# Object-Oriented Development (CIS1056-N) Worksheet 06: Methods

## 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.

Before proceeding with methods, you must be comfortable with program flow control. You should be able to read, understand, and write conditional (if) and iterative statements (while and for). Any issues seek help from your tutors.

## Introduction

We have now covered two of the main aspects of program flow control by experimenting with conditions and iteration.

Now it is time to start minimising code repetition as our programs grow by encapsulating common algorithms into their own single methods that we can call throughout a program.

**Hint:** In all future exercises you should consider if methods should be used to break up programs into logical sections.

## 1. getMark()

Create a new program GradeMethods. Complete the following program that **inputs** and **validates** a mark (**0-100**). You need to include a method called getMark(). See code below to get you started.

package grademethods;

public class GradeMethods {

    public static void main(String[] args) {

        int mark = getMark();

        System.out.print("The Mark is: " + mark);

    }

    public static int getMark() {

        // Complete this code"

        return 0;

    }

}

## 2. getGrade(int mark)

Extend the GradeMethods program to display the grade for the mark entered. The program should include a method that takes an integer parameter and return the grade (char). Further details on how to determine the grade are included in Week 3 Lecture 2. The rules are:

A is 70 or more, B is 60-69, C is 50-59, D is 40-49 and F is 0-39.

See the code below to get you started:

package grademethods;

public class GradeMethods {

    public static void main(String[] args) {

        int mark = getMark();

        System.out.print("The Mark is: " + mark);

        char grade = getGrade(mark);

        System.out.print("The Grade is: " + mark);

    }

    public static int getMark() {

        // Complete this code

        return 0;

    }

    //            return |  method   |  input

    //             type  |   name    | paramter

    public static  char  getGrade (int mark)

    {

        // Complete this code

        return 'F';

    }

}

## 3. calcArea

Complete the following program that inputs the width and height of a rectangle and outputs the area. You need to include a method called calcArea that takes two integer parameters (width and height) and returns the area.

import java.util.Scanner;

public class Area {

   public static void main(String[] args)

   {

        Scanner in = new Scanner(System.in);

        System.out.print("Please enter width: ");

        int width = in.nextInt();

        System.out.print("Please enter height: ");

        int height = in.nextInt();

        System.out.println("\nThe area is: " + calcArea(width, height));

   }

   // Add the calcArea method here

}

Compile and test the program.

## 4. Integer to Months

Develop a program that asks the user to input a number between 1 and 12 (inclusive) and displays the month name.

The program must **include a method** that given the month number prints out the name of the appropriate month.

Test your program with valid and invalid values (e.g. 13).

## 5. Times tables

Develop a program that asks the user to input 2 positive numbers. The two numbers are used to output a times table grid where the first number entered is the maximum rows and the second number entered is the maximum columns.

**Example run:**

**Please enter the first number: -1**

**\*\* Error! Negative number or 0 entered. Enter positive numbers only.**

**Please enter the first number: 5**

**Please enter the second number: 8**

**1 2 3 4 5 6 7 8**

**2 4 6 8 10 12 14 16**

**3 6 9 12 15 18 21 24**

**4 8 12 16 20 24 28 32**

**5 10 15 20 25 30 35 40**

## 6. Smallest Number

Write a program that asks the user to enter 3 numbers (1-100 inclusive) and outputs the smallest number. The program should include methods for:

1. Inputting and validating a number: Get the user input and check that it is between 1 and 100 (inclusive). Once a valid number has been entered the value is returned from the method.
2. Comparing the three numbers and find the smallest number. This method will take three parameters and return the smallest number.

## 7. Employee (Case Study) – Validation Methods

**EmployeeManager** is a program you have been working on in previous worksheets. If you have not already done so, implement the following:

### From Worksheet 2

Create a new java project called EmployeeManager that calculates and displays the weekly wage (without tax) from the number of hours worked and hourly rate.

For the initial version of the program, we will use fixed (literal) value for the variables:

* Hours Worked: **35**
* Hourly Rate: **9.75**

The weekly wage should be: **341.25**.

1. Using the values above, create variables (with appropriate identifiers) to calculate the weekly wage, then print to screen.
2. Update the solution to ask the user for the number of hours worked. The hourly rate should remain fixed at £9.75.
3. Modify your program, to pay staff an additional £4.75 for any hours above the basic 40 hour working week. For example, if an employee work 45 hours, the total payment would be:  
     
   £450.00 (£9.50 x 45 + £4.50 x 5).
4. Ensure you test your solution rigorously.

### From Worksheet 4

Update the **EmployeeManager** 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.

Before asking for the hours worked, ask the user to input their name and staff number.

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.

### Worksheet 6 Task

Move the input and validation for employee name and staff number into methods. For each input create two methods, for example, employee name would have:

* **isValidName**: takes a single parameter (the employee’s name) and returns true for valid names.
* **getName**: prompts the user to enter a name and then validates it. If the name is not valid, an appropriate error message is displayed, and the user is asked to input the name again. Once a valid name is entered it is returned from the method.

## 8. Leap Year

In week 2’s practical you were asked to write a program that asks the user to input a month number (e.g. 2) and year (e.g. 2000). The program then displays the number of days in the month (considering leap years).

* Leap years occur if the year is divisible by 4, except for the century years.
* Only the centuries divisible by 400 are leap years. e.g. 1900 was not a leap year; 2000 was a leap year.

Update the program to incorporate two methods: isLeapYear and daysInMonth.

## 9. Menu for Temperature Conversion

Write a menu-driven program that provides three options:

1. Option 1 allows the user to enter a temperature in Celsius (C) and displays the corresponding Fahrenheit (F) temperature.
2. Option 2 allows the user to enter a temperature in Fahrenheit (F) and displays the corresponding Celsius (C) temperature.
3. Option 3 allows the user to quit.

The formulae are: F = 9 / 5 x C + 32

C = 5 x (F – 32) / 9

Adapt your program so that the user is not allowed to enter a temperature below absolute zero (this is -273.15 C or -459.67 F).

## 10. Methods for Discount

If a customer places order for more than 10 items he/she receives a 1% discount on the total (quantity x item price).

Write and test a program to calculate the total price (including any discount).

Update your program to include the following rule: Trade Customer receive an additional 3%.

## 11. Binary to Decimal Conversion

Write and test a program that get a binary number as a string from the user and converts it to a decimal number. The program should include two methods:

1. getBinary: reads binary number from the user and validates it.
2. bin2Dec: converts a binary number (string) into a decimal value.

## 12. Decimal to Binary Conversion

Write and test a program that get a decimal number as an integer from the user and display the binary equivalent. The program should include a method called dec2Bin that converts an integer to a binary number (string).

## 13. Hexadecimal Conversion

Write two programs like exercise 10 and 11 to convert a hexadecimal number to binary and vice-versa.

## Document History

Revision 0 (22-Oct-21): This is the initial version of the 2021/22 exercise.