Daniel Pienaar

Student Number: 21004169

PROG5121

Lecturer: Handsome Mpofu

3 June 2021

Programming 1A Assignment 1

Table of Contents

TABLES, DIAGRAMS AND FIGURES

1. INTRODUCTION

This assignment will use Java programming concepts to develop programs to accomplish 3 different tasks. A computer program uses sets of instructions to make the computer do what the programmer wants (Farrel, 2019).

In this assignment we will use Data, Methods, Classes and Objects among other concepts to accomplish these tasks.

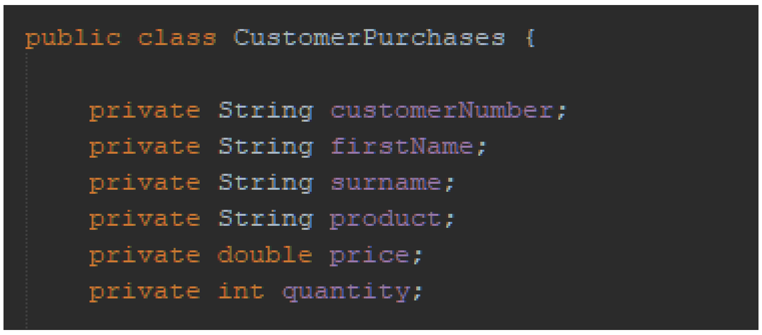
Please note the format of these 3 programs: Each has been separated into its own package within the Java project.

2. MUSIC PURCHASES APPLICATION

This application makes use of the scanner class to get input from the user, a CustomerPurchases class object to store the data and various static methods to print the resulting reports for a musical equipment store's purchases. The application has 3 classes: MusicPurchases (main class), Printing, and CustomerPurchases.

2.1. Data Storage and input

The CustomerPurchases cp object is created from the following class in the main method to prepare for the user input.



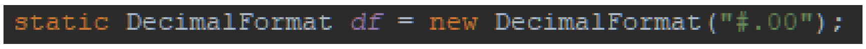
```java
public class CustomerPurchases {

    private String customerNumber;
    private String firstName;
    private String surname;
    private String product;
    private double price;
    private int quantity;
```

Figure 1: The CustomerPurchases class and its attributes. The class also contains get and set methods for these attributes.

Data is then obtained from the user with a scanner object and System.out.print() prompts.

2.2. Data formatting

This program makes use of the DecimalFormat class to properly format price values. Figure 2 shows the adapted piece of code (Oracle, 2020).



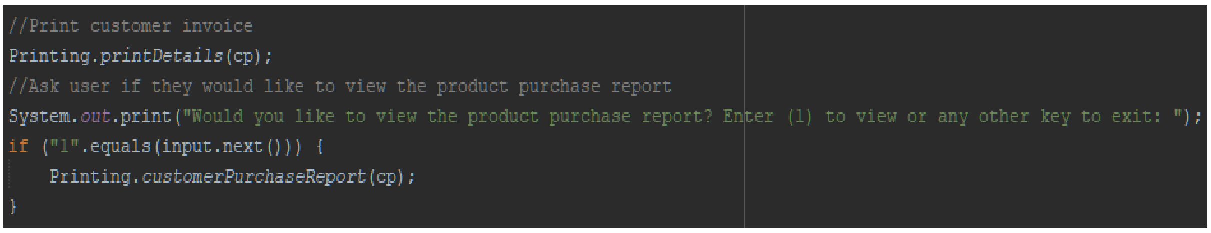
```java
static DecimalFormat df = new DecimalFormat("#.00");
```

Figure 2: A global static instantiation of DecimalFormat in the Printing class.

