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
  • Assignment 1: Due Tuesday, Sep 22 @ 11:55 PM
    http://cs.unc.edu/~aikat/courses/comp110/assignments/Assignment1

• Loops

CS/IT Career Fair

• Friday (9/18) from 11am - 2pm
• Great Hall of the Carolina Union
• 56 registered companies

Some advice 😊
1) business casual -- you want the recruiters to know that you take this seriously
2) research the companies that you want to stop and visit with: they don't much care for "What does your company do?" You should have an intelligent question to ask
3) bring your resume (preferably checked out by someone in career services)
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 Loop**

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

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

**While Loop**

- 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.

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

- 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

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

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

Infinite Loops

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

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

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

```java
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

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

```java
while (count < number) {
    System.out.println(count);
    count--;  // Corrected comparison operator
}
```

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 Loop**

- 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

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

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

• Syntax:
  – for (Initializing_Action; Boolean_Expression;
    Update_Action){
    Body;
  }

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

For Loop

• Flow chart
  – for (Initializing_Action;
    Boolean_Expression;
    Update_Action){
    Body;
  }

```plaintext
Start

Evaluate Boolean Expression

Body

Execute Body

End loop
```
For Loop

• Unrolled code

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

```c
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

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

• Repeat 3 times

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

• Repeat 4 times!

```c
for (int count = 0; count <= 3; count++)
{
    // all the actions
}
```
For Loop: Infinite Loop

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

```java
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

- More loops
- And Quiz3 – if-else and while loops