Pseudocode and various looping structures

Pseudocode is an artificial and informal language that helps programmers develop algorithms. Pseudocode is a "text-based" detail (algorithmic) design tool.

The rules of Pseudocode are reasonably straightforward. All statements showing "dependency" are to be indented. These include while, do, for, if, switch. Examples below will illustrate this notion.

Loops

1 We saw that the flow of control can be changed using conditional statements

1.1 Allow our program to choose which statement to execute next

2 Can also change flow of control using loops

2.1 Allow our program to execute a statement multiple times

Types of Loops

The 3 Java loops are:

1. while loops
2. for loops
3. do-while loops

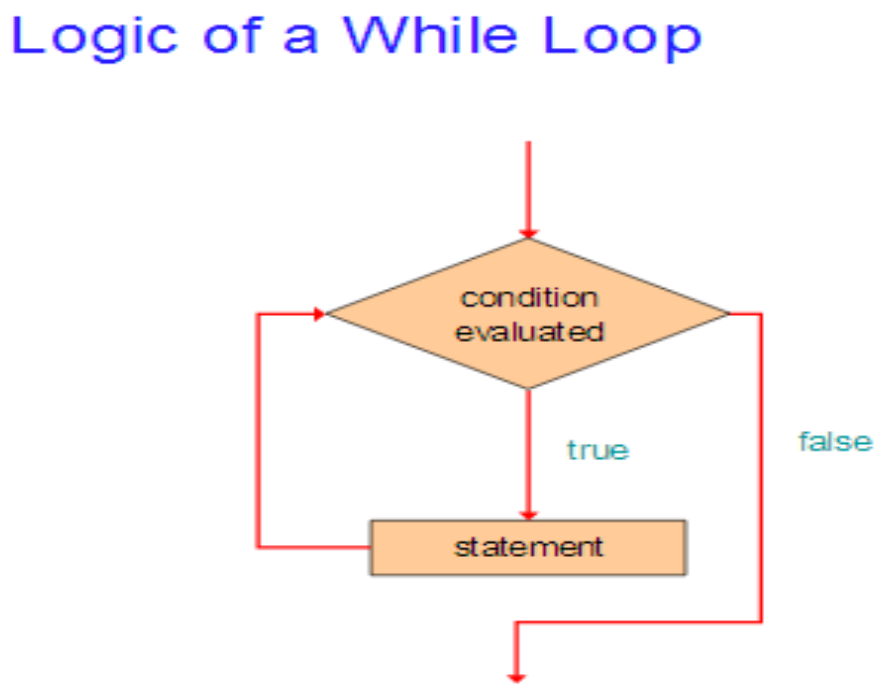
While Loop Pseudocode

the while loop implements the following pseudo code:
while some condition is true
do something

The While-Loop

Boolean expression

- If the condition is true, execute the statement. Repeat until condition is false.
- If condition is false, skip the statement and continue your program.
- Statement could be a block statement.
- If the condition is false initially, the statement is never executed.



Common Uses

- here are some common loop uses:
    1. input validation
    2. counter-controlled loops
    3. sentinel loops

- we'll discuss each of these in turn

Input Validation

- we have used if-statements to validate input — to check if an input value is valid
- EX:

```
double width;
width = Input.readDouble("Enter width: ");
if (width <= 0.0)
{
    Output.showMessage("NEGATIVE WIDTH");
    System.exit(0);
}
```

- but if an invalid value is entered, what we really want to do is give the user more chances to input a valid value

Input Validation Algorithm

- a while-loop is a better way to validate input
- input validation algorithm pseudocode:

```
input value
while value is not valid
    output "ERROR"
    input value
```

- recall that "input value" is itself abstract pseudocode for:
  - declare a variable named **value**
  - get input from the user and assign it to **value**
  - echo the input **value**

While Loop Input Validation Example

```
double width;
width = Input.readDouble("Enter width: ");
while (width <= 0.0)
    width = Input.readDouble("negative width " +
        width + " entered; enter a positive width: ");

Output.showValue("after loop width is ", width);
```

- if positive width is input first time, while loop never executes
- if negative width is input, while loop will keep executing until a positive width is input
- good idea to echo invalid input in your error message (so user can see their error)

