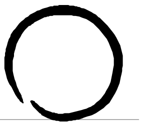
CSE 142

Iteration – Introduction to Loops



Outline for Today

- Iteration – repeating operations
- Iteration in Java – *while* statement
- Shorthand for definite (counting) iterations – *for* statement
- Nested loops

Programming a Teller Machine

- Suppose you are working on the code for a automated teller machine (ATM). Your code should give out the right number of bills when the user withdraws money. The ATM contains $20 and $5 bills.
- Problem: Hand out the right number of $20 and $5 bills to make up *d* dollars. Assume that *d* is a multiple of $5.
  - Best solution would use as many $20s as possible
  - Design an algorithm for this with your neighbors

ATM Algorithm for Dispensing Money

- Design your solution(s) here

ATM Algorithm

- Additional notes

Iteration/Repetition

- The ATM cash algorithm is an example of an *iteration* or repetition – repeatedly perform some operation
- A few more examples
  - Bake the roast; keep checking the internal temperature until it reaches 220 degrees
  - While there are still donuts in the box, eat one
  - Lather, rinse, repeat
  - Simulations/games – science, entertainment
    Repeatedly update actions of objects in the simulation
  - Video – display frames repeatedly
- Practically all interesting programs contain loops

Grow Your Money...

On your birthday, you spend $5000 to buy a Certificate of Deposit which earns 3.5% a year on its balance. There is a $10 "service fee" deducted from the balance each year. How much money will you have at the end of 5 years?

Work with your neighbor to design an algorithm to solve the problem. Write down a brief description of your algorithm (without using Java code).

Iteration in Java: *while* Syntax

- Basic form – *while* statement

```
while ( condition ) {
  list of statements
}
```

- Terminology
  - *condition* is sometimes called the ***loop condition***
  - *list of statements* is often called the ***loop body***

Iteration in Java

- Meaning of

```
while ( condition ) {
   list of statements
}
```

- Repeatedly do the following:
  - Evaluate the *condition*
  - If the *condition* is false, the loop terminates – execution continues with the statement following the loop body (after '}')
  - Execute the list of statements and repeat
- Note: condition is only reevaluated after finishing the *complete* execution of the loop body – not concurrently as loop body statements are executed

Flow Chart

- Another way to visualize loop execution

Exercise – Write Numbers and Squares

- Suppose we want to write a table of numbers and their squares for the numbers 1 to 5
- Brute force ("+" used to combine strings)

```
System.out.println(1 + " squared =  " +  1*1);
System.out.println(2 + " squared =  " +  2*2);
System.out.println(3 + " squared =  " +  3*3);
System.out.println(4 + " squared =  " +  4*4);
System.out.println(5 + " squared =  " +  5*5);
```

- How could we improve this?

What We're Really Trying to Do

- We really want to repeatedly execute

```
System.out.println(k + " squared =  " + k*k);
```

with k taking on the values 1 through 5 on successive repetitions

- Solution (?)

```
k = 1;
while (k <= 5) {
   System.out.println(k + " squared = " + k*k);
}
```

  - Does this work? How can we tell?

Tracing Loops

- You can “desk-check” code by hand simulating the steps the computer performs
- Desk-checking a loop involves simulating each iteration of the loop
- Check:

Exercise

- In arithmetic, n! (read as “n factorial”) is defined to be
  1 * 2 * 3 * 4 * ... * (n-1) * n
- Exercise: write a loop to compute 7! and check it
  - Hint(?): try writing this out by hand, then figure out what statements can be repeated while some values in them change

Loop to Calculate 7!

- Your code here

Check: Trace

Counting Loops – For Statement

- The loops we've seen so far all execute a *definite* number of times with some variable taking on a sequence of values
- Java, like most other languages, provides a special statement to make this convenient – the ***for statement***

```
for (initialization; condition; update) {
   list of statements
}
```

Counting Loops

- Example: Print the numbers 1 through 100
  - With a *while* loop

```
int k = 1;
while (k <= 100) {
   System.out.println(k);
   k = k + 1;
}
```

  - With a *for* loop

```
for (int k = 1; k <= 10; k = k + 1) {
   System.out.println(k);
}
```

- These mean exactly the same thing
- Style point: Java programmers would normally use *for* instead of *while* for a counting loop.

For Loops and *While* Loops

- A *for* statement is a convenient shorthand for an equivalent while statement

```
for (initialization; condition; update) {
   list of statements
}
```

has (for our purposes) exactly the same meaning as

```
initialization;
while ( condition ) {
   list of statements
   update
}
```

  - Note that the update executes after the loop body

For Statement Flow Chart

Factorial as a Method

- A calculation like factorial is a logically coherent operation. It makes sense to package it as a method. Complete the implementation below using a *for* statement

```
/** Return the value n! */
public int factorial(int n) {




}
```

Shortcut

- In loops like

```
for (int k = 1; k <= 10; k = k + 1) {
   ...
}
```

the update “k = k + 1” is so common that Java provides a shorthand way to write it: “k++”

- Equivalent loop

```
for (int k = 1; k <= 10; k++) {
   ...
}
```

Double Your Money

- Problem: Suppose you have invested $1000 at 3% annual interest (meaning that each year, 3% of the present value of the investment is added to it). How many years will it take to double the original investment?
- Analysis: repeatedly increase the investment value by 3% until it reaches $2000. Count how many times this has to be done.

A Non-Counting Iteration

- In this problem, the operation needs to be repeated until something happens (value >= $2000)
  - We don’t know how long this will take
- This is an ***indefinite iteration*** – the number of repetitions needed is not known in advance
- A *while* loop is appropriate here

Double Your Money

- Your Code Here

Nested Loops

- How would you print the following figure?

```
* * * * *
* * * * *
* * * * *
```

  - Useful information:

    System.out.print("*"); will print a single "*"

    System.out.println("*"); will print a single "*", then move to the beginning of the next output line

- How would your answer need to be changed if we changed the number of rows or columns?

Analysis

Solution

Check – Trace the Code

Another Problem: A Multiplication Table

- How would we print the following table?

```
        1    2    3    4
   1    1    2    3    4
   2    2    4    6    8
   3    3    6    9   12
```

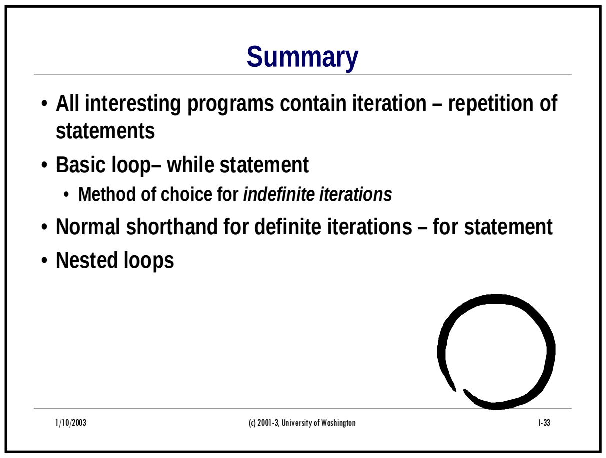
- Analysis

Multiplication Table Code

- Your Solution Here

Multiplication Table Check

- Trace your code here



Summary

- All interesting programs contain iteration – repetition of statements
- Basic loop– while statement
  - Method of choice for *indefinite iterations*
- Normal shorthand for definite iterations – for statement
- Nested loops

1/10/2003 (c) 2001-3, University of Washington I-33