COMP 110
Introduction to Programming

Fall 2015
Time: TR 9:30 – 10:45
Room: AR 121 (Hanes Art Center)

Jay Aikat
FB 314, aikat@cs.unc.edu

Previous Class

• What did we discuss?
Today

- Announcements
  - Lab1: **due Wed, Sep 9 at 11:55 PM**
  - Midterm is on Thu, Oct 8

- Scanner
- String
- If-else

---

**Some Scanner Class Methods**

- `Scanner_Object_Name.next()`
  Returns the `String` value consisting of the next keyboard characters up to, but not including, the first delimiter character. The default delimiters are whitespace characters.

- `Scanner_Object_Name.nextLine()`
  Reads the rest of the current keyboard input line and returns the characters read as a value of type `String`. Note that the line terminator `\n` is read and discarded; it is not included in the string returned.

- `Scanner_Object_Name.nextInt()`
  Returns the next keyboard input as a value of type `int`.

- `Scanner_Object_Name.nextDouble()`
  Returns the next keyboard input as a value of type `double`.

- `Scanner_Object_Name.nextFloat()`
  Returns the next keyboard input as a value of type `float`. 
Some **Scanner** Class Methods

- `Scanner_Object_Name.nextLong()`  
  Returns the next keyboard input as a value of type `long`.

- `Scanner_Object_Name.nextByte()`  
  Returns the next keyboard input as a value of type `byte`.

- `Scanner_Object_Name.nextShort()`  
  Returns the next keyboard input as a value of type `short`.

- `Scanner_Object_Name.nextBoolean()`  
  Returns the next keyboard input as a value of type `boolean`. The values of `true` and `false` are entered as the words `true` and `false`. Any combination of uppercase and lowercase letters is allowed in spelling `true` and `false`.

- `Scanner_Object_Name.useDelimiter(Delimiter_Word)`;  
  Makes the string `Delimiter_Word` the only delimiter used to separate input. Only the exact word will be a delimiter. In particular, blanks, line breaks, and other whitespace will no longer be delimiters unless they are a part of `Delimiter_Word`.
  
  This is a simple case of the use of the `useDelimiter` method. There are many ways to set the delimiters to various combinations of characters and words, but we will not go into them in this book.

**nextLine()** Method Caution

- The `nextLine()` method reads
  - The remainder of the current line,
  - Even if it is empty.
**nextLine() Method Caution**

- Example – given following declaration.

```java
int n;
String s1, s2;
n = keyboard.nextInt();
s1 = keyboard.nextLine();
s2 = keyboard.nextLine();
```

- Assume input shown

```
42
and don't you forget it.
```

n is set to 42

but s1 is set to the empty string.

---

**The class String**

- String
  - sequence of zero or more characters
  - enclosed in double quotation marks
  - null or empty strings have no characters
  - numeric strings consist of integers or decimal numbers
  - length is the number of characters in a string

- The class String is used to manipulate strings

- Examples:
  - "Hello World"
  - "1234"
  - "45.67"
  - ""

---
Strings

- Every character has a position in the string (starting with 0)
  "Hello World"
  0123456789...

- The length of the string is the number of characters in it
  - what's the length of "Hello World"?
    11
    (count the space)

Parsing Numeric Strings

- In Java, input from the user comes in the form of a string
  - we need to know how to get the numeric values out of the string

- Numeric String
  - a string with only integers or decimal numbers
  - "6723", "-823", "345.76"
String Concatenation

- A string cannot be split between two lines

\[
\text{String greeting = "How are you doing today";}
\]

- Concatenation (+) - produces one string where the second string has been appended to the first

\[
\text{String greeting = "How are you doing" + " today";}
\]

  greeting \textbf{How are you doing today?}

- Operator + can be used to concatenate two strings or a string and a numeric value or character

- Precedence rules still apply

- Example:

  String \texttt{str};
  int \texttt{num1 = 12, num2 = 26};
  \texttt{str = "The sum = " + num1 + num2;}

