Flow Control

Loops

Loops

- Java provides three mechanisms to repeat code in a loop.
  - while
  - do-while
  - for
- Loops require boolean conditions to control how many times the loop repeats.
  - `if` statements are not loops (there's no such thing as an "`if` loop").

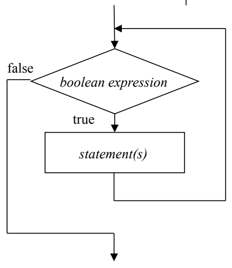
The `while` loop & flowchart

- Syntax:

```
while ( boolean_expression )
  statement ;
```

OR

```
while ( boolean_expression )
{
  statement_list
}
```



Example

- Compute the sum 1 + 2 + 3 + ... + n.

```
int sum = 0;
int i = 1;
while (i <= n)
{
 sum += i;
 i++;
}
System.out.println(sum);
```

n = 7

sum	i
0	1
1	2
3	3
6	4
10	5
15	6
21	7
28	8

Example

- The order of statements is important!!!

```
int sum = 0;
int i = 1;
while (i <= n)
{
 sum += i;
 i++;
}
System.out.println(sum);
```

If n = 7, output = 28

```
int sum = 0;
int i = 1;
while (i <= n)
{
      i++;
      sum += i;
}
System.out.println(sum);
```

If n = 7, output = ____

Off-By-1 Errors

```
int sum = 0;
int i = 1;
while (i < n)
{
     sum += i;
     i++;
}
System.out.println(sum);
```

It is very easy to have a loop run one too few or one too many times, so think carefully as you determine the loop condition.

Watch Out!

```
int sum = 0;
int i = 1;
while (i <= n)
    sum += i;
    i++;
System.out.println(sum);
```

MISSING BRACKETS!

What is the output of this code fragment?

Watch Out Again!

```
int sum = 0;
int i = 1;
while (i <= n);
{
    sum += i;
    i++;
}
System.out.println(sum);
```

BAD SEMICOLON!

What is the output of this code fragment?

Infinite Loops

- Infinite loops are generally bad (except in a few programming domains like GUIs), so watch out!
- Where is this "good" infinite loop?

Programming Exercise

- Write a simple Java program that prompts the user for a value of n $\geq$ 0 and computes the sum of the integers in this sequence using a `while` loop:
  sum = 1 + 2 + 4 + 8 + 16 + ... + 2^n

Examples:

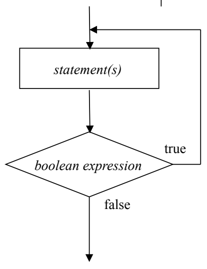
n = 5	sum = 63
n = 10	sum = 2047

FUN FACT!
Actually, this sum can be computed without a loop as sum = 2^{n+1} - 1.

The do-`while` loop & flowchart

- Syntax:

```
do
{
  statement_list
} while ( boolean_expression );
```



Example

```
int sum = 0;
int i = 1;
do
{
    sum += i;
    i++;
} while (i <= n);
System.out.println(sum);
```

How is this loop different than a while loop?

User Input

```
Scanner scan = new Scanner(System.in);
System.out.println(
  "Please input liquid water temperature "
      + "in degrees Fahrenheit: ");
double temperature = scan.nextDouble();
```

What happens if the user inputs an invalid temperature for water in a liquid state?

(For this problem, we will assume water is a liquid between 32º and 212º F, exclusive.)

User Input (cont'd)

```
Scanner scan = new Scanner(System.in);
double temperature;
do {
  System.out.println(
      "Please input liquid water temperature "
      + "in degrees Fahrenheit: ");
  temperature = scan.nextDouble();
} while (temperature [< <= > >= == !=] 32.0 [&& ||]

         temperature [< <= > >= == !=] 212.0);
```

Using DeMorgan's Law

```
Scanner scan = new Scanner(System.in);
double temperature;
do {
  System.out.println(
      "Please input liquid water temperature "
      + "in degrees Fahrenheit: ");
  temperature = scan.nextDouble();

} while (________________________________);
```

!(A || B) = !A && !B

A || B = !(!A && !B)

Programming Exercise

- Using a `do-while` loop, modify the program you wrote to compute the sum of 1+2+4+8+... +2^n , n $\geq$ 0, so that if a user inputs an invalid integer value of n, the program prompts the user again until the user finally inputs a valid integer value.

The `for` loop

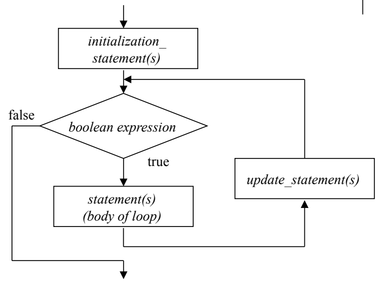
- Syntax:

for (*initialization_statement(s)* **;**
boolean_expression **;**
update_statement(s) **)**

{
statement_list
}

If the body of the loop contains only one statement, the brackets can be omitted. But be careful!