2.3. Report Printing

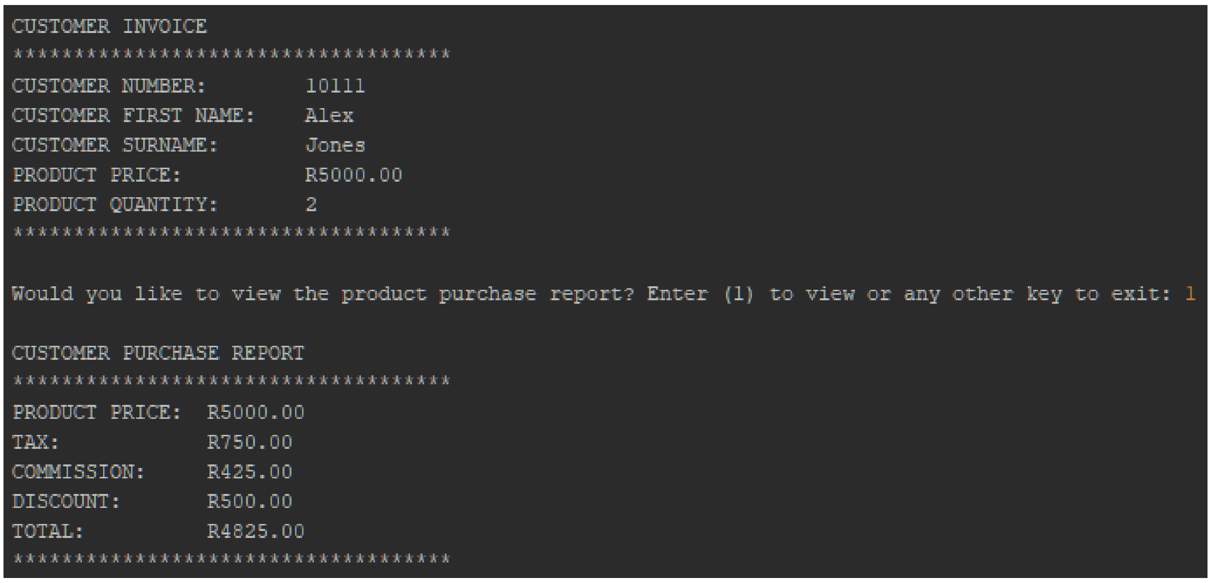
Using 2 static methods in the printing class, the details are obtained from a CustomerPurchases object passed to them, and then printed in the correct format.

The second report is only displayed if the user confirms it.



```
//Print customer invoice
Printing.printDetails(cp);
//Ask user if they would like to view the product purchase report
System.out.print("Would you like to view the product purchase report? Enter (1) to view or any other key to exit: ");
if ("1".equals(input.next())) {
    Printing.customerPurchaseReport(cp);
}
```

Figure 3: The code used to print the 2 reports. customerPurchaseReport is only called after confirmation.



```
CUSTOMER INVOICE
************************************
CUSTOMER NUMBER:        10111
CUSTOMER FIRST NAME:    Alex
CUSTOMER SURNAME:       Jones
PRODUCT PRICE:          R5000.00
PRODUCT QUANTITY:       2
************************************

Would you like to view the product purchase report? Enter (1) to view or any other key to exit: 1

CUSTOMER PURCHASE REPORT
************************************
PRODUCT PRICE:  R5000.00
TAX:            R750.00
COMMISSION:     R425.00
DISCOUNT:       R500.00
TOTAL:          R4825.00
************************************
```

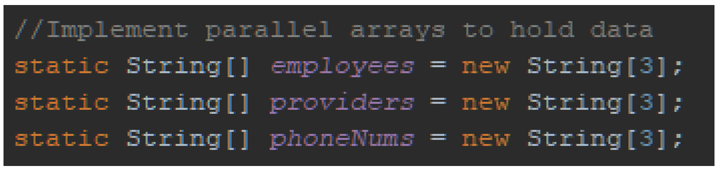
Figure 4: The resultant output after getting details.

3. PHONE NUMBER GENERATOR APPLICATION

This application gets the names of 3 employees and assigns them to one of 3 different network providers. It then generates random phone numbers for each of them. The two static methods are called once for each employee to generate the required information. The application has one main class: PhoneNumberGenerator.

3.1. Data storage and input

The method of data storage in this application comprises of 3 global static parallel string arrays:



```
//Implement parallel arrays to hold data
static String[] employees = new String[3];
static String[] providers = new String[3];
static String[] phoneNums = new String[3];
```

Figure 5: 3 global string arrays used to store data in parallel.

Data for the first employee is stored in index 0, index 1 for the second, and index 2 for the third.

3.2. Random assignment of network providers

The program calls the static String getNetworkProvider() method to return either "VODACOM", "CELL C", or "MTN" based on the value generated by the random number generator, as shown in the following figure (JavaTpoint, 2021):

```
int randProvider = (int) (Math.random() * (3 - 1 + 1) + 1);
```

