Lecture 4:
Interactive programs

CSC 1051 – Data Structures and Algorithms I

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

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

- 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.
Today:

• More on:
  – data types
  – assignment operators

• Interactive programs
  – Input using the Scanner class
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
The GPA Problem

Solution 1

- Not very exciting, is it?
- Let’s add some storage (remember our model of computing)
Solution 3

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 ("	GPA: " + gpa);
    }
}

Quick Check

What are the results of the following expressions?

12 / 2
12.0 / 2.0
10 / 4
10 / 4.0
4 / 10
4.0 / 10
12 % 3
10 % 3
3 % 10
Quick Check

What are the results of the following expressions?

\[
\begin{align*}
12 \div 2 &= 6 \\
12.0 \div 2.0 &= 6.0 \\
10 \div 4 &= 2 \\
10 \div 4.0 &= 2.5 \\
4 \div 10 &= 0 \\
4.0 \div 10 &= 0.4 \\
12 \% 3 &= 0 \\
10 \% 3 &= 1 \\
3 \% 10 &= 3
\end{align*}
\]
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);
    }
}

Solution 3

- A little more interesting but ...
- What happened to the output?
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
The difference between the numeric primitive types is their size and the values they can store:

<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:

  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.
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
Data Conversion

• For example, in a particular situation we may want to treat an integer as a floating point value.

• These conversions do not change the type of a variable or the value that's stored in it – they only convert a value as part of a computation.
Converting from one type to another

• **Widening conversions**
  – small data type → larger one (eg `short` to an `int`)

• **Narrowing conversions**
  – large data type → smaller one (eg `int` to a `short`)
  – *can lose information!*

• In Java, data conversions can occur in three ways:
  – assignment conversion
  – promotion
  – casting
## 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>
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:

  ```
  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
Solution 4

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);
    }
}

First casting occurs, copy of value in qp changed to double
Then promotion occurs to credits, i.e. copy of value in credits is changed to double

- That’s better ...
- Let’s add input
Outline

Character Strings
Variables and Assignment
Primitive Data Types
Expressions
Data Conversion
Interactive Programs
Graphics
Applets
Drawing Shapes
Interactive Programs

- The `Scanner` class has methods for reading input.
- We set up a “`Scanner` object” to read input from various sources, including the user typing values on the keyboard.

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

- Keyboard input is represented by the `System.in` object.
Reading Input

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

```java
answer = scan.nextLine();
```
Reading Input

- The `Scanner` class is part of the `java.util` class library, and must be imported into a program to be used.

- The `nextLine` method reads all of the input until the end of the line is found.

- 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 + \\"");
    }
}
Sample Run

Enter a line of text:
You want fries with that?
You entered: "You want fries with that?"

```java
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` and 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 input tokens*

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

- See [GasMileage.java](https://example.com/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);
Solution 5

- Note: Opening comment removed to allow better viewing.
- Now we are cooking.
- But, what if you haven’t taken any classes?
- Next we will add some control, i.e. some decision making capability.

```java
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 ("Quality Points: " + qp);
        System.out.println ("Credits: " + credits);
        System.out.println ("GPA: " + gpa);
    }
}
```
Homework

• Read Sections 2.5, 2.6
  – *Always* do all self-review exercises when you review material
• Do Exercises EX 2.10 and as many of PP 2.2-2.8 as you can

• Next class we will have a quiz and continue working on interactive programs (section 2.6)
• **Optional:** email me a 10 min quiz for next Wednesday