



Principles of Computer Science I

Prof. Nadeem Abdul Hamid
CSC 120 – Fall 2006
Lecture Unit 7 - Iteration



1




Lecture Outline

- Programming loops (iteration)
- Simple GUI animation
- Understand common loop errors
- Nested loops
- Processing input
- File input/output
- Random numbers and simulations

CSC120 — Berry College — Fall 2006


2



Cannonball

- Exercise P7.3
- [Cannonball.java](#)
- [CannonTester.java](#)

3



while Loop


- Looping = Iteration = Repetition

```
while ( condition )
    statement
```

- while statement repeatedly executes a block of code as long as condition is true

```
double curTime = 0.00;
while ( curTime <= 20.0 ) {
    ball.updatePosition( deltaT );
    curTime += deltaT;
}
```

4



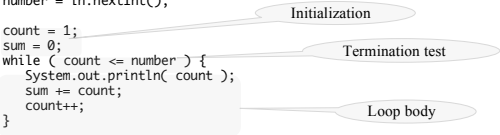
Counting Program

```
int count, number, sum;


System.out.println( "Enter a number to count up to: " );
Scanner in = new Scanner( System.in );
number = in.nextInt();

count = 1;
sum = 0;
while ( count <= number ) {
    System.out.println( count );
    sum += count;
    count++;
}

System.out.println();
System.out.println( "The sum is: " + sum );
```



5



Side Topic: Simple GUI Animation

- Basic idea: Add a Timer object to your component class
 - Every time the timer goes off (e.g. 10 msec) update the display
- Imports needed


```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JComponent;
import javax.swing.Timer;
```
- [CannonballComponent.java](#)
- [CannonballViewer.java](#)

6

GUI Animation Skeleton

```
public class ComponentName
    extends JComponent
    implements ActionListener {
    ...

    /**
     * Starts the timer going by constructing a Timer object with the
     * frequency (in milliseconds) of Timer activations and the object ('this')
     * that will be handling the Timer events
     */
    public void animate() {
        timer = new Timer( (int)Math.round(deltaT * 1000), this );
        timer.start();
    }

    /**
     * Processes a Timer activation event
     */
    public void actionPerformed((ActionEvent e) {
        // Code to update the state of the object
        // should go here
        ...

        repaint(); // repaint the component on the screen
    }

    Timer timer; // timer instance variable
    ... // other instance variables (Fields)
}
```

7

Infinite Loops

```
count = 1;
sum = 0;
while ( count <= number ) {
    System.out.println( count );
    sum += count;
}
```

```
// (two errors in this code?)
count = number; // count down from number
sum = 0;
while ( count <= number ) {
    System.out.println( count );
    sum += count;
    count++;
}
```

- Stop a running program using 'Ctrl'+ 'c' keys

8

Off-by-One Errors

```
count = 1;
sum = 0;
while ( count < number ) {
    count++;
    System.out.println( count );
    sum += count;
}
```

- Common type of error when programming loops
- Work through simple test cases to avoid these errors
- Common issues:
 - Should variable start at 0 or 1 ?
 - Should test condition be < or <= ?
 - Where should the loop variable be updated?
- Note: when processing strings, loops often start at 0 and use <

9

do Loop

- Executes the loop body at least once

```
do
    statement
while ( condition );
```

- Common use: Validating input

```
double value;
do {
    System.out.print( "Please enter a positive number: " );
    value = in.nextDouble();
} while (value <= 0);
...
```

10

Replacing do with while

- Introduce a boolean control variable

```
double value;

boolean done = false;
while ( !done ) {
    System.out.print( "Please enter a positive number: " );
    value = in.nextDouble();

    if ( value > 0 ) done = true;
}
...
```

11

A Common Loop Idiom

```
i = start;
while ( i <= end ) {
    ...
    i++;
}
```

- Special syntax supports this idiom

```
for ( i = start; i <= end; i++ ) {
    ...
}
```

12

for Loop

```
for ( initialization ; condition ; update )  
    statement
```

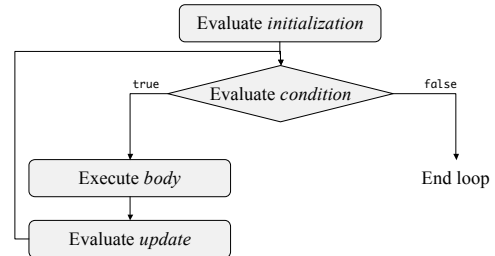
- Use a **for** loop when a variable runs from a starting to end value with constant increment or decrement
- Easy to abuse for notation – any expressions can be put in the header

```
for ( rate = 5; years-- > 0;  
      System.out.print(balance) ) . . .
```

13

How for Loops Work

```
for ( initialization ; condition ; update )  
    body
```



14

for Loops: Common Errors

- Extra semicolon

```
sum = 0;  
for ( i = 1; i <= 10; i++ );  
    sum = sum + i;  
    System.out.println(sum);
```
- Missing semicolon

```
for (years = 1; (balance = balance + balance *  
    rate / 100) < targetBalance; years++)  
    System.out.println(years);
```
- Using != condition instead of <=

```
for ( i = 1; i != 10; i+=2 ) . . .
```

15

Variable Scope in for Loops

- Scope: the area of code in which an identifier (name) is defined/can be used
- Possible to declare a new variable in the header of a for loop - only has scope within the loop

```
for ( int i = 1; i <= n; i++ ) {  
    . . .  
}  
// i is no longer defined here
```

16

Commas in for Statements

- Header of a for loop can contain multiple initializations/updates, separated by commas
- For example, this code:

```
product = 1;  
for ( n = 1; n <= 10; n++ )  
    product = product * n;
```
- Can be rewritten as:

```
for ( n=1, product=1; n<=10; product=product*n, n++ )  
    ;
```
- Considered 'clever' but not necessarily good coding practice