  \texttt{str \textbf{The sum = 1226}}
Questions

What is the result of the following String concatenations?
String str1 = "Hello";
String str2 = "World";
String str3 = str1 + " " + str2;

```
str3 Hello World
```

int num1 = 10;
int num2 = 2;
String str = "The difference is " + (num1 - num2);

```
str The difference is 8
```

Flow of Control

- *Flow of control* is the order in which a program performs actions.
  - Up to this point, the order has been sequential.
- A *branching statement* chooses between two or more possible actions.
- A *loop statement* repeats an action until a stopping condition occurs.
Flow Chart

Get up

Check Time

Before 7am?

YES

NO

Eat Breakfast

Pack Breakfast

Take Bus to School

Student.getUp();

if (time < 7) {
    Student.eatBreakfast();
}
else { // time >= 7
    Student.packBreakfast();
}
Student.takeBus();

Java Example

import java.util.*;

public class FlowChart {
    public static void main(String[] args) {
        System.out.println("Give me an integer:");
        Scanner keyboard = new Scanner(System.in);
        int inputInt = keyboard.nextInt();
        if (inputInt > 50)
            System.out.println("Big number");
        else
            System.out.println("Small number");
    }
}

What if your input is exactly 50?
If and Else

- Take two minutes and write down an example pseudocode of one If and Else Statement

- For example:

```java
if (avgQuizGrade >= 90) {
    Topic.startNext();
} else {
    Topic.redoLast();
}
```

Java Comparison Operators

### FIGURE 3.4 Java comparison operators

<table>
<thead>
<tr>
<th>Math Notation</th>
<th>Name</th>
<th>Java Notation</th>
<th>Java Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Equal to</td>
<td>==</td>
<td>balance == 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>answer == 'y'</td>
</tr>
<tr>
<td>!=</td>
<td>Not equal to</td>
<td>!=</td>
<td>income != tax</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>answer != 'y'</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater than</td>
<td>&gt;</td>
<td>expenses &gt; income</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
<td>&gt;=</td>
<td>points &gt;= 60</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less than</td>
<td>&lt;</td>
<td>pressure &lt; max</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less than or equal to</td>
<td>&lt;=</td>
<td>expenses &lt;= income</td>
</tr>
</tbody>
</table>
Expressions

- Expression?
  - An expression can be a variable, a value, or a combination made up of variables, values and operators
  - An expression has a value
  - Arithmetic expression: a combination of numbers with a number value
    - 10, taxRate/100, (cost + tax) * discount
  - String expression: a combination of Strings with a String value
    - “Hello”, “The total cost is ” + totalCost

Boolean Expressions

- A combination of values and variables by comparison operators. Its value can only be true or false
- Example expressions
  - 5 == 3; // false
  - variable <= 6; // depends on the value of variable
    - What if variable is 5? What if variable is 6?
  - myInt != temp; // depends on both values
    - What if myInt is 0 and temp is 2?
Syntax for IF statement

• Syntax rule for if statement:
  – \texttt{if (boolean expression)}
    \{ statements; \}

&&: and

• What if you need multiple expressions to be true?
• Syntax rule:
  – \texttt{(expression) \&\& (expression) \&\& ...}
    • Expressions go in ( )
  – (Time < 7) \&\& (I’ve prepared breakfast)
• Will only be true if \texttt{ALL} statements are true
| | : or

• What if you need ONE expression to be true out of many expressions
• Syntax rule:
  – \((expression) \text{ || } (expression) \text{ || } ...\)
  • Again, expressions go in ( )
  – (I’ve had breakfast) || (Time > 7)
• Will be true if ONE expression is true

!: not

• Syntax rule:
  – !\((expression)\)
  • Again, expressions go in ( )
  – !(I’ve had breakfast)
• Will be true if the expression is false
• ! is not recommended
  – You will get confused. Try to write expressions straightforward
  • Use (cost != 3) instead of !(cost == 3)
  • Use (time <= 7) instead of !(time > 7)
Logical Operators