Input with Validation Pseudocode

- once we have an algorithm defined for validating input, we can abstract input validation details out of our program pseudocode
- EX: part of word problem: "Have the user input a positive width, giving them as many chances as necessary to enter a valid value."
- would be a single line of pseudocode:

input positive width

- a programmer could refer to the detailed algorithm pseudocode as needed to implement the Java (see last slide)

Input with Validation Pseudocode

- once we have an algorithm defined for validating input, we can abstract input validation details out of our program pseudocode
- EX: part of word problem: "Have the user input a positive width, giving them as many chances as necessary to enter a valid value."
- would be a single line of pseudocode:

```
input positive width
```

- a programmer could refer to the detailed algorithm pseudocode as needed to implement the Java (see last slide)

Counter-Controlled Loops

- a counter-controlled loop repeats a task a specified number of times
- implements the algorithm:

```
do some task #-of-times times
```

Counter-Controlled Loop Algorithm

✶here is pseudocode for the algorithm "do some task #-of-times times" using a counter-controlled loop:

```
count = 0
while count is less than #-of-times
    do the task once
    count = count + 1
```

✶count keeps track of the number of times we have completed the task

- initially the task hasn't been done, and count is 0
- each time the task is done, count increases by 1

Counter-Controlled Loop Example

✶EX abstract pseudocode:

- output "another cat" #-of-cats times

✶would translate into the following counter-controlled loop in Java:

```
int count = 0;
while (count < numCats)
{
    Output.showMessage("another cat");
    count = count + 1;
} // while
```

✶would output "*another cat*" numCats times

Counter-Controlled Loop Example (2)

```
int numTimes, count;
numTimes = Input.readInt("Enter # of times: ");
count = 0;
while (count < numTimes)
{
  Output.showValue("count is ", count);
  count = count + 1;
} // while
```

- outputs the integers from 0 to one less than numTimes
- EX: user inputs 3, will output:

  count is 0
  count is 1
  count is 2

Infinite Loops

- since a while-loop will continue looping until the condition becomes false the loop must contain statements that change the condition value
- if the condition never becomes false, the loop is an infinite loop that will never terminate
- if this happens, click on the **JGrasp** End button (to the left of the output window)

Infinite Loop Example

```
int num = 3, count = 0;
while (count < num)
    Output.showValue("count is ", count);
    count = count + 1;
```

- note missing {}'s; only the showValue will get executed in the while-loop
- count never increases, so this is an infinite loop

Counter-Controlled Loop Example

- spend some time looking at and experimenting with the program Circles.java
  - uses a counter-controlled loop to calculate the areas of multiple circles
  - uses input validation to get a positive number from the user

Sentinel Loops

- a sentinel loop gets input values from the user until the sentinel value is input
- the sentinel value is a special input value that represents the end of input
- pseudocode:

```
input value
while value is not equal to the sentinel value
    do the task with value
    input value
```

Sentinel Loop Example

- spend some time looking at and experimenting with the program Total.java
  - uses a sentinel loop to calculate the sum of a series of user-input numbers

Arbitrary Loops

- loops don't need to follow these common patterns
- to write a loop for a particular problem you must answer three questions:
  1. what task(s) will be performed each time the loop executes?
  2. what will be true if the loop should execute again?
     - that will be your loop condition
     - sometimes helps to instead ask what will be true when the loop is **done** executing? Then use the opposite as the loop condition
  3. how will the variables in the loop condition change each time the loop executes?
     - remember that if they don't change you will have an infinite loop
     - will the changes that occurs eventually make the loop condition false?

Word Problem Example

- "Have the user enter the current tuition and an expected percentage rate of annual increase in tuition. Use a loop to determine how many years it will take for the tuition to double."
  1. what task(s) will be performed each time the loop executes?
     - increase the tuition amount by the percentage
     - add one to the year count
  2. what will be true if the loop should execute again?
     - the tuition amount will be less than twice the original tuition amount
  3. how will the variables in the loop condition change each time the loop executes?
     - the tuition amount will increase by the annual percenage

Word Problem Example

* pseudocode:

```
input tuition
input rate
number of years = 0
twice original tuition = 2 * tuition
while tuition is less than twice original tuition
      tuition = tuition + tuition * rate / 100
      number of years = number of years + 1
output number of years
```