University of British Columbia
CPSC 111, Intro to Computation
2009W2: Jan-Apr 2010

Tamara Munzner

Loops I

Lecture 17, Fri Feb 12 2010

borrowing from slides by Kurt Eiselt

http://www.cs.ubc.ca/~tmm/courses/111-10

Reading

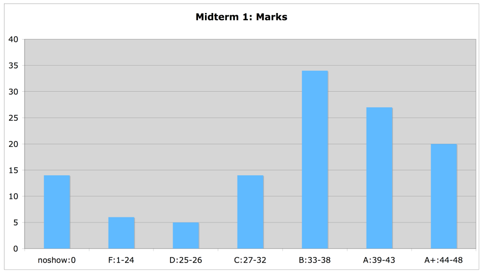
- This week: Chapter 5 all (5.1-5.4)
  - second edition: Chap 6
- Next week: Chapter 6 all (6.1-6.4)
  - second edition: Chap 7

News

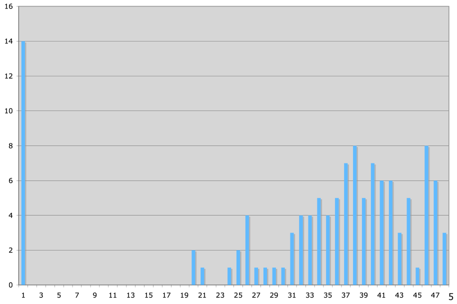
- Next week is reading week
  - no lectures or labs or tutorials
- Midterms returned today
  - Grades, statistics already posted on WebCT
  - returned end of class, line up by last name (A-Z)

Midterm Marks Distribution

- marks will not be scaled



Midterm Distribution: Detailed



Regrading

- Reminder: protocol for regrade requests
  - read solution and marking scheme first, carefully
    - no regrade requests accepted until at least 24 hours after material is handed back
      - exception: arithmetic errors
  - regrade requests must be in writing (paper or email)
    - assignments: to marker (listed on cover sheet)
      - if still have dispute after discussion with TA, can escalate to instructor
    - exams: to instructor

Recap: Comparing Strings

- Relational operator == is wrong way to compare

```
String name1 = "Bubba";
String name2 = "Bubba";
System.out.println(name1 == name2);  // prints false
```

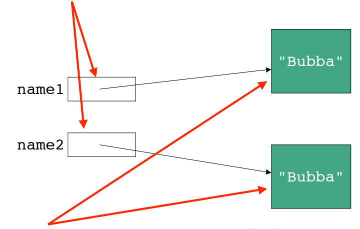
- **`equals`** method is right way to compare Strings

```
String name1 = "Bubba";
String name2 = "Bubba";
System.out.println(name1.equals(name2)); // prints true
```

  - why? diagrams will help

Recap: Comparing Strings

- **`name1 == name2`** : two different references, **`false`**



- **`name1.equals(name2)`** : contents same, **`true`**

Recap: Short-Circuting Evaluation

- Java evaluates complex expressions left to right
  - short-circuiting: Java stops evaluating once value is clearly true or false
    - aka lazy evaluation

```
if ((b > a) && (c == 10))
    System.out.println("when b<=a short-circuit");
if ((b > a) || (c == 10))
    System.out.println("when b>a short-circuit");
```

- Corollary: avoid statements with side effects

```
if ((b > a) || (c++))
    System.out.println("Danger Will Robinson!");
```

Recap: Conditional Syntax

`if` (boolean expression) statement
`else if` (boolean expression) statement
- optional: zero, one, or many

`else` statement
- optional

- **`if, else`** are reserved words
- parentheses mandatory
- statement can be
  - single line
  - block of several lines enclosed in **{ }**

Recap: Comparing Floats/Doubles

- Relational operator for equality not safe for floating point comparison

```
if (.3 == 1.0/10.0 + 1.0/10.0 + 1.0/10.0))
  System.out.println("Beware roundoff error");
```

- Check if difference close to 0 instead

```
if (Math.abs(f1 - f2) < TOLERANCE)
   System.out.println (“Essentially equal.”);
```

Recap: Comparing Characters

- Safe to compare character types with relational operators

```
char c = 'a';
char d = 'b';
if (c == d)
   System.out.println("they match");
```

Recap: Switch Syntax

```
switch ( expression ) {
  case value:
      statements
      break;
  case value:
      statements
      break;
  default:
      statements
```

- **`switch, case, break`** are reserved words
- expression and value must be int or char
  - value cannot be variable
- break important, or else control flow continues to next set
- statements can be one line or several lines
- default executed if no values match expression

Objectives

- Practice with conditionals
- Understand basic loops

```
public class NestTest3 {
    public static void main (String[] args) {
        respondToName("Flocinaucinihilipiliphication");
        respondToName("Supercalifragilisticexpialidocious");
        respondToName("Ambrose");
        respondToName("Kermit");
        respondToName("Miss Piggy!!!");
        respondToName("Spot");
        respondToName("me");
    }
    public static void respondToName(String name) {
        System.out.println("You're named " + name);
        if (name.length() > 20) {
            System.out.println("Gosh, long name");
            System.out.println("Keeping typists busy...");
        } else if (name.length() > 30) {
            System.out.println("Over the top");
        } else if (name.length() < 10) {
            if (name.charAt(0) == 'A')
                System.out.println("You're first");
            else if (name == "Kermit")
                System.out.println("You're a frog");
                System.out.println("I love animals");
        } else if (name.equals("Spot")) {
            System.out.println("You're spotted");
        } else if (name.length() < 3) {
            System.out.println("Concise!");
        }
    }
}
```

Repetition, Iteration, Loops

- Computers good at performing same task many times
- Loops allow repetitive operations in programs
  - aka iteration statements, repetition statements
- Loops handy in real life too

Climbing Stairs

- Am I at the top of the stairs?

