that it will hold

```java
int age;
String name, petname;
```

Multiple variables can be created in one declaration
Assignment Statement

- *Changes the value of a variable*
- The assignment operator is the `=` sign

```
total = 55 - discount;
```

- The expression on the right is evaluated and the result is stored in the variable on the left
- The old value that was in `total` is overwritten
- See [Geometry.java](#) (page 68)
Variable Initialization

- A variable can be given an initial value in the declaration

```java
int sum = 0;
String name = “Kripke”;```

- When a variable is referenced in a program, its **current** value is used.
Example: Problem Solving

- **Problem from textbook:**

PP 2.3 (page 109) Write an application that prompts for and reads a person’s name, age, college and pet’s name. Then print a paragraph inserting the appropriate data. For example:

Hello, my name is *name* and I am *age* years old. I’m enjoying my time at *college*, though I miss my pet *petname* very much!

- **Variables?**

- **Algorithm?**
Topic Thread

- 2.1 Character Strings
- 2.2 Variables, Assignment
- 2.3 Data Types, in particular int, double
- 2.4 Expressions (simple)
- 2.5 Data Conversion
- 2.6 Interactive Programs
- 5.1 Boolean Expressions
- 5.2 The if Statement
- 5.4 The while Statement
Primitive Data

• There are eight primitive data types
• Four of them represent *integers*:
  – byte, short, int, long
• Two of them represent *floating point* numbers:
  – float, double
• One of them represents *characters*:
  – char
• And one of them represents *boolean* values:
  – boolean
## Numeric Primitive Data

<table>
<thead>
<tr>
<th>Type</th>
<th>Storage</th>
<th>Min Value</th>
<th>Max Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>8 bits</td>
<td>-128</td>
<td>127</td>
</tr>
<tr>
<td>short</td>
<td>16 bits</td>
<td>-32,768</td>
<td>32,767</td>
</tr>
<tr>
<td>int</td>
<td>32 bits</td>
<td>-2,147,483,648</td>
<td>2,147,483,647</td>
</tr>
<tr>
<td>long</td>
<td>64 bits</td>
<td>&lt; -9 $\times 10^{18}$</td>
<td>&gt; 9 $\times 10^{18}$</td>
</tr>
<tr>
<td>float</td>
<td>32 bits</td>
<td>+/- 3.4 $\times 10^{38}$ with 7 significant digits</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>+/- 1.7 $\times 10^{308}$ with 15 significant digits</td>
<td></td>
</tr>
</tbody>
</table>
Characters

- A char variable stores a single character
- Character literals are delimited by single quotes:
  
  'a'  'X'  '7'  '$'  ','  ' '  '

- Example declarations:

  char topGrade = 'A';
  char terminator = ';', separator = ' ';

- Note the difference between a primitive character variable, which holds only one character, and a String object, which can hold multiple characters
Character Sets

• A character set is an ordered list of characters, with each character corresponding to a unique number

• A char variable in Java can store any character from the Unicode character set

• The Unicode character set uses sixteen bits per character, allowing for 65,536 unique characters

• It is an international character set, containing symbols and characters from many world languages
Characters

• The ASCII character set is older and smaller than Unicode, but is still quite popular

• The ASCII characters are a subset of the Unicode character set, including:

  uppercase letters   A, B, C, ...
  lowercase letters   a, b, c, ...
  punctuation         period, semi-colon, ...
  digits              0, 1, 2, ...
  special symbols     & , | , \ , ...
  control characters  carriage return, tab, ...
Boolean

• A boolean value represents a true or false condition

• The reserved words true and false are the only valid values for a boolean type

    boolean done = false;

• A boolean variable can also be used to represent any two states, such as a light bulb being on or off
Strings are **not** primitive data

- We can still declare and use String variables:

  ```java
  String message = "Roses are red";
  ```

  ... And use assignment to change their values:

  ```java
  message = message + "\nViolets are blue";
  ```
Topic Thread

• 2.1 Character Strings
• 2.2 Variables, Assignment
• 2.3 Data Types, in particular int, double
• 2.4 Expressions (simple)
• 2.5 Data Conversion
• 2.6 Interactive Programs
• 5.1 Boolean Expressions
• 5.2 The if Statement
• 5.4 The while Statement
Expressions

• An *expression* is a combination of one or more operators and operands

• *Arithmetic expressions* compute numeric results and make use of the arithmetic operators:

  Addition +
  Subtraction −
  Multiplication *
  Division /
  Remainder %

• If either or both operands used by an arithmetic operator are floating point, then the result is a floating point
Operator Precedence

- Operators can be combined into complex expressions
  
  \[ \text{result} = \text{total} + \text{count} / \text{max} - \text{offset}; \]

- Operators have a well-defined precedence which determines the order in which they are evaluated

- Multiplication, division, and remainder are evaluated prior to addition, subtraction, and string concatenation

- Arithmetic operators with the same precedence are evaluated from left to right, but parentheses can be used to force the evaluation order
Operator Precedence

• What is the order of evaluation in the following expressions?

\[
\begin{align*}
& a + b + c + d + e \quad \quad \quad \quad \quad a - b / c + d * e \\
& a / (b + c) - d \% e \\
& a / (b * (c + (d - e)))
\end{align*}
\]
Assignment Revisited

- The assignment operator has a lower precedence than the arithmetic operators.

\[
\text{First the expression on the right hand side of the } = \text{ operator is evaluated}
\]

\[
\text{answer } = \text{ sum } / 4 + \text{ MAX } * \text{ lowest;}
\]

- Then the result is stored in the variable on the left hand side.

CSC 1051 M.A. Papalaskari, Villanova University
Increment and Decrement

• The increment and decrement operators use only one operand

• The *increment operator* (++) adds one to its operand

• The *decrement operator* (--) subtracts one from its operand

• The statement

  \[ \text{count}++; \]

  is functionally equivalent to

  \[ \text{count} = \text{count} + 1; \]
Increment and Decrement

- The increment and decrement operators can be applied in *postfix form*:
  
  \[ \text{count}++ \]

- or *prefix form*:
  
  \[ ++\text{count} \]

- When used as part of a larger expression, the two forms can have different effects.

- Because of their subtleties, the increment and decrement operators should be used with care.
Division and Remainder

- If both operands to the division operator (/) are integers, the result is an integer (the fractional part is discarded)

  \[
  14 \div 3 \quad \text{equals} \quad 4 \\
  8 \div 12 \quad \text{equals} \quad 0
  \]

- The remainder operator (%) returns the remainder after dividing the second operand into the first

  \[
  14 \% 3 \quad \text{equals} \quad 2 \\
  8 \% 12 \quad \text{equals} \quad 8
  \]
Assignment Reminder

- The right and left hand sides of an assignment statement can contain the same variable.

  First, one is added to the original value of count.

  \[
  \text{count} = \text{count} + 1;
  \]

  Then the result is stored back into count (overwriting the original value).
Assignment operator

- Assignment ( = ) copies the value of the right side into the memory location associated with the left side.
- It does not (for primitive types) set up an ongoing equivalence.

```java
int davesAge = 21;
int suesAge;
suesAge = davesAge;
davesAge = 22;
System.out.println (davesAge); // prints 22
System.out.println (suesAge);  // prints 21
```

Tracing program code is an important skill!!
Topic Thread

• 2.1 Character Strings
• 2.2 Variables, Assignment
• 2.3 Data Types
• 2.4 Expressions (simple)
• 2.5 Data Conversion
• 2.6 Interactive Programs
• 5.1 Boolean Expressions
• 5.2 The if Statement
• 5.4 The while Statement
Interactive Programs

- The `Scanner` class has methods for reading input.
- We declare a `Scanner` object to read input from the keyboard:

  ```java
  Scanner scan = new Scanner (System.in);
  ```
Reading Input

• Once created, the Scanner object can be used to invoke various input methods, such as:

```
answer = scan.nextLine();
```

• The nextLine method reads all of the input until the end of the line is found
Reading Input

- The `Scanner` class is part of the `java.util` class library, and must be imported into a program to be used.
- The import statement goes at beginning of your program (above class definition)

```java
import java.util.Scanner;
```

(See `Echo.java`)

- The details of object creation and class libraries are discussed further in Chapter 3.
import java.util.Scanner;

public class Echo
{
    //-----------------------------------------------
    // Reads a character string from the user and prints it.
    //-----------------------------------------------
    public static void main (String[] args)
    {
        String message;
        Scanner scan = new Scanner (System.in);

        System.out.println ("Enter a line of text:");

        message = scan.nextLine();

        System.out.println ("You entered: \\
" + message + \\
        \\
        \\
        ");
    }
}
import java.util.Scanner;

public class Echo {
    //----------------------------------------------------------------------------
    //  Reads a character string from the user and prints it.
    //----------------------------------------------------------------------------
    public static void main (String[] args) {
        String message;
        Scanner scan = new Scanner (System.in);

        System.out.println ("Enter a line of text:");

        message = scan.nextLine();

        System.out.println ("You entered: \\
                           " + message + \\
                           \\
                           ");
    }
}
Reading in numbers

• `nextInt` reads in an integer:

• Example:

```java
age = scan.nextInt();
```

• `nextDouble` similar method for type `double`

• *White space (space, tab, new line) can be used to separate input tokens*

• `next` reads the next input token and returns it as a string

• See [GasMileage.java](GasMileage.java)
import java.util.Scanner;

public class GasMileage
{
  // Calculates fuel efficiency based on values entered by the user.
  public static void main (String[] args)
  {
    int miles;
    double gallons, mpg;

    Scanner scan = new Scanner (System.in);
    continue
System.out.print("Enter the number of miles: ");
miles = scan.nextInt();

System.out.print("Enter the gallons of fuel used: ");
gallons = scan.nextDouble();

mpg = miles / gallons;

System.out.println("Miles Per Gallon: " + mpg);
System.out.print("Enter the number of miles: ");
miles = scan.nextInt();

System.out.print("Enter the gallons of fuel used: ");
gallons = scan.nextDouble();

mpg = miles / gallons;

System.out.println("Miles Per Gallon: "+ mpg);
}
Homework

• Read Sections 2.1 – 2.4 and 2.6

  – **Always** do all self-review exercises when you review material

• Do Exercises EX 2.2 – 2.8 and 2.11