Conditional Loops

Counted Loops

The (`for loop`) loops allows us to perform an action a specific number of times.

Recall - Syntax for `for loop`

```
for (int counter= startValue; <condition>; <change counter>)
{
     Code that is will be repeated a known number of times
}
```

What if we do not know in advanced how many times an action will be repeated and cannot calculate it? We use conditional loops.

Conditional Loops Type 1 - The While Loop

```
while(<condition>)
{
     Code that will be repeated until the condition is false
}
     Something else
```

Before each pass of the loop, the `<condition>` is evaluation and the loop is executed if the `<condition>` is `TRUE`. If the `<condition>` is `FALSE`, the loop is not executed and the program continues after the loop.

Ex1

```
int x = 0;
System.out.println(“this loop runs until -1” +
                 “ is entered”);
while (x != -1)
{
        System.out.print(“enter another number: “);
        x = keyboard.nextInt();
}
System.out.println(“program finished”);
```

```
this loops run until -1 is entered
enter another number: 88
enter another number: 43
enter another number: 26
enter another number: -1
program finished
```

Input `88, 43, 26, -1, 12, 5`

The condition is `x != -1` and it checks if the value of x is NOT EQUAL to -1.
The initial value of x is 0, so it will enter the loop and continue checking the condition.

Ex2

```
int x = -1;
System.out.println(“this loop runs until -1 is entered”);
while (x != -1)
{
        System.out.print(“enter another number: “);
        x = keyboard.nextInt();
}
System.out.println(“program finished”);
```

```
this loops run until -1 is entered
program finished
```

Input `88, 43, 26, 12, 5`

The value of x was initialized to be -1 so the condition is not true and the program does no enter the loop.

Conditional Loops

Conditional Loops – Type 2 - The Do-While Loop

```
int x = -1;
System.out.println("this loop runs until -1 is entered");
do
{
  System.out.print("enter another number: ");
  x = c.readInt();
}
while (x != -1);

System.out.println("program finished");
```

```
this loops run until -1 is entered
enter another number: 88
enter another number: 43
enter another number: 26
enter another number: -1
program finished
```

Input `88, 43, 26, -1, 12, 5`

Even though the value of x was initialized to be -1, the condition is checked AFTER the loop is executed, so it will enter the loop. The loops does at least one pass, then the `<condition>` is evaluation and the loop is again executed if the expression is `TRUE`. If the `<condition>` is `FALSE`, the line after the loop is executed.

Ex3

```
int x = 55;
while( x < 25)
{
    System.out.println(“Pretty”);
    x = x + 5;
}
System.out.println(x);
```

```
55
```

Ex4

```
int count = 1
while(count < 100)
{
    System.out.println(count);
}
```

```
1
1
1
1
1
1
```

```
int x = 55;
do
{
    System.out.print(“Pretty ”);
    x = x + 5;
}
while (x < 25);
System.out.println(x);
```

```
Pretty 60
```

In ex4, the condition is always true, so the loop will continue forever. You have an infinite loop (not good).

Final Notes on Conditional Loops

1. A `While` or a `Do-While` loop has no maximum as to how many times it will be executed

2. When condition is evaluated at the beginning, the minimum number of times the loop will be executed is 0.

3. When condition is evaluated at the end of the loop, the minimum number of times the loop is executed is 1

Conditional Loops

Can I rewrite a `While` loop as a `For` loop and vice verca?

Ex5

```
int x = 3;
while(x < 20)
{
    System.out.print(x);
    x = x + 4;
}
```

↔

```
for (int x = 3; x < 20; x = x+4)
{
    System.out.print(x);
}
```

Ex6

```
for(int k = 13; k < 77; k = k+11)
{
    System.out.println(k);
}
```

↔

```
int k = 13;
while (k < 77)
{
    System.out.println(k);
    k = k + 11;
}
```

Every for loop can be written as a conditional loop, but not every conditional loop can be written as for loop.

The `While` or a `Do-While` loops are called a “conditional” because they are dependent on the condition. It is useful for validating use input.

Ex 7

```
int num;
System.out.print(“Enter a number between 1 and 100”);
num = keyboard.nextInt();

while(num < 1 || num > 100)
{
    System.out.print(“The number entered is outside the range, try again ”);
    num = keyboard.nextInt();

}
System.out.println(“Good job”);
```

We use a while loop when we don’t know how many times a loop must be carried out, where as in a for loop we knew how many times to loop.

For example: if we pick a basket of strawberries and have to cut the stem off every berry. We don’t know how many berries we have so we use a while loop.

```
While there are strawberries left in the basket
{
        Grab one berry
        Cut the stem off
}
```

Conditional Loops

Structure of a while loop

```
Initialize the test variable

While (a condition using the test variable is true)
{
      … code to do in the loop
      Update the test variable
}
```

Note: the condition is always tested at the start of the loop.

If the test variable is never changed (or not in the right direction) then the while condition will always be true causing the code to loop forever. This is called an infinite loop. When this happens you have to stop the program manually by clicking “quit” because the program will never end.

As a class, write a program that will take as input string typed on the keyboard until the string “z” is entered. When “z” is entered, the program will display the number of non “z” typed then exit.

```
 class MyChars
{

    public static void main (String[] args)
    {

        String input;
        int total=0;

        // Read the first char in (initialize the test variable)
        input = keyboard.next();

        // Loop until z is typed counting the number of letters types.
        while (!input.equals("z"))
        {
            total++;
            input = keyboard.next();
        }

        // Output the result of how many letters were typed.
        System.out.println(total + " number of letters were typed");

    } // main method
} // Chars class
```

Conditional Loops

Exercise:

1. Write a password program that will continuously ask for a “magic number” password until the user gets its right.

2. Write a code segment that prompts the user for a number between 5 and 12. Generate and display random numbers until one of the numbers produced matches the user number.

3. Write a program that reads in exam marks (integers) one at a time until a mark of less than zero is entered. The program will output the number of exam marks entered, the average exam mark, the minimum exam mark, and the maximum exam mark.