while and do-while Statements
Outline

- Introduction
- while Loop
- do-while Loop
- Introduction

- Loops in Java are similar to those in other high-level languages

- Java has three types of loop statements:
  - The while
  - The do-while
  - The for

- The code that is repeated in a loop is called the body of the loop

- Each repetition of the loop body is called an iteration of the loop
A **while** statement is used to repeat a portion of code (i.e., the loop body) based on the evaluation of a Boolean expression

- The Boolean expression is checked *before* the loop body is executed
  - When false, the loop body is not executed at all
- Before the execution of each following iteration of the loop body, the Boolean expression is checked again
  - If true, the loop body is executed again
  - If false, the loop statement ends
- The loop body can consist of a single statement, or multiple statements enclosed in a pair of braces (`{   }`)
-- while Loop Syntax

while ( <boolean expression> )

<statement>  //only one statement

OR

while ( <boolean expression> ) {

<statement>  //many

}

These statements are executed as long as number is less than or equal to 100.

while ( number <= 100 ) {

sum = sum + number;

number = number + 1;

}

These statements are executed as long as number is less than or equal to 100.
-- while Loop Control flow

```
int sum = 0, number = 1

while (number <= 100)
    if (number <= 100)
        sum = sum + number;
        number = number + 1;
    else
        break;
```
-do-while Loop

A do-while statement is used to execute a portion of code (i.e., the loop body), and then repeat it based on the evaluation of a Boolean expression.

- The loop body is executed at least once
  - The Boolean expression is checked after the loop body is executed

- The Boolean expression is checked after each iteration of the loop body
  - If true, the loop body is executed again
  - If false, the loop statement ends
  - Don’t forget to put a semicolon after the Boolean expression

- Like the while statement, the loop body can consist of a single statement, or multiple statements enclosed in a pair of braces ({   })
-- do-while Loop Syntax

do {
    <statement>
}
} while (<boolean expression>);

These statements are executed as long as sum is less than or equal to 1,000,000.
-- do-while Loop Control Flow

int sum = 0, number = 1

sum += number;
number++;

sum <= 1000000 ?

true

false
THE END
Examples
Questions

1. Write a Java program which computes the sum of all the odd numbers between 0 and 100.

2. Write a Java program which reads 20 numbers using a scanner and computes their average.

3. Write a Java program which reads unknown number of integers using a scanner and counts the number of odd numbers and the number of even numbers. Assume the input integers are all positive. Use a negative number as a sentinel.
Solution using while loop
Q1 Solution

Write a Java program which computes the sum of all the odd numbers between 0 and 100.

```java
int n = 1;
int sum = 0;
while (n < 100) {
    sum += n;
    n = n + 2;
}
System.out.println("The sum is " + sum);
```
Q2 Solution

Write a Java program which reads 20 numbers using a scanner and computes their average.

```java
Scanner kb = new Scanner(System.in);
int cnt = 0;
double x;
double sum = 0;
While (cnt < 20) {
    x = kb.nextDouble();
    sum += x;
    cnt++;
}
System.out.println("The Average is " + sum/cnt);
```
Q3 Solution

Write a Java program which reads unknown number of integers using a scanner and counts the number of odd numbers and the count of even numbers. Assume the input integers are all positive. Use any negative number as a sentinel.

```java
Scanner kb = new Scanner(System.in);
int even_cnt = 0;
int odd_cnt = 0;
double x = kb.nextInt();
while (x > 0) {
    if (mod(x, 2) == 0)
        even_cnt++;
    else
        odd_cnt++;
    x = kb.nextInt();
}
System.out.println("Even numbers are = " + even_count);
System.out.println("Odd numbers are = " + odd_count);
```
Solution using do-while loop
Q1 Solution

Write a Java program which computes the sum of all the odd numbers between 0 and 100.

```java
int n = 1;
int sum = 0;
do {
    sum += n;
    n = n + 2;
} While ( n < 100)
System.out.println(“The sum is “ + sum);
```
Write a Java program which reads 20 numbers using a scanner and computes their average.

```java
Scanner kb = new Scanner(System.in);
int cnt = 0;
double x;
double sum = 0;
do {
    System.out.println("Enter a number");
    x = kb.nextDouble();
    sum += x;
    cnt++;
} while (cnt < 20);
System.out.println("The Average is “ + sum/cnt);
```
Q3 Solution

Write a Java program which reads unknown number of integers using a scanner and counts the number of odd numbers and the count of even numbers. Assume the input integers are all positive. Use any negative number as a sentinel.

```java
Scanner kb = new Scanner(System.in);
int even_cnt = 0;
int odd_cnt = 0;
double x = kb.nextInt();
if (x > 0) {
    do {
        if ( mod(x, 2) == 0)
            even_cnt++;
        else
            odd_cnt++; 
        x = kb.nextInt();
    } while ( x > 0)
}
System.out.println("Even numbers are = "+ even_count);
System.out.println("Odd numbers are = "+ odd_count);
```
Additional Slides
**while Loop Pitfall - 1**

1. ```java
   int product = 0;
   while ( product < 500000 ) {
       product = product * 5;
   }
```

2. ```java
   int count = 1;
   while ( count != 10 ) {
       count = count + 2;
   }
```

**Infinite Loops**
Both loops will not terminate because the boolean expressions will never become false.
while Loop Pitfall - 2

1

\[\text{double count} = 0.0;\]

\[\text{while ( count != 1.0 ) { } \]
\[\text{count} = \text{count} + 1.0/3.0;\]
\[}\]

2

\[\text{double count} = 0.0;\]

\[\text{while ( count <= 1.0 ) { } \]
\[\text{count} = \text{count} + 1.0/3.0;\]
\[}\]

Using Real Numbers
Loop 2 terminates, but Loop 1 does not because only an approximation of a real number can be stored in a computer memory.
while Loop Pitfall - 3

Goal: Execute the loop body 10 times.

1. count = 1;
   while (count < 10) {
       ...
       count++;
   }

2. count = 1;
   while (count <= 10) {
       ...
       count++;
   }

3. count = 0;
   while (count <= 10) {
       ...
       count++;
   }

4. count = 0;
   while (count < 10) {
       ...
       count++;
   }

① and ③ exhibit off-by-one error.
Checklist for Repetition Control

1. Watch out for the off-by-one error (OBOE).
2. Make sure the loop body contains a statement that will eventually cause the loop to terminate.
3. Make sure the loop repeats exactly the correct number of times.