THE UNIVERSITY of NORTH CAROLINA at CHAPEL HILL

COMP 110
Introduction to Programming

Tuesday September 23, 2014

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

Previous Class

- What did we discuss?

Announcements

- **Midterm – Tue, Oct 14 (before Fall Break)**
- **Assignment 3 OPEN today**
- Lab3 due Thu, Sep 25

Notes…

- Note: To exit your program before end of main method

  System.exit (0);
- Code Style Guidelines: http://cs.unc.edu/~aikat/courses/comp110/assignments/style-guidelines.pdf

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……

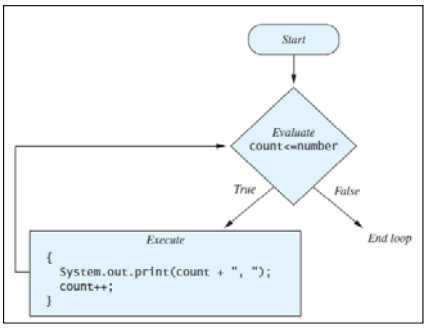
Loop Statement

- 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

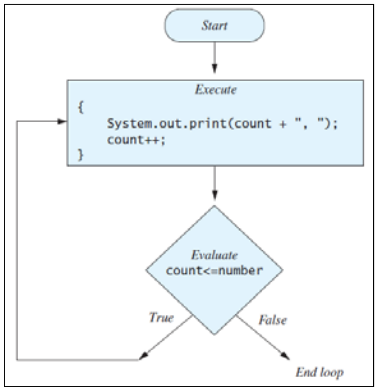
```
while (count <= number) {
  System.out.println(count);
  count++;
}
```

```
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;
}
```

Infinite Loops

- Always make sure that your loop will end
  - Never forget to change the counter

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

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

==

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


Infinite Loops

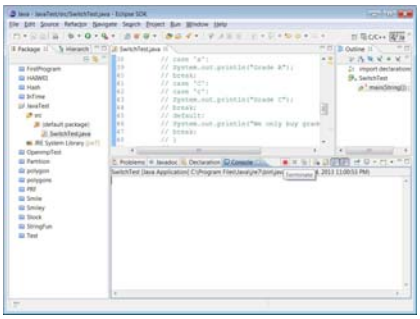
- Always make sure that your loop will end
  - Never forget to change the counter
  - Use comparison rather than "==" or "!="in the control expression
  - Know whether your counter is increasing or decreasing

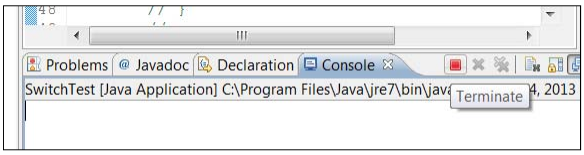
```
while (count != number) {
  System.out.println(count);
  count+=2;
}
```

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

Infinite Loops

- If you wrote an infinite loop and executed it
- Use the **terminate** button of eclipse
  - If it is red, the program is running





Infinite Loops

- Infinite loop is not a syntax error. It's a logical error
- eclipse will not help you in this case
- Write pseudo code, think, and rethink before coding

For Statement

- Is there a better way to organize the code?
- For statement (or usually called *for loop*)
  - Used to execute the body of a loop a **fixed number** of times

```
number = keyboard.nextInt();
count = 1;
while (count <= number) {
  // all the actions
  count++;
}
```

```
number = keyboard.nextInt();
int count;
for (count = 1;
   count<=number; count++) {
  // all the actions
}
```

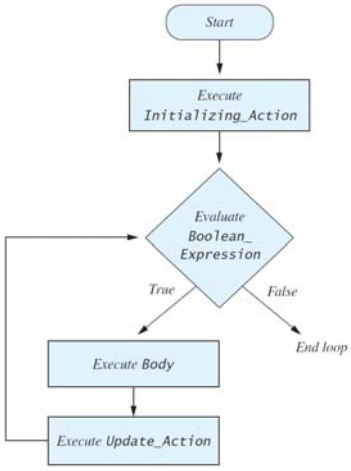

For Statement

- Syntax:
  - **for** (*Initializing_Action; Boolean_Expression; Update_Action){*
    Body;
    }

```
for (count = 1; count <= number; count++) {
  // all the actions
}
```

For Statement

- Flow chart
  - **for** (*Initializing_Action; Boolean_Expression; Update_Action){*
    Body;
    }



For Statement

- Unrolled code

```
for (count = 1;
count<= 2; count++)
{
  // all the actions
}
```

```
count = 1; // initialize for only once
if (count <= 2) {// count == 1, so yes
        // all the actions
        count++;
}
if (count <= 2) {// count == 2, yes again
        // all the actions again
        count++;
}
if (count <= 2) {// count == 3, so no
        // no action;
        // no count++;
}
// stop
```

For Loop: Don't Overcount

- Repeat 3 times

```
for (int count = 1; count <= 3; count++) {
  // all the actions
}
```

- Repeat 3 times

```
for (int count = 0; count < 3; count++) {
  // all the actions
}
```

- Repeat **4 times**!

```
for (int count = 0; count <= 3; count++) {
  // all the actions
}
```


For Loop: Infinite Loop

- Still, if you get things wrong, it may never end

```
int num = 3;
// initializing action; boolean expression; update action
for (count = 5; count >= num; count++)
{
      System.out.print(count + ", ");
}
```

Ending a Loop

- If you know number of loop iterations?
  - Count-controlled loops
  - ***for(count = 0; count < iterations; count++)***
- User controlled ending
  - Ask-before-iterating (e.g. “yes/no”)
  - Sentinel value

Next class (Thu, Sep 25)

- More on loops