Climbing Stairs

- Am I at the top of the stairs?
- No.
- Climb up one step.

Climbing Stairs

- Am I at the top of the stairs?
- No.
- Climb up one step.
- Am I at the top of the stairs?

Climbing Stairs

- Am I at the top of the stairs?
- No.
- Climb up one step.
- Am I at the top of the stairs?
- No.
- Climb up one step.

Climbing Stairs

- Am I at the top of the stairs?
- No.
- Climb up one step.
- Am I at the top of the stairs?
- No.
- Climb up one step.
- Am I at the top of the stairs?

Climbing Stairs

- Am I at the top of the stairs?
- No.
- Climb up one step.
- Am I at the top of the stairs?
- No.
- Climb up one step.
- Am I at the top of the stairs?
- No.
- Climb up one step.
- Am I at the top of the stairs?
- No.
- Climb up one step.
- ...and so on...

Washing Hair

- Lather

Washing Hair

- Lather
- Rinse

Washing Hair

- Lather
- Rinse
- Repeat

Washing Hair

- Lather
- Rinse
- Repeat

- When do you stop??

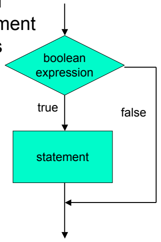
While Statement

```
while (boolean expression)
   body
```

- Simplest form of loop in Java
- Body of loop can be
  - single statement
  - whole block of many statements in curly braces
- Control flow
  - body executed if expression is true
  - then boolean expression evaluated again
  - if expression still true, body executed again
  - repetition continues until expression false
  - then processing continues with next statement after loop

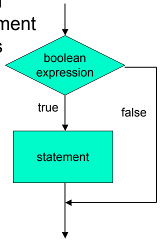
If Versus While Statements

how if statement works

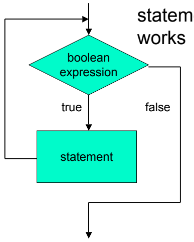


If Versus While Statements

how if statement works

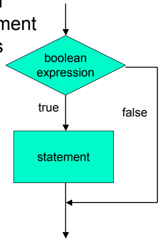


how while statement works

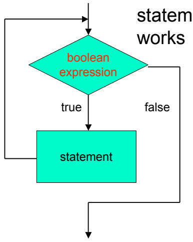


If Versus While Statements

how if statement works



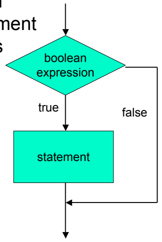
how while statement works



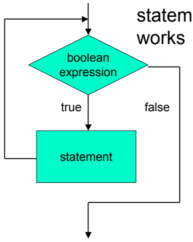
- How can loop boolean change from false to true?

If Versus While Statements

how if statement works



how while statement works



- These diagrams called flowcharts

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

- **while** statement

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

- boolean expression

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

- **while** statement body

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

- statement after **while**
  - control flow resumes here when boolean is false

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

- trace what happens when execute

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

limit 3

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

limit 3 **counter** 1

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

limit 3 **counter** 1 **Is counter <= limit? yes**

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

limit 3 **counter** 1 **Is counter <= limit? yes**

"The square of 1 is 1" printed on monitor

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

limit 3 **counter** 2

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

limit 3 **counter** 2 **Is counter <= limit? yes**

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

limit 3 **counter** 2 **Is counter <= limit? yes**

"The square of 2 is 4" printed on monitor

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

limit 3 **counter** 3

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

limit 3 **counter** 3 **Is counter <= limit? yes**

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

limit 3 **counter** 3 **Is counter <= limit? yes**

"The square of 3 is 9" printed on monitor

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

limit 3 **counter** 4

Using while Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

limit 3 **counter** 4 **Is counter <= limit? NO!**

Using `while` Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

limit 3 **counter** 4 **Is counter <= limit? NO!**

“End of demonstration" printed on monitor

Climbing Stairs Again

- Am I at the top of the stairs?
- No.
- Climb up one step.
- Am I at the top of the stairs?
- No.
- Climb up one step.
- Am I at the top of the stairs?
- No.
- Climb up one step.
- Am I at the top of the stairs?
- No.
- Climb up one step.
- ...and so on...

Climbing Stairs Again

while (I'm not at the top of the stairs)
{
  Climb up one step
}

- Climbing stairs is a while loop!

Using `while` Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter >= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

- change termination condition

Using `while` Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter >= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

- change termination condition
  - body of loop never executed

Using `while` Statements

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter >= counter)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

- change termination condition
  - always true

Infinite Loops

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter >= counter)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 1;
    }
    System.out.println("End of demonstration");
  }
}
```

- if termination condition always true, loop never ends
  - infinite loop goes forever

Infinite Loops

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 3;
    int counter = 1;

    while (counter <= limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter - 1;
    }
    System.out.println("End of demonstration");
  }
}
```

- good termination condition
- but process never gets closer to condition

Infinite Loops

```
public class WhileDemo
{
  public static void main (String[] args)
  {
    int limit = 9;
    int counter = 0;
    while (counter != limit)
    {
      System.out.println("The square of " + counter +
                         " is " + (counter * counter));
      counter = counter + 2;
    }
    System.out.println("End of demonstration");
  }
}
```

- process gets closer to termination condition
- but never satisfies condition, keeps going past it

Another `while` Example

```
public class PrintFactorials
{
  public static void main (String[] args)
  {
    int limit = 10;
    int counter = 1;
    int product = 1;

    while (counter <= limit)
    {
      System.out.println("The factorial of " + counter +
                         " is " + product’\);
      counter = counter + 1;
      product = product * counter;
    }
    System.out.println("End of demonstration");
  }
}
```

- accumulate product

Questions?