Daniel Pienaar

Student Number: 21004169

PROG6112

Lecturer: Handsome Mpofu

21 September 2021

Programming 1B Assignment 1

Table of Contents

TABLES, DIAGRAMS AND FIGURES

1. INTRODUCTION

This assignment consists of 2 questions, the first consisting of a product management system for an IT company, and the second being an open-ended question that makes use of various concepts, specifically advanced arrays, and inheritance. This second question was therefore made into a management system for classes in a fictitious IT Coding School.

Both questions make use of console keyboard input to store data into arrays in the program and make use of menus to prompt the user to change and display data.

2. EXTREME IT PRODUCTS APPLICATION

This application manages the products of a company known as Extreme IT Products. It separates its different functions into methods that are called by input from a console menu. It has 2 classes: Products (main class with the working methods) and ReportData (Stores product details and has methods to format and produce a report with the product details).

2.1. Data storage and input

The main method of storing data in this application is the ArrayList, which is a dynamically resizable array that can hold any type of object by specifying its data type in angle brackets (Farrel, 2019). This ArrayList stores ReportData objects. Once product details are captured by the program, a new ReportData object is created and added to the array.

```
//Arraylist storing captured products
static ArrayList<ReportData> productArr = new ArrayList<>();
```

Figure 1: The productArr ArrayList storing ReportData objects. This ArrayList is global in order to be accessed anywhere in the Products class.

```
//Constructor
ReportData(String code, String name, String warranty, String category, double price, int stock, String supplier) {
    this.code = code;
    this.name = name;
    setWarranty(warranty);
    setCategory(category);
    this.price = price;
    this.stock = stock;
    this.supplier = supplier;
}
```

Figure 2: The constructor in the ReportData class that takes the user input.

As seen in figure 2, the data entered for the warranty and category goes through further processing in their set methods before being stored.

Once the user starts the program, they will be met with this menu:

```
BRIGHT FUTURE TECHNOLOGIES APPLICATION
**************************************
Enter (1) to launch menu or any other key to exit
>> 1
Please select one of the following menu items:
(1) Capture a new product.
(2) Search for a product.
(3) Update a product.
(4) Delete a product.
(5) Print report.
(6) Exit application.
>>
```

Figure 3: The header and main menu of the application.

"Capture a new product" is the way for users to store new ReportData objects in the ArrayList. From then on, users can make use of the other menu options to work with the data in the ArrayList. After each operation is complete, users will be asked if they would like to return to this menu or exit.

2.2. Error handling and input validation

This program makes generous use of error handling techniques in order to prevent the user from being able to enter invalid data or crash the program. One such technique is the try-catch block (W3Schools, 2021):

```
//This try catch block handles errors that arise from the user not entering a double for the price
//The try catch implementations in this program were adapted from:
//https://www.w3schools.com/java/java_try_catch.asp
//Accessed 19 September 2021
try {
    double price = input.nextDouble();
    rd.setPrice(price);
    valid = true;
} catch (InputMismatchException e) {
    System.out.println("Please enter a valid price (numbers and decimals only).");
    valid = false;
}
```

Figure 4: An example of the try catch block being used to prevent the user from entering an invalid price value.

Another example of an interesting error encountered during development was the ConcurrentModificationException. According to JavaTPoint (2021), you can not modify a Collection such as an ArrayList, while simultaneously iterating over it. To mitigate this issue, I opted for a regular for loop instead of a for each loop, as seen below:

```java
//Instead of iterating over the Arraylist using a foreach loop, I use a regular for loop to avoid a ConcurrentModificationException
//Explanation adapted from:
//https://www.javatpoint.com/concurrentmodificationexception-in-java
//Accessed 19 September 2021
for (int i = 0; i < productArr.size(); i++) {
    if (delete.equals(productArr.get(i).getCode())) {
        found = true;
        System.out.print("Deleting product: " + productArr.get(i).getName()
                + "\nAre you sure? Enter (y) to delete or any other key to cancel.\n>> ");
        String confirm = input.nextLine();
        if (confirm.equalsIgnoreCase("y")) {
            productArr.remove(i);
            System.out.println("Product successfully deleted.");
            break;
        } else {
            System.out.println("Cancelling operation.");
        }
    }
}
```

Figure 5: Avoiding a ConcurrentModificationException

An important detail to note about using a scanner to get input is that you need to advance the scanner past invalid input entered for hasNext...() methods (polygenelubricants, 2010). If you do not then this can cause wrong input values, or in this program's case an infinite loop of invalid input. To solve this issue, I advance the scanner in this manner:

```java
//Capture price
String clear;
boolean validPrice = false;
double price = 0;
while (!validPrice) {
    try {
        System.out.print("Enter the price for " + name + " >> ");
        price = input.nextDouble();
        validPrice = true;
    } catch (InputMismatchException e) {
        System.out.println("Please enter a valid price (numbers and decimals only).");
        //Advances scanner past invalid input to prevent an infinite loop
        //Logic adapted from:
        //https://stackoverflow.com/questions/1794281/java-infinite-loop-using-scanner-in-hasnextint
        //User answered:
        //https://stackoverflow.com/users/276101/polygenelubricants
        //Accessed 19 September 2021
        clear = input.nextLine();
        validPrice = false;
    }
}
```

Figure 6: Using nextLine() to advance the scanner past invalid input.