Figure 6: Inclusive random number generation from 1 to 3

3.3. Random phone number generation

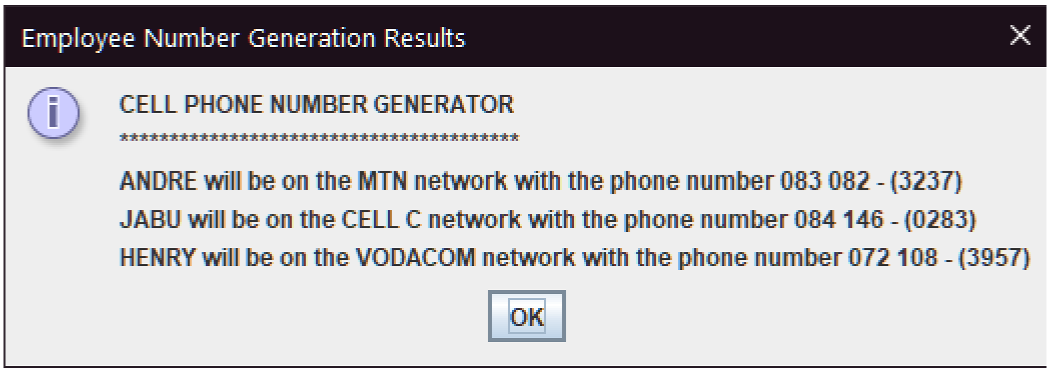
After the provider is assigned, the number generation begins in generatePhoneNumber(). First, the 3-digit provider prefix is added to the number using an if statement that checks the previously assigned provider name. Then the next 3 digits are generated individually and added to the number. Then some formatting, then the final 4 digits, followed by the ending formatting.

```
randNum = (int) (Math.random() * (9 - 0 + 1) + 0);
number += randNum;
```

Figure 7: Concatenation of individual digits to the phone number. This occurs in two for loops that run 3 times and 4 times, respectively.

3.4. Displaying the message with results

Once the details are generated for the 3 employees, they are displayed in a JOptionPane message dialog.



```
CELL PHONE NUMBER GENERATOR
******************************************
ANDRE will be on the MTN network with the phone number 083 082 - (3237)
JABU will be on the CELL C network with the phone number 084 146 - (0283)
HENRY will be on the VODACOM network with the phone number 072 108 - (3957)
```

Figure 8: Results of the number generation.

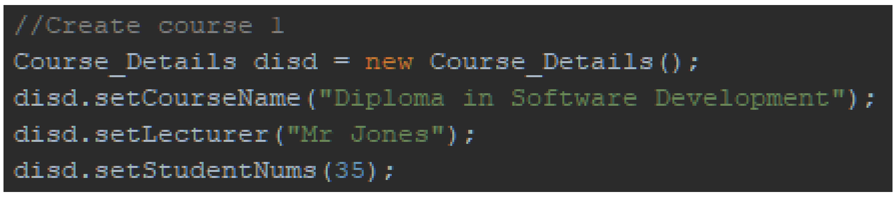
4. COURSE REPORT APPLICATION

This application prints out course reports for 3 different courses. It makes use of Course_Details objects to store information, and continuously asks the user which course they would like to view the report for until they exit.

There are 2 classes: CourseReport (main class) and Course_Details.

4.1. Storing data

The data for this application is stored directly in the application in a Course_Details object, not entered by a user. The figure below shows the data entry method of one of the 3 courses:

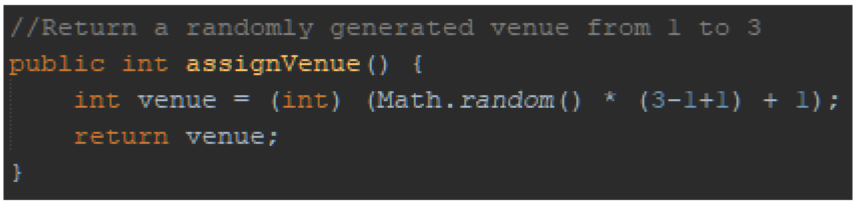


```java
//Create course 1
Course_Details disd = new Course_Details();
disd.setCourseName("Diploma in Software Development");
disd.setLecturer("Mr Jones");
disd.setStudentNums(35);
```

Figure 9: Storing the data of one of the courses, making use of the set methods to fill the 3 class attributes.

4.2. Assigning a venue to a course

A random venue from 1 to 3 is assigned to a course upon printing the report to the console. This is done by a method in the Course_Details class called assignVenue, which returns the random int as shown below:

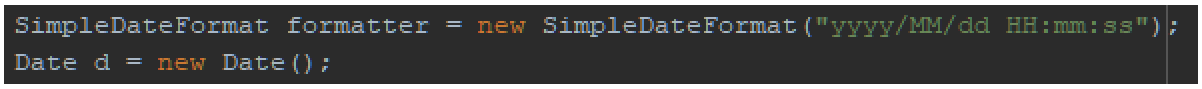


```java
//Return a randomly generated venue from 1 to 3
public int assignVenue() {
    int venue = (int) (Math.random() * (3-1+1) + 1);
    return venue;
}
```

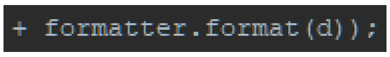
Figure 10: Returning a random venue for the course report.

4.3. Printing out the course report, and asking the user to continue or exit

A date is attached to every course report printed. This was done with the help of a SimpleDateFormat formatter (Oracle, 2021).



```java
SimpleDateFormat formatter = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
Date d = new Date();
```



```java
+ formatter.format(d));
```

Figure 11 and 12: The creation of a SimpleDateFormat formatter, and its further use in the course report.

