Branching & Loops
COMP 110: Lecture 9
Assignment 2

Assignment 2 is on the website.

Getting started demo…
Assignment 2

Due Friday, May 27

Get started today.
It will be good practice for the midterm.
Finish Lecture 8

On Friday we skipped the last parts of 8 to preview Assignment 2 (did I mention it was posted?)

Let’s finish it up…
Guessing Game

In order to get the GuessingGame to keep ask enough times we put in 7 guess() calls.

Why 7?

That's how many times we can halve 100 before we get 1.

$ java GuessingGame
Shall we play a game? [y/n] y
Is it 50? [yes/higher/lower] higher
Is it 75? [yes/higher/lower] higher
Is it 88? [yes/higher/lower] higher
Is it 94? [yes/higher/lower] higher
Is it 97? [yes/higher/lower] higher
Is it 99? [yes/higher/lower] higher
It is 100
Game over

$ log_2(100) \approx 6.6439
2^{6.6439} \approx 100
Guessing Game

What if the user typed “nope” instead of “higher/lower”?

What if we wanted to play guess a number between 1 and 1,000,000?
(You’d need 20 guesses btw)
Guessing Game

What we really want is something like this.

This is called a loop.
the while loop

while (condition) {
    statements to execute;
}

Pattern look familiar?

if (condition) {
    statements to execute
}
the while loop

while (condition) {
    statements to execute;
}

While condition is true, execute these statements.
the while loop

while (condition) {
    statements to execute;
}

CPU:
1. check condition, if it is true, proceed to step 2
2. execute statements
3. check condition, if it is true, proceed to step 4
4. execute statements
5. check condition, if it is true, proceed to step 6
6. execute statements
7. ...
class LoopForever {
    public static void main(String[] args) {
        while (true) {
            System.out.println("Hello, World!");
        }
    }
}

How do you stop it?

CTRL + C
If your program gets caught in a loop.
Remember it now.
void play() {
    String answer = _console.readLine("Shall we play a game? [y/n] ");
    if (!answer.equals("y")) {
        return;
    }
    if (guess()) {
        return;
    }
    if (guess()) {
        return;
    }
    if (guess()) {
        return;
    }
    if (guess()) {
        return;
    }
    if (guess()) {
        return;
    }
    if (guess()) {
        return;
    }
    while (true) {
        guess();
    }
}

Almost… we’ll need CTRL-C here.
void play() {
    String answer = _console.readLine("Shall we play a game? [y/n] ");
    if (!answer.equals("y")) {
        return;
    }

    boolean gotIt = false;
    while (!gotIt) {
        gotIt = guess();
    }
}
Guessing Game: CPU

```java
boolean gotIt = false;
while (!gotIt) {
    gotIt = guess();
}
```

Variable `gotIt` is created, and set to `false`
Guessing Game: CPU

```java
boolean gotIt = false;
while (!gotIt) {
    gotIt = guess();
}
```

Since `gotIt` is false, `(!gotIt)` is true, and the CPU enters body of while loop
Guessing Game: CPU

boolean gotIt = false;
while (!gotIt) {
    gotIt = guess();
}

In `guess()`, the user types “higher”, so `guess()` returns `false`, and `gotIt` is thus assigned to `false`
Guessing Game: CPU

```java
boolean gotIt = false;
while (!gotIt) {
    gotIt = guess();
}
```

The end of the while loop is reached, so CPU jumps back to the beginning of the loop to check the condition again.
Guessing Game: CPU

```java
boolean gotIt = false;
while (!gotIt) {
    gotIt = guess();
}
```

Since `gotIt` is (still) false, `(!gotIt)` is true, and the CPU enters body of while loop.
Guessing Game: CPU

```java
boolean gotIt = false;
while (!gotIt) {
    gotIt = guess();
}
```

In `guess()`, the user types “yes”, so `guess()` returns `true`, and `gotIt` is thus assigned to `true`
Guessing Game: CPU

boolean gotIt = false;
while (!gotIt) {
    gotIt = guess();
}

The end of the while loop is reached, so CPU jumps back to the beginning of the loop to check the condition again.
Guessing Game: CPU

```java
boolean gotIt = false;
while (!gotIt) {
    gotIt = guess();
}
```

