Repetition Statements
(Loops)

CS 99 – Summer 2000
Michael Clarkson
Lecture 5

Administration

- Prelim 1 Review Session Tonight
  - 7:30-8:30pm
  - Upson 211
  - Bring questions or it will be over quickly!
- Lab 4 in progress, due tomorrow
- Lab 5 posted today
- Prelim 1 on Wednesday in class, covers:
  - Lectures & Labs 1-4
  - assigned reading (esp. 1.2, 1.4, 1.5, 2.2, 2.4)

Agenda

- Repetition statements
- Three repetitions statements in Java:
  - for
  - while
  - do

Repetition

- Computers are great at performing repeated tasks
- So far, we don’t know how to repeat tasks (conveniently) in a program
- Examples:
  - Add all the integers from 1 to 100
  - Calculate grades for an entire class

Repetition [2]

- The three statements for repetition in Java are:
  
  ```
  for(; ; ;) {
  ... } while(;) {
  ... } do ( ; ) while(;) 
  ```

- Most loops share these characteristics:
  - a variable is assigned some value before the loop
  - the variable’s value changes at some point in the loop
  - repetition continues until some condition is true (e.g., the variable reaches some predetermined value)

Repetition [3]

- Pretest loop: a loop that uses a condition to control whether or not the body of the loop is executed before going through the loop:
  - condition is true, body is executed
  - condition is false, body is skipped
  - while, for

- Posttest loop: executes the body of the loop, then checks a condition to decide whether to execute it again
  - condition is true, body is executed again, and condition checked again
  - condition is false, move on to next executable statement
  - do
Repetition [4]

- Variable repetition: the number of times the loop body will execute is unknown
  - e.g., adding numbers the user enters until the sum is greater than 100
  - while, do
- Fixed repetition: the number of times the loop body will execute is predetermined (but not nec. constant)
  - e.g., adding integers from 1 to 100
  - for, (while, do)

Repetition [5]

- while, do, and for loops are all equivalent in that each can be rewritten as the others
  - though it may require the addition of one or more statements
- However, each loop is more appropriate at different times, based on whether you want fixed or variable repetition, and pretests or posttests

for Loops

- The for loop is:
  - pretest
  - fixed repetition
  - A convenient structure for writing certain types of loops more concisely than while allows
- It combines 3 statements into one

First for

// sum the integers from 1 to 100
sum = 0;
for (i = 1; i <= 100; i++) {
  sum = sum + i;
}

- Execution:
  - Initialize i to 1
  - Check if i is less than or equal to 100
    - If so, execute body
    - If not, stop repeating
  - Update i by incrementing it
  - Repeat

Syntax of for

for (initializer; condition; update) {
  ...
}

- initializer and update are statements
  - initializer can be a variable declaration, with scope through the end of the block that the for statement is in
- condition is a boolean expression
- the first line is called the loop header

Control Flow of for
Common for loops

// count from low to high using var
for (int var = low; var <= high; var++) {
   ...
}

// count from high down to low using var
for (int var = high; var >= low; var--) {
   ...
}

These types of for loops should never change the value of var inside of the body!

for Example #1

- Print the numbers 1-10 along with their squares and cubes

```java
for (int i = 1; i <= 10; i++) {
   System.out.println(i + "\t" + i*i + "\t" + i*i*i);
}
```

for Example #2

- Average a set of numbers entered by the user. Begin by inputting how many numbers are in the set.

```java
System.out.print("Enter how many numbers there are: ");
int size = Console.readInt();
int sum = 0;
for (int i = 1; i <= size; i++) {
   System.out.print("Enter #" + i + ": ");
   sum += Console.readInt();
}
double average = (double) sum / size;
```

for Example #3

- Sum the multiples of 7 between 1 and 1000

```java
int sum = 0;
for (int num = 7; num <= 1000; num += 7) {
   sum += num;
}
double average = (double) sum / size;
```

A variable used in a loop to keep a sum of the value of some other variable is called an accumulate. The variable that is declared, checked, and updated is called the loop index or loop control variable.

for Example #4

- Use a for loop to produce the following output:

```java
for (int spaces = 3; spaces >= 0; spaces--) {
   for (int i = 1; i <= spaces; i++) {
      System.out.print(" ");
   }
   System.out.println("****");
}
```
while Loops

- The while loop is:
  - pretest
  - variable repetition
  - very similar to an if statement

First while

```java
int num = 1, sum = 0;
while (num <= 100) {
    sum = sum + num;
    num++;
}
```

Syntax and Semantics of while

while (boolean-expression) {
    ...
}

- Evaluate the expression
- If it is true, execute the body of the loop and repeat
- If it is false, transfer control to the next statement after the loop

Flow of Control of while

true

body

false

next statement

Note that the body of the loop might never be executed.

Rewriting for as while

```java
sum = 0;
for (int i = 1; i <= 100; i++) {
    sum = sum + i;
}
```
Example #2

- A sentinel is an input value that indicates the end of input
  - e.g., "Enter a number, -999 to quit:"
- Average a set of numbers input from the user, terminated by a sentinel

```java
int count = 0;
int sum = 0;
System.out.print("Enter a number, -1 to quit: ");
int num = Console.readInt();
while (num != -1) {
    count++;
    sum += num;
    System.out.print("Enter a number, -1 to quit: ");
    num = Console.readInt();
}
if (count > 0) {
    double average = (double) sum / count;
    System.out.println("average = " + average);
}
```

Example #3

- What’s wrong with this loop?

```java
// print powers of 3 between 1 and 100
pow3 = 1;
while (pow3 != 100) {
    System.out.println(pow3);
    pow3 = pow3 * 3;
}
```

- Infinite loop, should be:

```java
// print powers of 3 between 1 and 100
pow3 = 1;
while (pow3 < 100) {
    System.out.println(pow3);
    pow3 = pow3 * 3;
}
```

Example #4

- What’s wrong with this loop?

```java
// print the first five powers of 3 between 1 and x, inclusive
pow3 = 1;
count = 1;
while (count <= 5 && pow3 <= x) {
    System.out.println(pow3);
    pow3 = pow3 * 3;
}
```

- Again, an infinite loop, should be:

```java
// print the first five powers of 3 between 1 and x, inclusive
pow3 = 1;
count = 1;
while (count <= 5 && pow3 <= x) {
    System.out.println(pow3);
    pow3 = pow3 * 3;
    count++;
}
```
**do Loops**

- **Syntax:**
  ```java
do {
  ...
} while (boolean-expression);
```

- **Semantics:**
  - Execute the body
  - Check the condition
    - If true, repeat
    - If false, transfer control to the statement after the loop

**Flow of control in do**

```
body

condition?
  true

false
next statement
```

Note that the body of the loop is always executed at least once.

**do Example #1**

- Average a set of numbers input from the user, terminated by a sentinel
  ```java
  int count = 0;
  int sum = 0;
  do {
    System.out.print("Enter a number, -999 to quit: ");
    int num = Console.readInt();
    if (num != -999) {
      count++;
      sum += num;
    }
  } while (num != -999);
  if (count > 0) {
    double average = (double) sum / count;
    System.out.println("average = " + average);
  }
  ```

**do Example #2**

- Data validation
  ```java
  String response;
  do {
    System.out.println("Do you wish to continue? "
    + "(yes or no): ");
    response = Console.readString();
  } while ( !(response.equals("yes")
    || response.equals("no") ) );
  ```