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 documentation (comments) to your source code.

• Learn about variables, string literals, concatenation. *E.g.*,
  – `int x = 42, count = 100;`
  – "Many wise words"
  – "Many " + " many wise words"
  – "Many \"wise\" words"

• Explore Java syntax
• Experience some errors!
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`
/**
 * Demonstrates the difference between print and println.
 */

public class 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.");
    }
}
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("Letters in the Hawaiian alphabet: 12");
    }
}
continue

    // 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");
}
Output

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

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](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,\n        Violets are blue,\n        Sugar is sweet,\n        But I have "commitment issues",\n        So I'd rather just be friends\n        At this point in our relationship.");
    }
}
public class Roses {
  // Prints a poem (of sorts) on multiple lines.
  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.
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
Example: 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.

• The appropriate formula is

\[
GPA = \frac{QP}{\text{credits}}
\]
For Example

<table>
<thead>
<tr>
<th>Course</th>
<th>Credits</th>
<th>Grade</th>
<th>QPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Underwater Basket Weaving</td>
<td>3</td>
<td>A = 4</td>
<td>12</td>
</tr>
<tr>
<td>Main Line Boutiques</td>
<td>3</td>
<td>B = 3</td>
<td>9</td>
</tr>
<tr>
<td>Winning the Hoops Lottery</td>
<td>3</td>
<td>C = 2</td>
<td>6</td>
</tr>
<tr>
<td>Web Surfing</td>
<td>3</td>
<td>B = 3</td>
<td>9</td>
</tr>
<tr>
<td>Alg and Data Structures</td>
<td>4</td>
<td>A = 4</td>
<td>16</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td><strong>16</strong></td>
<td></td>
<td><strong>52</strong></td>
</tr>
</tbody>
</table>

\[
GPA = \frac{52}{16} = 3.25
\]
The GPA Problem

Solution 1

- Not very exciting, is it?
- Let’s add some variables

```java
// GPA01.java

// Prints out a QPA

public class GPA01 {
    public static void main (String[] args) {
        System.out.println ("Quality Points: 52");
        System.out.println ("Credits: 16");
        System.out.println ("\n\tGPA: 3.25");
    }
}
```
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.

```
data type    variable name
  int credits;
  int count, temp, result;
```

Multiple variables can be created in one declaration.
Variable Initialization

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

```java
int sum = 0;
int base = 32, max = 149;
```

• When a variable is referenced in a program, its **current** value is used.
Solution 2

Still not very exciting, is it?
Let’s add some processing
Assignment Statement

- Changes the value of a variable
- The assignment operator is the $=\ $ sign

$\text{total} = 55 - \text{discount};$

- The expression on the right is evaluated and the result is stored in the variable on the left
- The old value that was in $\text{total}$ is overwritten
- See Geometry.java (page 68)
Assignment Revisited

• 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, 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 x 10^{18}</td>
<td>&gt; 9 x 10^{18}</td>
</tr>
<tr>
<td>float</td>
<td>32 bits</td>
<td>+/- 3.4 x 10^{38} with 7 significant digits</td>
<td></td>
</tr>
<tr>
<td>double</td>
<td>64 bits</td>
<td>+/- 1.7 x 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:

  ```java
  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:
  
<table>
<thead>
<tr>
<th>Category</th>
<th>Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>uppercase letters</td>
<td>A, B, C, …</td>
</tr>
<tr>
<td>lowercase letters</td>
<td>a, b, c, …</td>
</tr>
<tr>
<td>punctuation</td>
<td>period, semi-colon, …</td>
</tr>
<tr>
<td>digits</td>
<td>0, 1, 2, …</td>
</tr>
<tr>
<td>special symbols</td>
<td>&amp;,</td>
</tr>
<tr>
<td>control characters</td>
<td>carriage return, tab, …</td>
</tr>
</tbody>
</table>
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:

        String message = "Roses are red";

... And use assignment to change their values:

        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?
  
  \[ a + b + c + d + e \]
  \[ a - b / c + d * e \]
  \[ a / (b + c) - d \% e \]
  \[ a / (b * (c + (d - e))) \]
Assignment Revisited

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

First the expression on the right hand side of the `=` operator is evaluated:

```
answer  =  sum / 4 + MAX * lowest;
```

Then the result is stored in the variable on the left hand side.
Increment and Decrement

- The increment and decrement operators use only one operand.
- The \textit{increment operator} \((++\)} adds one to its operand.
- The \textit{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

\[
14 \mod 3 \quad \text{equals} \quad 2
\]

\[
8 \mod 12 \quad \text{equals} \quad 8
\]
Solution 3

- A little more interesting but ...
- What happened to the output?