The date and the course report are printed using a static method printCourseReport() in the main class. Once this is complete, the following is displayed to the user:

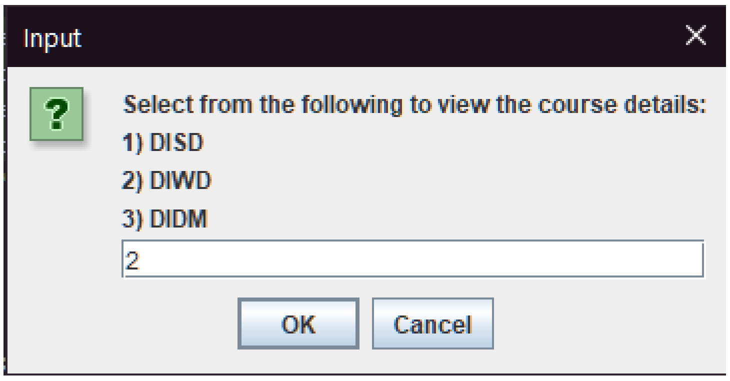
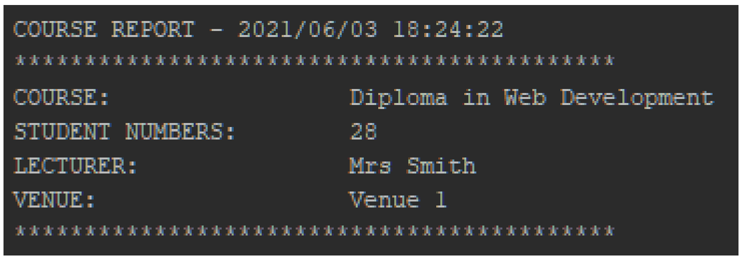


Figure 13: The initial menu showed to the user when starting the application, with option 2 selected.



```
COURSE REPORT - 2021/06/03 18:24:22
*******************************************
COURSE:                 Diploma in Web Development
STUDENT NUMBERS:        28
LECTURER:               Mrs Smith
VENUE:                  Venue 1
*******************************************
```

Figure 14: The printed course report

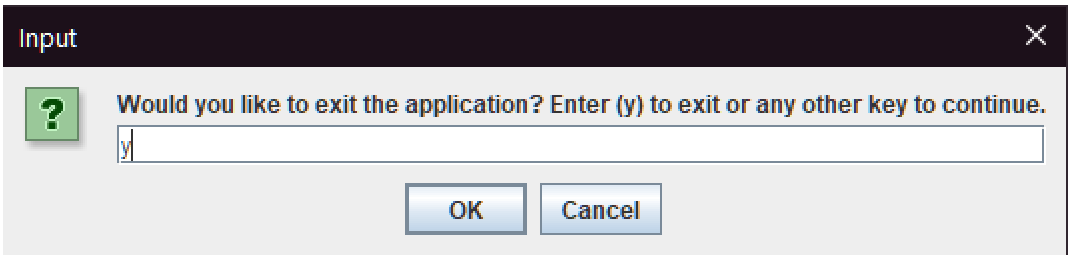

Input
Would you like to exit the application? Enter (y) to exit or any other key to continue.
y
OK
Cancel

Figure 15: The dialog menu asking the user if they would like to exit or view another report.

5. CONCLUSION

These solutions show how data can be stored with java concepts such as parallel arrays and objects, and especially how objects can be used effectively when needing to store multiple values of related information.

Get and set methods were used to work with the data in these objects and to display information to the user.

Loops were used to control the flow of the programs, as well as to generate multiple individual random numbers.

Lastly, the applications explored some of the formatting options used to properly display prices, dates and times.

REFERENCE LIST

Farrel, J. 2019. *Java Programming* 9th ed. Boston: Cengage Learning.

JavaTpoint 2021. *How to Generate Random Number in Java* (n.d.). [Online] Available at: https://www.javatpoint.com/how-to-generate-random-number-in-java [Accessed: 3 June 2021].

Oracle 2021. *How to Make Dialogs (The Java™ Tutorials)*, (n.d.). [Online] Available at: https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html [Accessed: 2 June 2021].

Oracle 2021. *SimpleDateFormat (Java SE 16 & JDK 16)* (2021). [Online] Available at: https://docs.oracle.com/en/java/javase/16/docs/api/java.base/java/text/SimpleDateFormat.html [Accessed: 3 June 2021].

Oracle 2020. *DecimalFormat (Java Platform SE 7)* (n.d.). [Online] Available at: https://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html [Accessed: 3 June 2021].