Loops

- Within a method, we can alter the flow of control using either conditionals or loops.
- The loop statements while, do-while, and for allow us execute a statement(s) over and over.
- Like a conditional, a loop is controlled by a boolean expression that determines how many times the statement is executed.

E.g., You may want to calculate the interest paid on a mortgage for each year of the loan term.

The while statement

- The form of the while statement is
  
  \[
  \text{while (<boolean_expression>)} \\
  \text{<statement>}
  \]
- If boolean_expression evaluates to true, then statement is executed.
- Then, the boolean_expression is evaluated again. If it evaluates to true, statement is executed again.
- This repetition continues until the boolean_expression evaluates to false.

How is the while loop different from the if statement?
The while Flowchart

```
while boolean_expression {
  statement
}
```

### A while Example

**Print n asterisks**

```java
test
```

```java
int i = 0;
while (i < n) {
  System.out.print("*");
  i ++;
}
System.out.println();
```

For n = 5 the output is:

```
*****
```

### The Loop Control Variable

- The variable `i` (known as the **loop control variable**) is used in three ways: it is **initialized**, **tested**, and **updated**.

```java
// initialize
int i = 0;
while (i < n) {
  // test
  System.out.print("*");
  i ++;
  // update
}
System.out.println();
```

- All three things must be coordinated in order for the loop to work correctly!

### Off-by-1 Errors

```java
// initialize
int i = 1;
while (i < n) {
  System.out.print("*");
  i ++;
}
System.out.println();
```

Output?
### Off-by-1 Errors

```java
int i = 0;
while (i < n) {
    System.out.print("*");
    i++;
} System.out.println();
```

For `n = 5` the output is

```
*****
```

### Warning!

```java
int i = 0;
while (i <= n) {
    System.out.print("*");
    i++;
} System.out.println();
```

What is the output if `n = 5`?

### Infinite Loops

```java
int i = 0;
while (i < n) {
    System.out.print("*");
    i--;
} System.out.println();
```

Do you know which company has this address?

1 Infinite Loop
Cupertino, CA 95014

### A While Example

```java
int i = 0;
while (i < n) {
    System.out.print("*");
    i++;
} System.out.println();
```

What is the output if `n = 0`?
Exercise

• Write a method with a while loop to prints 1 through n in square brackets. For example, if n = 6 print

Exercise: Cumulative Sum

• Write a method with a while loop that computes the sum of first n positive integers:
  \[ \text{sum} = 1 + 2 + 3 + \ldots + n \]

Examples:
  n = 5    sum = 15
  n = 19   sum = 190

Exercise: Fencepost Loop

• Write a method with a while loop that prints 1 through n, separated by commas. E.g., for n = 9 print
  1, 2, 3, 4, 5, 6, 7, 8, 9

The do Statement

• The form of the do statement is
  \[
  \text{do} \\
  \quad <\text{statement}> \\
  \text{while} (\text{<boolean_expression>});
  \]

• First, statement is executed.
• Then, the boolean_expression is evaluated. If it evaluates to true, statement is executed again.
• This repetition continues until the boolean_expression evaluates to false.
The do Flowchart

- **Boolean Expression**
  - **Statement**
  - **True**
  - **False**

**Example**

```java
int i = 0;  // initialize
do {
    System.out.print("*");
i++;
} while (i < n);  // test
System.out.println();
```

For `n = 7` what is the output?
How is it different from the while loop?

**User Input**

```java
Scanner keyboard = new Scanner(System.in);
System.out.print("Please enter the month [1-12]: ");
int month = keyboard.nextInt();
```

What if the user enters a month outside the range?

**User Input (cont’d)**

- Use a do-while loop to test whether a user has entered data of the correct form and, if not, ask repeatedly until the data entered is correct.

```java
Scanner keyboard = new Scanner(System.in);
int month;  // Must be declared outside the loop
do {
    System.out.print("Please enter the month [1-12]: ");
    month = keyboard.nextInt();
} while (month < 1 || month > 12);
```
User Input

- Sometimes it is easier to think of what you want the input to be and negate.

Scanner keyboard = new Scanner(System.in);
int month;
do {
    System.out.print("Please enter the month [1-12]: ");
    month = keyboard.nextInt();
} while (!{(month >= 1 && month <= 12)});

Use de Morgan’s law to prove the Boolean expressions are the same!

Sentinel Controlled Loops

- Suppose you want to find the maximum of the data entered from the keyboard.
- It is not known in advance how many data values a user might want to enter. (And the user may not want to count them!)
- A sentinel is a special value that is used to detect a special condition, in this case that the user is done entering values.
- The sentinel, of course, must be distinct from any value the user may want to input.

Scanner console = new Scanner(System.in);
System.out.print("Enter count (enter -1 to quit): ");int count = console.nextInt();int maxSoFar = count;
while (count != -1) {
    if (count > maxSoFar) maxSoFar = count;
    System.out.print("Enter count (enter -1 to quit): ");
    count = console.nextInt();
}
if (maxSoFar > -1)
    System.out.println("The maximum is " + maxSoFar);
else
    System.out.println("No counts entered");

Sentinel Example

Consider making -1 a named constant

Scanner keyboard = new Scanner(System.in);
int month;
Scanner console = new Scanner(System.in);
System.out.print("Enter count (enter -1 to quit): ");int count = console.nextInt();int maxSoFar = count;
while (count != -1) {
    if (count > maxSoFar) maxSoFar = count;
    System.out.print("Enter count (enter -1 to quit): ");
    count = console.nextInt();
}
if (maxSoFar > -1)
    System.out.println("The maximum is " + maxSoFar);
else
    System.out.println("No counts entered");