```java
public class GPA03
{
    public static void main (String[ ] args)
    {
        int qp = 52;
        int credits = 16;
        double gpa = qp / credits;

        System.out.println ("Quality Points: " + qp);
        System.out.println ("Credits: " + credits);
        System.out.println ();
        System.out.println ("\tGPA: " + gpa);
    }
}
```
int qp = 52;
int credits = 16;
double gpa = qp / credits;

System.out.println("Quality Points: ");
System.out.println("Credits: ");

Both variables are of type int so the result is an int
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
Assignment Conversion

- *Assignment conversion* occurs when a value of one type is assigned to a variable of another type.

- Example:

  ```java
  int dollars = 20;
  double money = dollars;
  ```

- Only *widening conversions* can happen via assignment.
Converting from one type to another

- **Widening conversions**
  - small data type → larger one (eg int to an double)

- **Narrowing conversions**
  - large data type → smaller one (eg double to a int)
  - *can lose information!*
Converting from one type to another

- **Widening conversions**
  - small data type $\rightarrow$ larger one (eg `int` to an `double`)

- **Narrowing conversions**
  - large data type $\rightarrow$ smaller one (eg `double` to a `int`)
  - can lose information!

- In Java, data conversions can occur in three ways:
  - assignment conversion
  - promotion
  - casting

```java
double money;
money = 3;
money = 5 / 2.0;
money = (double) 8;
```
Assignment Conversion

- *Assignment conversion* occurs when a value of one type is assigned to a variable of another.

- Example:

  ```java
  int dollars = 20;
  double money = dollars;
  ```

- Only widening conversions can happen via assignment.
Promotion

- *Promotion* happens automatically when operators in expressions convert their operands

- Example:

  ```java
  int count = 12;
  double sum = 490.27;
  result = sum / count;
  ```

- The value of `count` is converted to a floating point value to perform the division calculation
Casting

- *Casting* forces a change of type, even if information is lost
  - Both widening and narrowing conversions

- To cast, the type is put in parentheses in front of the value being converted:

  ```java
  int total = 50;
  float result = (float) total / 6;
  ```

- Without the cast, the fractional part of the answer would be lost
# Data Conversion

## Widening Conversions

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>short, int, long, float, or double</td>
</tr>
<tr>
<td>short</td>
<td>int, long, float, or double</td>
</tr>
<tr>
<td>char</td>
<td>int, long, float, or double</td>
</tr>
<tr>
<td>int</td>
<td>long, float, or double</td>
</tr>
<tr>
<td>long</td>
<td>float or double</td>
</tr>
<tr>
<td>float</td>
<td>double</td>
</tr>
</tbody>
</table>

## Narrowing Conversions

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>byte</td>
<td>char</td>
</tr>
<tr>
<td>short</td>
<td>byte or char</td>
</tr>
<tr>
<td>char</td>
<td>byte or short</td>
</tr>
<tr>
<td>int</td>
<td>byte, short, or char</td>
</tr>
<tr>
<td>long</td>
<td>byte, short, char, or int</td>
</tr>
<tr>
<td>float</td>
<td>byte, short, char, int, or long</td>
</tr>
<tr>
<td>double</td>
<td>byte, short, char, int, long, or float</td>
</tr>
</tbody>
</table>
public class GPA04 {
    public static void main (String[] args) {
        int qp = 52;
        int credits = 16;
        double gpa = (double) qp / credits;

        System.out.println("Quality Points: " + qp);
        System.out.println("Credits: " + credits);
        System.out.println();
        System.out.println("\tGPA: " + gpa);
    }
}
int qp = 52;

int credits = 16;

double gpa = (double) qp / credits;

System.out.println("Quality Points: "+qp);
System.out.println("Credits: "+credits);

Then promotion occurs to make both expressions the same type - so, value from credits is changed to double.

First casting occurs, value from qp changed to double.
int qp = 52;
int credits = 16;
double gpa = qp / credits;

Solution 3

Previous version

Promotion occurs after the division --too late!

No casting
int division
public class GPA04 {
  
  public static void main (String[] args) {
    int qp = 52;
    int credits = 16;
    double gpa = (double) qp / credits;
    
    System.out.println ("Quality Points: " + qp);
    System.out.println ("Credits: " + credits);
    System.out.println ("
    GPA: " + gpa);
  }
}
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:

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

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

   \[ \text{answer} = \text{scan.nextLine}(); \]

• The \texttt{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 {

    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:

  ```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](#)
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);
```java
Sample Run

Enter the number of miles: 328
Enter the gallons of fuel used: 11.2
Miles Per Gallon: 29.28571428571429
```
import java.util.Scanner;
public class GPA05 {
    public static void main (String[] args) {
        int qp;
        int credits;
        double gpa;
        Scanner scan = new Scanner(System.in);
        // get input
        System.out.print("Enter Quality Points > ");
        qp = scan.nextInt();
        System.out.print("Enter Credits > ");
        credits = scan.nextInt();
        // calculate gpa
        gpa = (double) qp / credits;
        // output information
        System.out.println("\nQuality Points: "+qp);
        System.out.println("Credits: "+credits);
        System.out.println("\n\tGPA: "+gpa);
    }
}
Homework

• Read Sections 2.1 - 2.6

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

• Do Exercises EX 2.2 - 2.11
Some slides added to this presentation for discussion of algorithms
How many of each can you pack in the black box?
How many of each can you pack in the black box?
How many of each can you pack in the black box?