# Assignment 1

**Object Oriented Design and Development 1 (15%)**

## DUE DATE

Session 5

## OBJECTIVES

- Describe the requirements, software and the packages necessary to create Java applications.
- Use the correct Java syntax for input/output statements, decision structures, repetition structures and the various data types and operators.
- Declare a class and use it to create an object.
- Implement a class's behaviour as methods and attributes as instance variables.
- Compare and contrast local method variables and instance variables and how they are handled in Java.
- Use the if and if …else selection statements to choose between alternative actions.
- Use the while, counter controlled, and sentinel controlled iteration statements to execute statements in a program repeatedly.
- Use the logical operators to form complex conditional expressions in control statements.

## DESCRIPTION

For each of the following problems:

- Start by developing the program logic in form of Flowchart, or pseudocode algorithm.
- Based on the program logic, implement each solution in Java code.

## INSTRUCTIONS

1. Write an application that reads an integer and determines and prints whether it's odd or even. [Hint: Use the remainder operator. An even number is a multiple of 2. Any multiple of 2 leaves a remainder of 0 when divided by 2.]

2. In Chapter 2, you learned about integers and the type int. Java can also represent uppercase letters, lowercase letters, and a considerable variety of special symbols. Every character has a corresponding integer representation. The set of characters a computer uses together with the corresponding integer representations for those characters is called that computer's character

set. You can indicate a character value in a program simply by enclosing that character in single quotes, as in `‘A’`.
You can determine a character’s integer equivalent by preceding that character with (`int`), as in

```
(int) ‘A’
```

An operator of this form is called a cast operator. The following statement outputs a character and its integer equivalent:

```
System.out.printf("The character %c has the value %d%n", ‘A’, ((int)
‘A’));
```

When the preceding statement executes, it displays the character A and the value 65 (from the Unicode® character set) as part of the string. The format specifier %c is a placeholder for a character (in this case, the character ‘A’).

Using statements similar to the one shown earlier in this exercise, write an application that displays the integer equivalents of some uppercase letters, lowercase letters, digits, and special symbols. Prompt the user to enter a character and convert the character to the integer equivalents.

3. Create a class called `Invoice` that a hardware store might use to represent an invoice for an item sold at the store. An Invoice should include four pieces of information as instance variables—a part number (type `String`), a part description (type `String`), a quantity of the item being purchased (type `int`) and a price per item (`double`). Your class should have a constructor that initializes the four instance variables. Provide a set and a get method for each instance variable. In addition, provide a method named `getInvoiceAmount` that calculates the invoice amount (i.e., multiplies the quantity by the price per item), then returns the amount as a double value. If the quantity is not positive, it should be set to 0. If the price per item is not positive, it should be set to 0.0. Write a test app named `InvoiceTest` that demonstrates class Invoice’s capabilities.

4. Create a class called `Date` that includes three instance variables—a `month` (type `int`), a `day` (type `int`) and a `year` (type `int`). Provide a constructor that initializes the three instance variables and assumes that the values provided are correct. Provide a set and a get method for each instance variable. Provide a method `displayDate` that displays the `month`, `day` and `year` separated by forward slashes (/). Write a test app named `DateTest` that demonstrates class Date’s capabilities. Drivers are concerned with the mileage their automobiles get (Gas Mileage). One driver has kept track of several trips by recording the miles driven and gallons used for each tankful. Develop a Java application that will input the miles driven and gallons used (both as integers) for each trip. The program should calculate and display the miles per gallon obtained for each trip and print the combined miles per gallon obtained for all trips up to this point. All averaging calculations should produce floating-point results. Use class Scanner and sentinel-controlled iteration to obtain the data from the user.

5. Sales Commission Calculator: A large company pays its salespeople on a commission basis. The salespeople receive $200 per week plus 9% of their gross sales for that week. For example, a salesperson who sells $5,000 worth of merchandise in a week receives
$200 plus 9% of $5,000, or a total of $650. You’ve been supplied with a list of the items sold by each salesperson. The values of these items are shown in the chart below .
Develop a Java application that inputs one salesperson’s items sold for last week and calculates and displays that salesperson’s earnings. There’s no limit to the number of items that can be sold.

| Item | Value |
|---|---|
| 1 | 239.99 |
| 2 | 129.75 |
| 3 | 99.95 |
| 4 | 350.89 |

6. Write an application that displays the following patterns separately, one below the other. Use `for` loops to generate the patterns. All asterisks (*) should be printed by a single statement of the form `System.out.print('*');` which causes the asterisks to print side by side. A statement of the form `System.out.println();` can be used to move to the next line. A statement of the form `System.out.print(' ');` can be used to display a space for the last two patterns. There should be no other output statements in the program. [*Hint:* The last two patterns require that each line begin with an appropriate number of blank spaces.]

| (a) | (b) | (c) | (d) |
|---|---|---|---|
| * | ********** | ********** | * |
| ** | ********* | ********* | ** |
| *** | ******** | ******** | *** |
| **** | ******* | ******* | **** |
| ***** | ****** | ****** | ***** |
| ****** | ***** | ***** | ****** |
| ******* | **** | **** | ******* |
| ******** | *** | *** | ******** |
| ********* | ** | ** | ********* |
| ********** | * | * | ********** |

7. ***Diamond-Printing Program:*** Write an application that prints the following diamond shape. You may use output statements that print a single asterisk (*), a single space or a single new-line character. Maximize your use of iteration (with nested `for` statements), and minimize the number of output statements.

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

8. ***Reversing Digits:*** Write a method that takes an integer value and returns the number with its digits reversed. For example, given the number 7631, the method should return 1367. Incorporate the method into an application that reads a value from the user and displays the result.

9. ***Guess the Number:*** Write an application that plays “guess the number” as follows: Your program chooses the number to be guessed by selecting a random integer in the range 1 to 1000. The application displays the prompt `Guess a number between 1 and 1000.` The player inputs a first guess. If the player’s guess is incorrect, your program should display `Too high. Try again.` or `Too low. Try again.` to help the player “zero in” on the correct answer. The

program should prompt the user for the next guess. When the user enters the correct answer, display `Congratulations. You guessed the number!`, and allow the user to choose whether to play again. [*Note:* The guessing technique employed in this problem is similar to a binary search, which is discussed in **Chapter 19**, Searching, Sorting and Big O.]

-

## SUBMISSION INSTRUCTIONS

Your submission must include a Word document with the logic for each problem in this assignment as well as each problem's source code. Zip your final project and submit the zip file.

Work must be submitted in the correct file type and be properly labelled as per the College naming convention:
NAME_COURSE_ASSIGNMENT. E.g. XuXiaLing_FM50D_A01.

## GRADING CRITERIA

Assignment Value: **15%**

| Grading Criteria | Grading |
|---|---|
| **Problem 1** | /10 |
| **Problem 2** | /10 |
| **Problem 3** | /10 |
| **Problem 4** | /10 |
| **Problem 5** | /10 |
| **Problem 6** | /10 |
| **Problem 7** | /10 |
| **Problem 8** | /10 |
| **Problem 9** | /10 |
| **TOTAL** | **/90** |