THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

COMP 110
Introduction to Programming

Tuesday September 16, 2014

Jay Aikat
Fall 2014
TR 9:30 – 10:45, GS-G100

Previous Class

- What did we discuss?

Announcements

- **Assignment 2 DUE today**
- Lab2 due Mon, Sep 22
- Readings: 4.2

- Getting your assignments checked

Review…

- QUIZ
- String declaration

  String myString = “it’s a sunny day today!”;
- char declaration

  char myChar = ‘a’;
- calling method with object

```
System.out.println(“Length of my string is ” + myString.length());
```

Using If-Else

- Pay attention to **the brackets {}**
  - As a good habit, don't discard them, even if you have only one statement in it
    - The only exception: **multibranch if-else**

Using If-Else

- Never put a semicolon after ***if*** or ***else***
  - ***if (inputInt > 0);***
    System.out.println("What's happening now?");
- Compiler will interpret it as
  - ***if (inputInt > 0)***
    { ;}
    System.out.println("What's happening now?");

Nested If-Else Statement

- Without brackets, every ***else*** will automatically match the nearest ***if***

```
if ( num < 50 )
    if ( num < 25 )
        System.out.println("Number is less than 25");
else
    System.out.println("Number is greater than 50");
```

- Is this piece of code correct?

Nested If-Else Statement

- Without brackets, every ***else*** will automatically match the nearest ***if***

```
if ( num < 50 ) {
    if ( num < 25 )
        System.out.println("Number is less than 25");
    else
        System.out.println("Number is between 25 and 50");
}
```

- **Use brackets and indentation to avoid such errors**

Nested If-Else Statement

- Without brackets, every ***else*** will automatically match the nearest ***if***

```
if ( num < 50 ) {
    if ( num < 25 )
        System.out.println("Number is less than 25");
}
else
    System.out.println("Number is greater than 50");
```

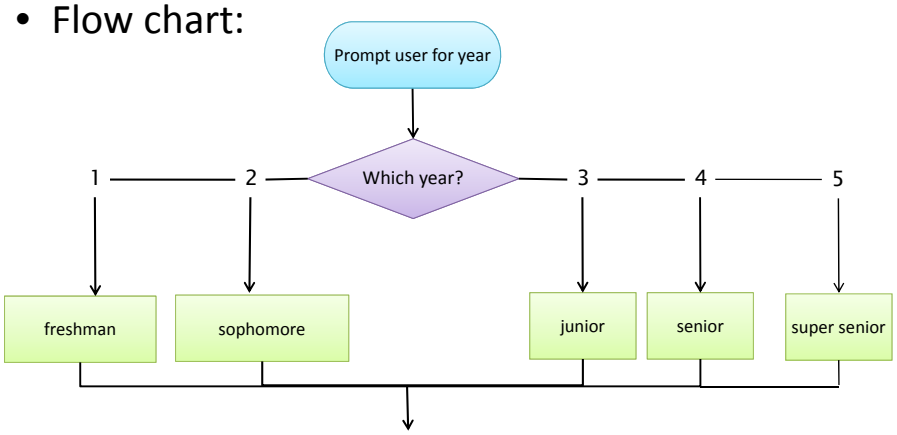
- **Use brackets and indentation to avoid such errors**

Multibranch If-Else Statement

- Example
  - Write a program that takes as input your year in college (as an integer) and outputs your year as freshman, sophomore, junior, senior, or super senior

Multibranch If-Else Statement

- Flow chart:



Multibranch If-Else Statement

- We can write a program like this

```
if (year == 1)
System.out.println("freshman");
          else {
                    if (year == 2)
                              System.out.println("sophomore");
                    else {
                              if (year == 3)
                                        System.out.println("junior");
                              else {
                                        if (year == 4)
                                                  System.out.println("senior");
                                        else {
                                                  if (year == 5)
                                                            System.out.println("super senior");
                                                  else
                                                            System.out.println("huh?");
                                                  }
                                        }
                              }
                    }
```

