Outline

° Problem: How can I perform the same operation again and again based on a condition?

° Remembering conditional statements
  • Same as if statements

° The “while” statement
  • Keep performing operations while the condition is true

° Nested for loops
  • Determining how many times an operation takes place
Repetition Statements

- Repetition statements allow us to execute a statement multiple times
- Often they are referred to as loops
- Like conditional statements, they are controlled by boolean expressions
- Java has three kinds of repetition statements:
  - the while loop
  - the do loop
  - the for loop
- The programmer should choose the right kind of loop for the situation

The while Statement

- A while statement has the following syntax:

```java
while ( condition )
{
  statement;
}
```

- If the condition is true, the statement is executed
- Then the condition is evaluated again, and if it is still true, the statement is executed again
- The statement is executed repeatedly until the condition becomes false
Logic of a while Loop

Note: there is a ‘pretest.’
What does this mean??

The while Statement

- An example of a while statement:

  ```java
  int count = 1;
  while (count <= 5) {
      System.out.println (count);
      count++;
  }
  ```

- If the condition of a while loop is false initially, the statement is never executed
- Therefore, the body of a while loop will execute zero or more times
The while Statement

- Let's look at some examples of loop processing
- A loop can be used to maintain a *running sum*
- A *sentinel value* is a special input value that represents the end of input
- A loop can also be used for *input validation*, making a program more *robust*

Infinite Loops

- The body of a `while` loop eventually must make the condition false
- If not, it is called an *infinite loop*, which will execute until the user interrupts the program
- ➡ This is a common logical error
- Double check the logic of a program to ensure that your loops will terminate normally

Many funny stories as a result of infinite loops
Infinite Loops

- An example of an infinite loop:

  ```java
  int count = 1;
  while (count <= 25)
  {
    System.out.println (count);
    count = count - 1;
  }
  ```

- This loop will continue executing until interrupted (Control-C) or until an underflow error occurs.

Nested Loops

- Similar to nested if statements, loops can be nested as well
- That is, the body of a loop can contain another loop
- For each iteration of the outer loop, the inner loop iterates completely
Nested Loops

- How many times will the string "Here" be printed?

```
count1 = 1;
while (count1 <= 10)
{
    count2 = 1;
    while (count2 <= 20)
    {
        System.out.println("Here");
        count2++; // end inner while
    }
    count1++; // end outer while
}
```

The do Statement

- A do statement has the following syntax:

```
do
{
    statement;
}
while ( condition )
```

- The statement is executed once initially, and then the condition is evaluated
- The statement is executed repeatedly until the condition becomes false
Logic of a do Loop

The do Statement

- An example of a do loop:

```java
int count = 0;
do{
    count++;
    System.out.println (count);
} while (count < 5);
```

- The body of a do loop executes at least once

Note: no ‘pretest.’ Statement will at least be executed one time!
Comparing do-while and while

- Done using do-while

```c
numberOfDigits = 0;
rest = number;
do {
    rest = rest / 10;
    numberOfDigits++;
} while (rest != 0);
```

- Equivalent while loop

```c
numberOfDigits = 0;
rest = number;
if (number == 0)
    numberOfDigits = 1;
else
    while (rest > 0) {
        rest = rest / 10;
        numberOfDigits++;
    }
```

do can be written as while

- A do loop can be easily re-written as a while loop.

```
do
    statement;
while (condition);
```

turns into

```
statement;
while (condition)
    statement;
```
do can be written as while

- A do loop can be easily re-written as a while loop.

```
do
  statement;
while (condition);
```

These are exactly the same statement.

---

Comparing while and do

**The while Loop**

1. condition evaluated
   1. true
      - statement
   2. false

**The do Loop**

1. statement
   1. true
      - condition evaluated
         - false
         - statement
      - false
   2. false
Summary

° Loops form an essential part of programming
  • Allow for repetitive actions

° Pre-test loops (while) are only executed if the condition is true
  • Can lead to “infinite loops”

° Do-while loops allow the statements in the loop to execute at least once
  • Not used very often

° Next time: For loops