17

Fibonacci Numbers

- Write a program to compute the n'th Fibonacci number

```
f1 = 1;  
f2 = 1;  
  
cur = 3;  
while ( cur <= n ) {  
    long fnew = f1 + f2;  
    f1 = f2;  
    f2 = fnew;  
    cur++;  
}  
  
System.out.println( n + "th Fibonacci number is: " + f2 );
```

```
f1 = 1;  
f2 = 1;  
  
for ( cur = 3; cur <= n; cur++ ) {  
    long fnew = f1 + f2;  
    f1 = f2;  
    f2 = fnew;  
}
```

18

Nested Loops

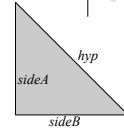
- Often one loop may be *nested* (contained) in another
 - Typical example: Printing table of rows and columns
- Write a program to print out a triangular shape, given a maximum width (e.g. 5):

```
□
□□
□□□
□□□□
□□□□□
```

19

Nested Loops

- Pythagorean Triples
 - Set of integer values such that
- Write a program to find all such triples, where the side lengths are less than 100



- [PythagTriples.java](#)

20

Processing Sentinel Values

- Sentinel: value that is not valid input and indicates the end of input
 - 0 or -1 are not always good sentinel values

```
Enter value, Q to quit: 1
Enter value, Q to quit: 2
Enter value, Q to quit: 3
Enter value, Q to quit: 4
Enter value, Q to quit: Q
Average = 2.5
Maximum = 4.0
```

- [DataSet.java](#)

21

Loop and a Half

- Sometimes the termination condition can only be checked in the middle of a loop
 - Then, introduce a boolean variable to control the loop

```
boolean done = false;
while ( !done ) {
    System.out.print( "Enter value, Q to quit: " );
    String input = in.next();
    if ( input.equalsIgnoreCase( "Q" ) )
        done = true;
    else {
        double x = Double.parseDouble( input );
        data.add(x);
    }
}

System.out.println("Average = " + data.getAverage());
System.out.println("Maximum = " + data.getMaximum());
```

22

break Statement

- Used to break out of a switch statement
- Also used to exit (immediately) a while, for, or do loop
 - See Advanced Topic 7.4 (pg 258-259)

```
while ( true ) {
    System.out.print( "Enter value, Q to quit: " );
    String input = in.next();
    if ( input.equalsIgnoreCase( "Q" ) )
        break;
    double x = Double.parseDouble( input );
    data.add(x);
}
```

23

File Input/Output

- (Section 16.1)
- Two ways of storing data in files
 - *Text* format – human readable sequence of characters
 - Convenient for humans
 - *Binary* format – bytes of data
 - More compact and efficient
- We will use
 - Scanner class to read input from text files
 - PrintWriter class to write output to text files

24

Reading Text File

- First construct **FileReader** object with the name of the input file
- Then use it to construct a **Scanner** object
- Use the **Scanner** object for input just as if it was keyboard input
 - Use `next`, `nextLine`, `nextInt`, `nextDouble` methods

```
FileReader reader = new FileReader( "input.txt" );
Scanner in = new Scanner( reader );
```

- After done reading input, call the **close** method on the **FileReader** object

25

Writing Text File

- Construct a **PrintWriter** object with the name of the output file
 - Use `print`, `println`, `printf` methods

```
PrintWriter out = new PrintWriter( "output.txt" );
```

- Close the file when done
 - Otherwise not all output may be written to the file

```
out.close();
```

26

Skeleton Code for File Input/Output

```
// import necessary classes
import java.io.IOException;
import java.io.PrintWriter;
import java.io.FileReader;
import java.util.Scanner;

public class ... {

    public ... { // method

        try {
            // Do file input/output stuff here
            // ...
        } catch ( IOException exc ) {
            System.out.println( "Error processing file: " + exc );
        }
        // ...
    }
}
```

LineNumberer.java

27

Random Numbers and Simulation

- In a simulation, you repeatedly generate random numbers and use them to simulate an activity

```
Random generator = new Random();
int n = generator.nextInt(a); // 0 <= n < a
double x = generator.nextDouble(); // 0 <= x < 1
```

28

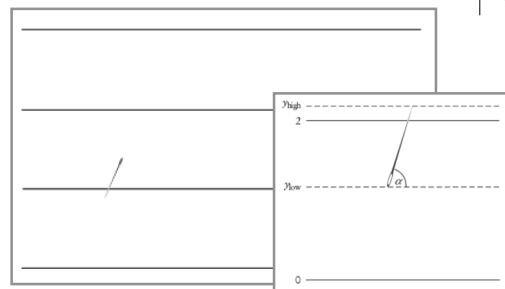
Random Numbers

- **Random** class (`java.util` package) provides a (pseudo)random number generator
 - Produces long sequences of non-repeating numbers that behave like a random sequence
- Two useful methods
 - **nextInt(n)** – returns 'random' *integer* between 0 (inclusive) and n (exclusive)
 - **nextDouble()** – returns 'random' *floating-point* number between 0.0 (inclusive) and 1.0 (exclusive)

- *Die.java*
- *DieTester.java*

29

Buffon Needle Experiment



30

Needle Position



- When does a needle fall on a line?
 - Needle length = 1in, distance between lines = 2in
- Generate random y_{low} between 0 and 2
- Generate random angle α between 0 and 180 degrees
- $y_{high} = y_{low} + \sin(\alpha)$
- Hit if $y_{high} \geq 2$

[Needle.java](#)

[NeedleTester.java](#)

31