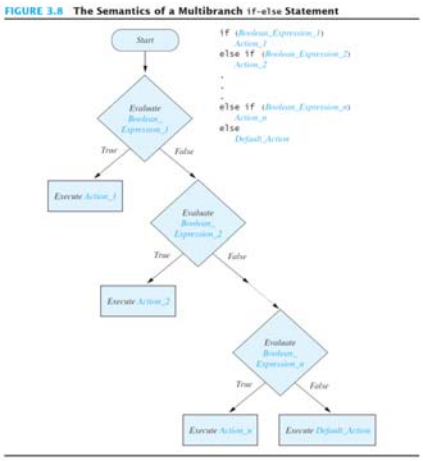
Multibranch If-Else Statement

- Because the previous version is too tedious, we use the **multibranch** statement instead
  - It is not a new syntax rule. We only ignore the brackets so that the logical structure is clear.

```
if (year == 1)
          System.out.println("freshman");
else if (year == 2)
          System.out.println("sophomore");
else if (year == 3)
          System.out.println("junior");
else if (year == 4)
          System.out.println("senior");
else if (year == 5)
          System.out.println("super
senior");
else
          System.out.println("huh?");
```


Multibranch If-Else Statement

- Though all the branches look equal, there is a precedence order among them
  - Only the **first** satisfied branch will be executed



Multibranch If-Else Statement

- What's wrong with this piece of code?

```
if (num < 50)
    System.out.println("Number is less than 50");
else if (time < 25)
    System.out.println("Number is less than 25");
else
    System.out.println("Number is greater than 50");
```


Multibranch If-Else Statement

- What's wrong with this piece of code?

```
if (num < 50)
    System.out.println("Number is less than 50");
else if (time < 25)
    System.out.println("Number is less than 25");
else
    System.out.println("Number is greater than 50");
```

Will this branch get executed?

Switch Statement

```
if (year == 1)
     System.out.println("freshman");
else if (year == 2)
     System.out.println("sophomore");
else if (year == 3)
     System.out.println("junior");
else if (year == 4)
     System.out.println("senior");
else if (year == 5)
     System.out.println("super senior");
else
     System.out.println("huh?");
```

```
switch (year) {
  case 1:
    System.out.println("freshman");
    break;
  case 2:
    System.out.println("sophomore");
    break;
  case 3:
    System.out.println("junior");
    break;
  case 4:
    System.out.println("senior");
    break;
  case 5:
    System.out.println("super senior");
    break;
  default:
    System.out.println("unknown");
  break;
}
```


Switch Statement

- Syntax rules:
  - If *controlling expression == case_label_n*, then execute *statements_n*;
  - The *break* means: jump out of the statement. Without the *break*, next statement will also be executed

```
switch (controlling expression
        /variable)
{
    case case_label_1:
        statements_1;
        break;
    case case_label_2:
        statements_2;
        break;
    default:
        statements;
        break;
}
```

Switch Statement

- Without a break (after *case 'A'* and *case 'C'*), the program will continue to the next case and execute that statement
- Pay attention to colons and semicolons

```
switch (eggGrade) {
  case 'A':
  case 'a':
    System.out.println("Grade A");
    break;
  case 'C':
  case 'c':
    System.out.println("Grade C");
    break;
  default:
    System.out.println("We only buy
grade A and grade C.");
    break;
}
```


Switch Statement

- The *default* case is optional
  - It means "everything else"
- The case labels must be of **the same type** as controlling expression
- The controlling expression can only be ***int***, ***short***, ***byte*** or ***char***
  - Why not ***float*** or ***double***?
  - Why not *String*?
    - Hint: Think about "=="

Multibranch vs. Switch

- Switch statement is more straightforward if you only use "==" to check a single expression/variable
  - Using proper ***break*** can allow for shorter code
- Multibranch if-else statement is more powerful
  - You can use it to check the range of a variable
  - You can use it to check the value of ***float/double/String***
  - You can check more than one variable

LOOPS

- Loops are designed to repeat instructions
  - Think about the requirement: Print number 1 to 10
    - It's easy
      - System.out.println("1");
      - System.out.println("2");
      - ……
  - Think about the requirement: Print number 1 to 100
    - We can still do this
  - Let the user input a value n, then print 1 to n
    - We are in trouble……