Comparison vs. Logical Operators

- Comparison operators connect values or variables
  - After connection, it’s a boolean expression
    - $a > b$
    - $c == d$
- Logical operators connect boolean expressions
  - $(a > b) \&\& (c == d)$
More Complex Boolean Expressions

• Combination of && and ||
  – ((3 < 7) || (2 == 5)) && ((4 != 2) && (1 <= 1))
  – ((true) || (false)) && (true) && (true)
  – (true) && (true)
  – true

• if ((I’m at Subway) && (You’re at Subway)) ||
  ((I’m at Starbucks) && (You’re at Starbucks))
  {
    I will see you;
  }

Boolean Variable

• A boolean variable saves a boolean value

  boolean systemsAreOK =
  ((temperature <= 100) && (thrust >= 12000) && (cabinPressure > 30));
  // You can use “=” to assign a boolean value to a boolean variable

  if (systemsAreOK){
    // It’s the same as if (systemsAreOK == true)
    System.out.println("Initiate launch sequence.");
  }
  else{
    System.out.println("Abort launch sequence.");
  }
Assignment vs. Equal To

• int n1=1;
• Int n2=2;
• if ( n1 = n2 ) {...}  
  – Error!!!! It’s an assignment statement!
• if ( n1 == n2 ) {...}  
  – Correct. It’s a boolean expression now.

Some Terminology follows...
Class Loader

- A Java program typically consists of several pieces called classes.
- Each class may have a separate author and each is compiled (translated into byte-code) separately.
- A class loader (called a linker in other programming languages) automatically connects the classes together.

Programmer, User, Package...

- The person who writes a program is called the programmer.
- The person who interacts with the program is called the user.
- A package is a library of classes that have been defined already.
  
  \[ \text{import java.util.Scanner;} \]
Arguments, Variables...

- The item(s) inside parentheses are called *argument(s)* and provide the information needed by methods.
- A *variable* is something that can store data.
- An instruction to the computer is called a *statement*; it ends with a semicolon.
- The grammar rules for a programming language are called the *syntax* of the language.

Programming

- Programming is a creative process.
- Programming can be learned by discovering the techniques used by experienced programmers.
- These techniques are applicable to almost every programming language, including Java.
Object-Oriented Programming

- Our world consists of objects (people, trees, cars, cities, dogs, etc.).
- Objects have state and behavior. A car has state (model, color, fuel level, etc), and behavior (start, change gear, brake, etc).
- An object’s behavior can affect its state and the state of other objects.
- Object-oriented programming (OOP) treats a program as a collection of objects, each with behaviors and state.

OOP Terminology

- Behaviors are included as methods.
- State is contained in a set of attributes.
- A class defines the methods and attributes.
- An object is an instance of a class. A program may have several instances of a class. (myCar, yourCar, herCar)
- Each object (instance of a class) has a the same set of methods, but its own attribute values. (state)
Scanner Class

Scanner keyboard = new Scanner(System.in);
keyboard.nextLine();

• **Scanner** is a class
• **keyboard** is an instance of the Scanner class
• **nextLine()** is a method (behavior) of the Scanner class.
• The Scanner has an internal attribute (state) which stores the delimiter.

Scanner Class

• The object performs an action when you *invoke* or *call* one of its methods

    `objectName.methodName(argumentsTheMethodNeeds);`
Algorithms

• By designing methods, programmers provide actions for objects to perform.
• An algorithm describes a means of performing an action.
• Once an algorithm is defined, expressing it in Java (or in another programming language) usually is easy.

Algorithms

• An algorithm is a set of instructions for solving a problem.
• An algorithm must be expressed completely and precisely.
• Algorithms usually are expressed in English or in pseudocode.
Next class

• More if-else

→ Reading Assignment: Chapter 2