ICS 111
Java While and Do...While Loops
Problem-Solving Techniques

- while loops
- infinite loops
- do ... while loops (do loops)
- off-by-one errors
- hand-tracing loops

Review: Repetitions/Loops

- Now that we've mastered conditionals, it is time to look at loops (repetitions)
- The instructions on the shampoo bottle say “lather, rinse, repeat”:
  - this repeats a sequence of two operations
  - in plain English, this repeats the sequence twice
  - to a computer, this may repeat forever!
- We may repeat a single basic statement, or a sequence, a sequence with conditionals, or any other combination of statements

How long do we keep repeating?

- When repeating a set of statements, one essential question is: how long do we keep repeating?
- If we are cooking pancakes, we want to make the right number of pancakes
  - this number may depend on the number of people at breakfast
  - or may depend on the amount of pancake batter we have
- One kind of loop tests a boolean condition to see whether to continue the loop
- This is called a **while** loop

While Loop Example

```
double batterLeft = 60.0;  // ounces
long hungryPeople = 5;
final double BATTER_PER_PANCAKE = 11.0;
while ((batterLeft >= BATTER_PER_PANCAKE) &&
       (hungryPeople > 0)) {
  make a pancake // can a computer do this?
  batterLeft = batterLeft - BATTER_PER_PANCAKE;
  hungryPeople = hungryPeople - 1;
}
```

While Loops

- The condition of a while loop tells us how long to keep going
  - it is an error to have the condition tell us when the loop should stop!
- The body of the loop usually changes the condition, in such a way that the loop eventually ends
  - unless you want an infinite loop!
-

Infinite Loop

```
while (true) {
    System.out.println (“this is an ∞ loop!”);
}
```

- It is easy to write infinite loops
- It is not always easy to guarantee that our loops will terminate!

Infinite Loop Self-Exercise

- Why is this program an infinite (or near infinite) loop?

```
double batterLeft = 60.0;  // ounces
final double BATTER_PER_PANCAKE = 11.0;
while (batterLeft >= BATTER_PER_PANCAKE) {
  make a pancake // can a computer do this?
  batterLeft = batterLeft + BATTER_PER_PANCAKE;
}
```

Infinite Loop Self-Exercise Answer

- Why is this program an infinite (or near infinite) loop?

```
double batterLeft = 60.0;  // ounces
final double BATTER_PER_PANCAKE = 11.0;
while (batterLeft >= BATTER_PER_PANCAKE) {
  make a pancake // can a computer do this?
  batterLeft = batterLeft + BATTER_PER_PANCAKE;
}
```

- Because batterLeft goes from 60, to 71, to 82, and onwards and upwards for a very long time -- batterLeft never gets below 11 ounces, so the loop never ends

Syntax of While Loops

1. `while`
2. the condition in parentheses
3. the statement(s) to be repeatedly executed
   - always in { braces } if there is more than one statement
   - always in { braces } if you are writing code for ICS 111

```
while (I am in ICS 111) {
    I will put braces around the body of my while statements (and if statements too!)
}
```

- This syntax parallels the syntax of `if` statements
  - but `while` loops have no `else`, no `else if`

`do ... while` Loops (`do` Loops)

- While loops test the condition before ever executing the loop
- What if you wanted to make sure the code in the loop is executed at least once?
  - you could have a complicated condition such as `while (firstLoop || ...`
  - or you could use a `do ... while` loop!

```
do {
  System.out.println (“hello world”);
} while (false);
```

prints hello world exactly once

- The condition, which in this example is always false, is evaluated after the loop has been executed the first time

Applications of Do Loops

- Do loops are useful when the condition to be tested is only valid after the first execution of the loop body

Examples of Do Loops

- something we might do for breakfast:

```
do {
  cook pancake
  eat pancake
} while (hungry);
```

```
int x = 1;
do {
  x = x * 2;
} while (x < 1000);
```