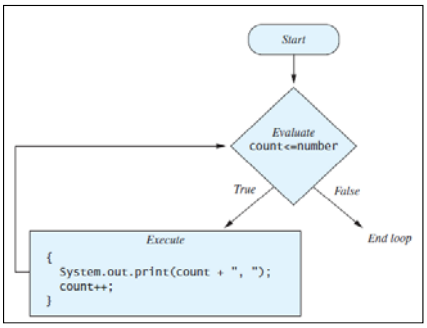
Loop Statement

- What is the pseudo code to fulfill the requirement?
  - Count to 1, if 1<=n, write it down, otherwise stop
  - Count to 2, if 2<=n, write it down, otherwise stop
  - Count to 3, if 3<=n, write it down, otherwise stop
  - ……
  - Count to i, if i<=n, write it down, otherwise stop
  - Count to i+1, if i+1<=n, write it down, otherwise stop
  - ……
  - While a counter<=n, write it down, increase the counter. Otherwise stop

While Statement

- Flow of while statement
  - Start from expression evaluation
  - As long as it's true, repeat instructions in brackets

```
while (count <= number) {
  System.out.println(count);
  count++;
}
```



While Statement

- You have to do some initialization before the statement
- The loop body typically contains an action that ultimately causes the controlling boolean expression to become false.

```
number = keyboard.nextInt();
count = 1;
while (count <= number) {
  System.out.println(count);
  count++;
}
```


While Statement

- Usually there is a counter variable in the statement
  - You can use it in different ways
- Requirement: print the odd numbers from 1 to 10000

```
int count = 1;
while (count < 10000) {
  System.out.println(count);
  count += 2;
}
```

```
int count = 1;
while (count * 2 - 1 < 10000) {
  System.out.println(count * 2 - 1);
  count++;
}
```

Do-While Statement

- Another pseudo code to fulfill the requirement?
  - Write down 1 and count to 2, continue if 2<=n, otherwise stop
  - Write down 2 and count to 3, continue if 3<=n, otherwise stop
  - Write down 3 and count to 4, continue if 4<=n, otherwise stop
  - ……
- See the difference?
  - Count to 1, if 1<=n, write it down, otherwise stop
  - Count to 2, if 2<=n, write it down, otherwise stop
  - Count to 3, if 3<=n, write it down, otherwise stop
  - ……

Do-While Statement

- The main difference: do-while statement will at least execute the body statements once
  - If start from ***count = 1*** and ***number = 0***
  - While statement will output **nothing**
  - Do-While statement will output **1**, then stop

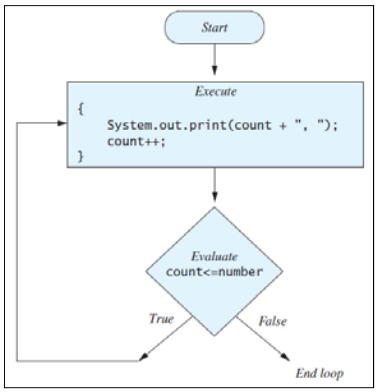
```
while (count <= number) {
  System.out.println(count);
  count++;
}
```

```
do {
  System.out.print(count);
  count++;
} while (count <= number);
```

Do-While Statement

- Flow of do-while statement
  - Start from body statements
  - Repeat instructions in brackets as long as the expression is true

```
do {
  System.out.print(count);
  count++;
} while (count <= number);
```



Do-While Statement

- Don't forget the semicolon after the whole statement
- Not recommended, unless you really need the body statement to be executed once

```
do {
  System.out.print(count);
  count++;
} while (count <= number);
```

While and Do-While Statements

- These two pieces of code perform identically

```
initialization;
do {
  body_statments
} while (boolean_expression);
```

```
initialization;
body_statements;
while (boolean_expression) {
  body_statements;
}
```


Next class (Thu, Sep 18)

- Prof. Gary Bishop