The `for` loop flowchart



Example

```
int sum = 0;
for (int i = 1; i <= n; i++)
{
    sum += i;
}
System.out.println(sum);
```

We say that this region of the program is the **scope** of the variable `i`.

NOTE: In this example, the variable `i` is defined locally for the loop only. It is not accessible outside of the loop.

Example

```
int i;
int sum = 0;
for (i = 1; i <= n; i++)
{
    sum += i;
}
System.out.println(
    "i = " + i +
    " and sum = " + sum);
```

scope of `i`

Finding the Maximum

1. Assume the first data value is the maximum.
2. Look at the next data value and compare it to the current maximum. If it is greater than the current maximum, set this value as the maximum.
3. Repeat step 2 for all remaining data values one at a time.

Finding the mininum uses a similar algorithm.

Computing Figure Skating Score

(old Olympics algorithm)

1. There are 9 judges.
2. Each judge gives the skater a score between 0.0 (unbelievably horrible) to 6.0 (perfect).
3. The highest score from a judge is thrown out.
4. The lowest score from a judge is thrown out.
5. The skaters score is computed as the average of the remaining 7 scores.

Using Sentinels

- Compute the average for an exam based on scores entered at the keyboard.
  - Valid scores are in the range [0,100]
- We don't know how many scores there will be ahead of time.
- How do we know when to stop reading input from the keyboard?
- Use a sentinel: a invalid value that signals that there is no more input (e.g. a score of -1).

Sentinel Example

```
Scanner scan = new Scanner(System.in);
int sum = 0;
int count = 0;
System.out.println("Input next score [-1 to exit]:");
int score = scan.nextInt();
while (score != -1)
{
   if (score < -1 || score > 100)
       System.out.println("Invalid Score");
   // continued on next slide
```

Sentinel Example (cont'd)

```
    else {
        sum += score;
        count++;
    }
    System.out.println("Input next score " +
          "[-1 to exit]:");
    score = scan.nextInt();
}
if (count != 0)
  System.out.println((double)sum/count);
else
  System.out.println("No exam average");
```

25

Nested Loops

- A loop can have another loop inside of it.
- **The inside loop runs completely for each iteration of the outside loop.**
- Example:
  How many times is JAVA printed out?

```
for (int i = 1; i <= 7; i++)
    for (int j = 1; j <= 7; j++)
        System.out.println("JAVA");
```

26

Nested Loops

- Example:
  How many times is JAVA printed out?

```
for (int i = 1; i <= 7; i++)
    for (int j = i; j <= 7; j++)
        System.out.println("JAVA");
```

- When i = 1, the inside loop runs 7 times.
- When i = 2, the inside loop runs another 6 times.
- When i = 3, the inside loop runs another 5 times...

27

Nested Loops

- For help understanding these loops, try this:

```
for (int i = 1; i <= 7; i++)
    for (int j = i; j <= 7; j++)
        System.out.println(i + " " + j);
```

- NOTE: Don't write `println(i + j)` since this will cause the sum of `i` and `j` to be printed, not the individual values of `i` and `j`!

28

Nested Loops Example

- What shape does this print out?

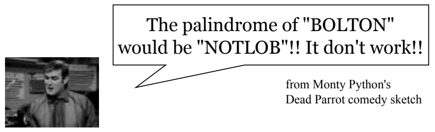
```
for (int i = 1; i <= 7; i++)
{
    for (int j = 1; j <= i; j++)
         System.out.print("*");
    System.out.println();
}
```

29

Palindromes

- A palindrome is a word that reads exactly the same forwards and backwards.

KAYAK	**NOON**
REDDER	**RACECAR**


The palindrome of "BOLTON" would be "NOTLOB"!! It don't work!!

from Monty Python's
Dead Parrot comedy sketch

30

Palindrome Algorithm

1. Start with the leftmost and rightmost characters.
2. Continue moving in toward the center of the word as long as
   - there are still more characters to compare
   - the current pair of characters you're comparing match each other
3. Once you are done with step 2, if you matched all pairs of characters successfully, you have a palindrome.

(See website for programming exercise.)

31

Comparing Loops

- **`for` loops are generally used for loops where we know exactly how many iterations we need.**
  - **The loop variable acts as a counter and can be used in the loop calculation as well**
- **while and do-while loops are generally used for loops that have a variable number of iterations unknown before runtime.**
  - **Use the `while` loop if the minimum number of iterations could be 0 (i.e. the loop might be skipped altogether).**
  - **Use the `do-while` loop in cases where the loop must execute at least once (for example, for user input validation).**

32