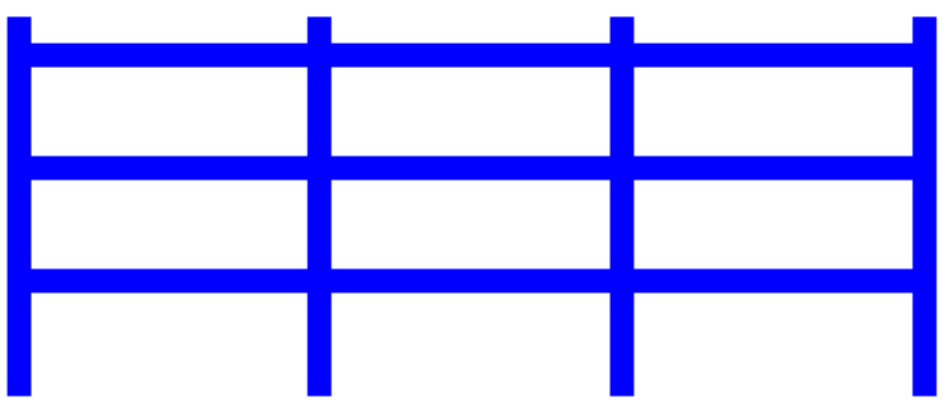
- after 10 loops, x has the value 1,024

Syntax of Do Loops

- `do`
- open brace `{` --- required in do loops
- body of the loop
- closing brace `}` --- also required
- `while`
- condition in parentheses
- semicolon `;`

Off-By-One Errors

- If you have a 20-foot fence, and you want to put a post every 2 feet, how many posts do you need?



Off-By-One Errors

- If you have a 20-foot fence, and you want to put a post every 2 feet, how many fenceposts do you need?
- The intuitive answer is 10 fenceposts
- The correct answer is 11 fenceposts:
  - one post each to the left and right of the first two-foot section
  - one more post for each additional two-foot section
- If you answered 10, you are off by one
  - this is also known as a fencepost error
  - even when no fencing is involved!

Example of Off-By-One Error

```
final long MAX_CONTENTS = ...
long contents = 0;
while (contents <= MAX_CONTENTS) {
  ...
  contents = contents + 1;
}
```

- At the end of the loop, the contents will be MAX_CONTENTS + 1
  - which is too much!
- In this case, we should have used < instead of <=

More Examples

- Assuming that the first index in a string is 1, instead of 0
  - in String.substring or String.charAt
- Assuming that the last valid index in a string is String.length()
  - the last valid index is String.length() - 1

```
int x = 0;
while (x < 10) {
  statements
  x = x + 1;
}
```

- How many times do we execute these statements?

Answers to the Last Example

```
int x = 0;
while (x < 10) {
  statements
  x = x + 1;
}
```

- How many times do we execute these statements?
- Answer: 10 times
  - with x being 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9
- but if the condition is x <= 10, then 11
- unless we initialize x = 1, then it is 10 times
- if we initialize x = 1, and the condition is x < 10, we execute the statements 9 times

- If you are confused, try making the condition x < 2, or x <= 2

Hand Tracing of Loops

```
int x = 0;
while (x < 3) {
  statements
  x = x + 1;
}
```

- start: x = 0
- condition is true, so enter the loop
- execute statements with x being 0
- now: x = 0 + 1 = 1 ~~0~~
- 1 < 3, so the condition is true
- enter the loop, execute statements with x being 1
- now: x = 1 + 1 = 2 ~~1 0~~
- 2 < 3, so the condition is true
- enter the loop, execute statements with x being 2
- now: x = 2 + 1 = 3 ~~2 1 0~~
- x not less than 3, so the condition is false, the loop ends with x being 3

Summary

- While loops and Do (Do/While) loops execute as long as the condition is true
  - while loops execute 0 or more times, do loops execute 1 or more times
  - while loops are much more common than do loops
- If the condition never becomes false, the loop is an infinite loop
- A subtler error is to be off by one
  - can have one too many, or one too few
- Hand tracing can help us understand what loops are doing
  - as long as the number of loops is small