Since `gotIt` is true, `(!gotIt)` is false, and the CPU leaves the loop.
Guessing Game: CPU

```java
boolean gotIt = false;
while (!gotIt) {
    gotIt = guess();
}
```

The CPU runs the next line after the loop.
Loop $n$ times

Suppose you wanted to loop 3 times...

class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}

Formula:
1. Initialize an iteration variable to 0
2. Loop while variable $< n$ (the limit)
3. Increment variable each loop
Loop 3 times

```java
class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}
```

Initialize variable `i` to 0.
Loop 3 times

```java
class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}
```
Loop 3 times

class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}

Prints out the string “Hello”
class Counting {
public static void main(String[] args) {
    int i = 0;
    while (i < 3) {
        System.out.println("Hello");
        i = i + 1;
    }
}
}

Loop 3 times

Increments i, now i is 1.
class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}

Loop 3 times

End of loop, CPU jumps back to while condition

i

Output
Hello

1
Loop 3 times

class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}

Since \(i\) is 1, \(i < 3\), and CPU enters loop
Loop 3 times

class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}

Prints out the string “Hello” (again)
class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}
Loop 3 times

class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}

End of loop, CPU jumps back to while condition
Loop 3 times

```java
class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}
```

Since i is 2, i < 3, and CPU enters loop
Loop 3 times

class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}

Prints out the string “Hello” (again)
class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}

Increments i, now i is 3.
class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}

End of loop, CPU jumps back to while condition
Loop 3 times

class Counting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello");
            i = i + 1;
        }
    }
}

Since i is 3, i is not less than 3, and CPU leaves loop

Done!
class GenerateTable {
    public static void main(String[] args) {
        int i = 0;
        while (i < 20) {
            int div = i / 3;
            int mod = i % 3;
            System.out.println(i+"/3="+div+", "+i+"%3="+mod);
            ++i;
        }
    }
}
for Loops

The previous pattern of (1. init, 2. check, 3. increment) is so common, that we have another loop construct for it: the for loop.

The previous example is as a for-loop is:

```java
for (int i=0 ; i<3 ; ++i) {
    System.out.println("Hello");
}
```
for Loops

1. **Initialization**: before first iteration.
   (variables declared here are local to the loop)
2. **Check**: run before each iteration, loop stops when false
3. **Increment**, run after each iteration

```cpp
for ( initialization ; check ; increment ) {
    statements to execute each iteration
}
```
for Loops

1. **Initialization**: before first iteration.
   (variables declared here are local to the loop)
2. **Check**: run before each iteration, loop stops when false
3. **Increment**, run after each iteration

```java
for (int i=0 ; i<3 ; ++i) {
    System.out.println("Hello");
}
```
for Loops

```cpp
for ( initialization ; check ; increment ) {
    statements to execute each iteration
}
```

Initialization can be:
a variable declaration, an expression, or empty.

Check can be:
an expression, or empty (which means true)

Increment can be:
an expression, or empty.

*Initialization and increment here are used generically.*

*They can be any expression.*
class WhileCounting {
    public static void main(String[] args) {
        int i = 0;
        while (i < 3) {
            System.out.println("Hello: while "+i);
            i = i + 1;
        }
    }
}

class ForCounting {
    public static void main(String[] args) {
        for (int i=0 ; i<3 ; ++i) {
            System.out.println("Hello: for "+i);
        }
    }
}
Nesting Loops

Generate a text checkerboard

<table>
<thead>
<tr>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
<td>#</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
<tr>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
<td>B</td>
</tr>
</tbody>
</table>
class Checkers {
    public static void main(String[] args) {
        for (int i=0 ; i<8 ; ++i) {
            for (int j=0 ; j<8 ; ++j) {
                if ((i+j)%2 == 0) {
                    System.out.print("| ");
                } else if (i < 3) {
                    System.out.print("|R");
                } else if (i > 4) {
                    System.out.print("|B");
                } else {
                    System.out.print("|#");
                }
            }
            System.out.println("|");
        }
    }
}
Reading

Chapter 7