

CIS 110: Introduction to Computer Programming

Lecture 13 Indefinite Loops (§ 5.1-5.2)

10/26/2011

CIS 110 (11fa) - University of Pennsylvania

1

Outline

- Indefinite loops with while
- Fencepost and sentinel loops

10/26/2011

CIS 110 (11fa) - University of Pennsylvania

2

Indefinite Loops

10/26/2011

CIS 110 (11fa) - University of Pennsylvania

3

Indefinite Loop Bounds

- So far we've known the bounds of our loops *before we've executed the loop themselves.*

– e.g., `for (int i = 0; i < 10; i++) { /* ... */ }`

- Many loops don't offer that luxury...

```
// while the user hasn't input "yes" yet
//   Ask the user for input
```

10/26/2011

CIS 110 (11fa) - University of Pennsylvania

4

Problem: firstDivisor

- Problem: write a method `firstDivisor(x, y)` that returns the first number that divides `x` starting at `y` and going up.
 - Example: `firstDivisor(26, 10) = 13`
 - Indefinite behavior: the amount of numbers we'll check depends on `x` and `y`.

10/26/2011

CIS 110 (11fa) - University of Pennsylvania

5

firstDivisor Solution

```
public static int firstDivisor(int x, int y) {
    while (x % y != 0) {
        y += 1;
    }
    return y;
}

public static void main(String[] args) {
    // Output: 13
    System.out.println(firstDivisor(26, 10));
}
```

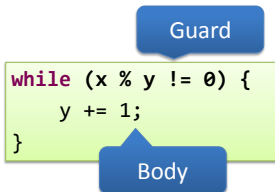
10/26/2011

CIS 110 (11fa) - University of Pennsylvania

6

While loops

- "While the guard is true, execute the body".
 - Like an if-statement, but looping!



10/26/2011

CS 110 (11fa) - University of Pennsylvania

7

While Loops vs. For Loops

```
for (int i = 0; i < 10; i++) {
    System.out.println(i);
}
```

```
int i = 0;
while (i < 10) {
    System.out.println(i);
    i++;
}
```

- Can express the same kinds of loops.
- Some benefit to **for** over **while** (i.e., scoping).
- **for** is meant for *definite loops*: "loop x times".
- **while** is meant for *indefinite loops*: "loop until some condition is met".

10/26/2011

CS 110 (11fa) - University of Pennsylvania

8

The Random Object

```
Random rand = new Random();
double value = 0;
while(value <= 0.5) {
    System.out.printf("%.2f is less than or equal to 5.\n", value);
    // nextDouble returns a double between 0.0 and 1.0
    value = rand.nextDouble();
}
System.out.printf("%.2f is greater than 5!\n", value);
```

- Random objects to generate (pseudo)-random numbers
 - "Pseudo"-random because they are still the result of mathematical formula

10/26/2011

CS 110 (11fa) - University of Pennsylvania

9

Method calls of the Random object

```
Random rand = new Random();
// Prints a random integer between -2^31 to (2^31)-1
System.out.println(rand.nextInt());

// Prints a random integer between 0 and 9
System.out.println(rand.nextInt(10));

// Prints out a random double starting at 0.0 up to
// (but not including) 1.0
System.out.println(rand.nextDouble());

// Prints either true or false randomly
System.out.println(rand.nextBoolean());
```

10/26/2011

CS 110 (11fa) - University of Pennsylvania

10

Simulations and Games

- Application of indefinite loops.
 - Repeatedly executes until some condition is met.
 - E.g., simulating a *random walk*.

```
Random rand = new Random();
int position = 1;
while (position > 0) {
    System.out.println("I am currently at " + position);
    if (rand.nextBoolean()) {
        position += 1;
    } else {
        position -= 1;
    }
}
System.out.println("I am back home!");
```

10/26/2011

CS 110 (11fa) - University of Pennsylvania

11

Fencepost and Sentinel Loops

10/26/2011

CS 110 (11fa) - University of Pennsylvania

12

The fencepost problem

- Problem: write a method `fencepost(n)` that takes an integer and draws a fencepost of length `n`.
 - e.g., `fencepost(5)` prints `|=|=|=|=|`

10/26/2011

CIS 110 (11fa) - University of Pennsylvania

13

Fencepost solution?

```
public static void fencepost(int n) {
    for (int i = 0; i < n; i++) {
        System.out.print("|=");
    }
    System.out.println();
}
```

- Not good enough!
 - Prints out an extra wire, e.g., `|=|=|=|=|`

10/26/2011

CIS 110 (11fa) - University of Pennsylvania

14

Hoisting is the solution!

```
public static void fencepost(int n) {
    System.out.print("|");
    for (int i = 1; i < n; i++) {
        System.out.print("=");
    }
    System.out.println();
}
```

- We *hoisted* part of the first iteration of the loop (i.e., the first post) and flipped the body.
 - Now the pattern works!
- *Loop-and-a-half* is a common pattern!

10/26/2011

CIS 110 (11fa) - University of Pennsylvania

15

Sentinels

- *Sentinels* are values that designate when a loop should end.
- Problem: write a loop that sums up positive integers from the user until they enter -1 to end the process.
 - -1 is the *sentinel value* in this loop.

```
// while the user's input isn't -1
// get an input from the user and add it to our running sum.
```

10/26/2011

CIS 110 (11fa) - University of Pennsylvania

16

Sentinel solution?

```
Scanner in = new Scanner(System.in);
int sum = 0;
int input = 0; // Prime loop so we enter it initially.
while (input != -1) {
    System.out.print("num? ");
    input = in.nextInt();
    sum += input;
}
System.out.println("sum = " + sum);
```

- Not good enough!
 - Prints out one less than the sum? Why?

10/26/2011

CIS 110 (11fa) - University of Pennsylvania

17

Solution: hoist out some input!

```
Scanner in = new Scanner(System.in);
int sum = 0;
// Hoist out half of the loop!
System.out.print("num? ");
int input = in.nextInt();
while (input != -1) {
    sum += input;
    System.out.print("num? ");
    input = in.nextInt();
}
System.out.println("sum = " + sum);
```

- Now it works!
 - We hoisted out one prompt out of the loop and changed the order of summation and prompting.

10/26/2011

CIS 110 (11fa) - University of Pennsylvania

18