# Object-Oriented Development (CIS1056-N) Worksheet 04: Iteration Using While

## Before You Start

Remember: You are not expected to complete the entire brief within the allotted two hours, but to make a start and continue outside of the class.

Ensure you have completed all assessment tasks from Worksheet 1 and are comfortable with Java primitive types before beginning this worksheet. Attempt to complete this set of tasks before your next session. Any issues seek help from your tutors.

**Hint:** It is good practice to plan your programs first on paper using pseudocode. When we say pseudocode, we mean code that is halfway between English and a programming language, such as Python.

Read more about it here: <https://en.wikipedia.org/wiki/Pseudocode>

## Introduction

Last week we discovered how to make branching decisions in our code using if. This week we go further with program flow control by experimenting with iteration using the while keyword.

## 1. Paper Exercise Expected Outputs

Paper exercise so **no programming** (initially). What output would you expect from the following?

### Expected Output One

int count = 11;

while (count > 0)

{

    System.out.println("count is " + count);

    count--;

}

### Expected Output **Two**

int count = 0;

while (count < 10)

{

    System.out.println("count is " + count);

    count++;

}

### Expected Output Three

int count = 1;

while (count++ < 11)

{

    System.out.println("count is " + count);

}

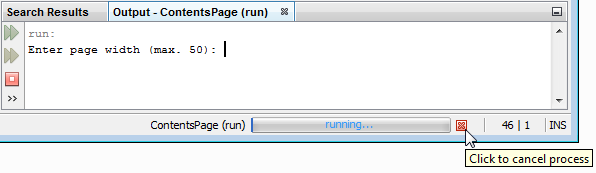
If unsure create a new NetBeans project and implement each of the loops in the main method. Were the outputs as you expected?

## 2. Paper Exercise While Conditions

Write in pseudocode the while loop constructions to print:

1. 1 6 11 16
2. 27 22 17 12
3. 9 8 6 5 4 3 2   
    (the number 7 is *deliberately* omitted for c)

Now implement your pseudocode in a new NetBeans project.

**Hint:** Whilst developing, programmers can sometimes accidentally produce code that enters an infinite (endless) loop. In NetBeans, if your program stops responding, click the X next to running… at the bottom of the output window to cancel/stop the program:

## 3. Counting Numbers

1. Write an algorithm to prompt the user to enter a series of integers (each on a separate line). The input is to be terminated with the value zero. Once the user has entered zero, display the number (count) of negative and positive values.
2. Write and test a program called CountNums, with a main method that implements the algorithm developed above.

## 4. Range Validation

1. Write a program to input a number between 1 and 100. The user should be asked to re-enter the number if it outside of the range.
2. Prepare a test plan (using example below) and test your program.

|  |  |  |  |
| --- | --- | --- | --- |
| Test Value | Purpose of test | Expected Result | Actual Result |
| 55 | Valid number. | Accepted |  |
| 999 | Invalid input | Rejected |  |
| … | … | … | … |

## 5. Triangle Size

1. Write a program to input a triangle size from the user and validate it to ensure it is less than 20. The user should be prompted to re-enter the triangle size if the input is invalid.
2. Update your program to ensure that triangle size is an odd number between 3 and 19 inclusive.

## 6. Yes/No Check

Write a program that asks the user “Are you sure you want to continue (Y/N)?” and validates the input to ensue only Y, N, Yes and No responses are accepted.

Invalid responses should result in the user being asked the question again (with an appropriate error message).

## 7. Loan Balance

A simple program is required to show the remaining balance on a loan. The program will prompt the user to enter a loan amount and the monthly payment, and then display a list of monthly balances. The process should repeat until the user enters a negative number for the loan amount. For example:

Loan Amount: 300

Repayment Amount: 50

Month 1: £250

Month 2: £200

Month 3: £150

Month 4: £100

Month 5: £50

Month 6: £0

Optional: Display the Balance as Year 1, Month 1: £xxx

To get you started look the following pseudocode:

Get loan amount

Get monthly payment

Set balance to loan amount

Set month to 0

While balance > 0

balance = balance – payment

If balance < 0

Set balance to 0

Increment month

Display month and balance

End of Loop

## 8. Implement Multiple Transactions

Take a copy of solution to question 8. Loan Balance. Update the algorithm to prompt a user to enter a new loan and monthly payment amount each time the previous balance is cleared. Use the pseudocode below to guide you.

**Get loan amount**

**While loan amount >= 0**

Get monthly payment

Set balance to loan amount

Set month to 0

While balance > 0

balance = balance – payment

If balance < 0

Set balance to 0

Increment month

Display month and balance

End of Loop

**Get loan amount**

**End of Loop**

Improve you solution to include validation. Ensure that a user is unable to specify a monthly payment that is greater than the initial loan amount.

## 9. Vending Machine Simulator

A vending machine has the following items:

1. Coke (45p)
2. Crisp (35p)
3. Small Chocolate Bar (23p)
4. Bottled Water (32p)

The user is shown a menu and asked to select an item (1-4). Once a valid option has been selected the price is displayed and the user is asked to enter the appropriate amount. The amount is checked, and the appropriate item is dispensed with the correct change. The machine only accepts and returns 1p, 2p, 5p, 10p and 20p coins.

The program is repeated until the shutdown code (999) is entered at the menu prompt.

Write the pseudocode to implement the above operation. include the logic to determine how many of each coin will be required to make-up the change. Show your pseudocode to the tutor and once approved, implement using a new NetBeans project.

## 10. Employee (Case Study) Validation

Update the **EmployeeManager** (see Worksheet 2) to improve the validation. If the validation rule (see below) fails, the program should display an appropriate error message and ask for the input again.

You may need to do some *independent* research to fulfil each of the validation rules. This exercise relies on understanding *Strings* (coming next week).

Each input should be validated before moving on to the next.

Validation Rules:

1. Employee Name cannot be empty (must be at least one character long).
2. Staff number must start with a letter and be followed by two digits; for example: D65.
3. Number of hours worked cannot be zero or below or over 100.

## Document History

Revision 0 (14-Oct-22): This is the initial version of the 2022/23 exercise.