2.3. Data formatting and presentation

Once the data is added to the array, methods in the Product class can call methods from the ReportData class to get the data for the array objects. This can be done either though the object's get methods, or in a format specified by the displayReport method:

```
//Display the product report on the console
public void displayReport() {
    System.out.println(""
            + "PRODUCT CODE:\t\t" + getCode()
            + "\nPRODUCT NAME:\t\t" + getName()
            + "\nPRODUCT WARRANTY:\t" + getWarranty()
            + "\nPRODUCT CATEGORY:\t" + getCategory()
            + "\nPRODUCT PRICE:\t\tR" + formatPrice(getPrice())
            + "\nPRODUCT STOCK LEVELS:\t" + getStock()
            + "\nPRODUCT SUPPLIER:\t" + getSupplier());
}
```

Figure 7: The displayReport() method. This method is used for displaying search results, and for the main report containing all products.

The formatPrice method shown in figure 7 uses a DecimalFormat object (Ligios, A. 2021) to return a double price with 2 decimal places.

```
//Price formatter
//Adapted from:
//https://www.baeldung.com/java-decimalformat
//Accessed 19 September 2021
public static final DecimalFormat df = new DecimalFormat("###.00");
```

Figure 8: The DecimalFormat price formatter.

Once the user has added and edited the data to their liking, they can print a report in this manner, using option 5 in the menu:

```
PRODUCT REPORT
============================================================
PRODUCT 1
------------------------------------------------------------
PRODUCT CODE:           A55
PRODUCT NAME:           EliteBook
PRODUCT WARRANTY:       2 years
PRODUCT CATEGORY:       Laptop
PRODUCT PRICE:          R15000.00
PRODUCT STOCK LEVELS:   3
PRODUCT SUPPLIER:       IT_4_Africa
------------------------------------------------------------
PRODUCT 2
------------------------------------------------------------
PRODUCT CODE:           A54
PRODUCT NAME:           PS5
PRODUCT WARRANTY:       6 months
PRODUCT CATEGORY:       Gaming Console
PRODUCT PRICE:          R12450.50
PRODUCT STOCK LEVELS:   6
PRODUCT SUPPLIER:       Gaming_4_Africa
------------------------------------------------------------
============================================================
TOTAL PRODUCT COUNT:    2
TOTAL PRODUCT VALUE:    R27450.50
AVERAGE PRODUCT VALUE:  R13725.25
============================================================
```

Figure 9: The final product report, showing every product in the ArrayList, including some statistics.

3. SCHOOL MANAGEMENT APPLICATION

This application uses a 2D array of People to create a table showing the classes in an IT coding school. It also makes use of inheritance, where Staff and Student extend Person, and the SchoolManagement main class makes use of these classes.

3.1. Data storage and input

This program stores the members of the school in a fixed size 2D array that defaults to 5 classes and 15 members:

```
//CLASSES and MEMBERS variables. The capacity limit for the school is by default 5 CLASSES and 15 MEMBERS per class. Change here if necessary.
static final int CLASSES = 5;
static final int MEMBERS = 15;
static Person[][] classTable = new Person[MEMBERS][CLASSES];
//Tracks sizes of the 5 CLASSES. Initialized to 1, because staff space is reserved.
static int[] classSizes = {1, 1, 1, 1, 1};
```

Figure 10: The 2D array and the size variables and array.

The classSizes array stores the current number of members of each class. It is primarily used when students are added or deleted.

The user will be shown the following menu on start-up:

```
--------------------------------------------------
Welcome to the IT Coding School management program
--------------------------------------------------


--------------------------------------------------
(1) Add a person
(2) Search for a person's details
(3) Update a person's details
(4) Delete a person
(5) Display school class report
(6) Exit
>>
```

Figure 11: The main menu with the displayed options

The user can use “Add a person” to add staff or students, then edit or display them with the other options.

3.2. Data formatting and presentation

The main feature here is a table, as seen below. This code was adapted from JavaTPoint (2021).

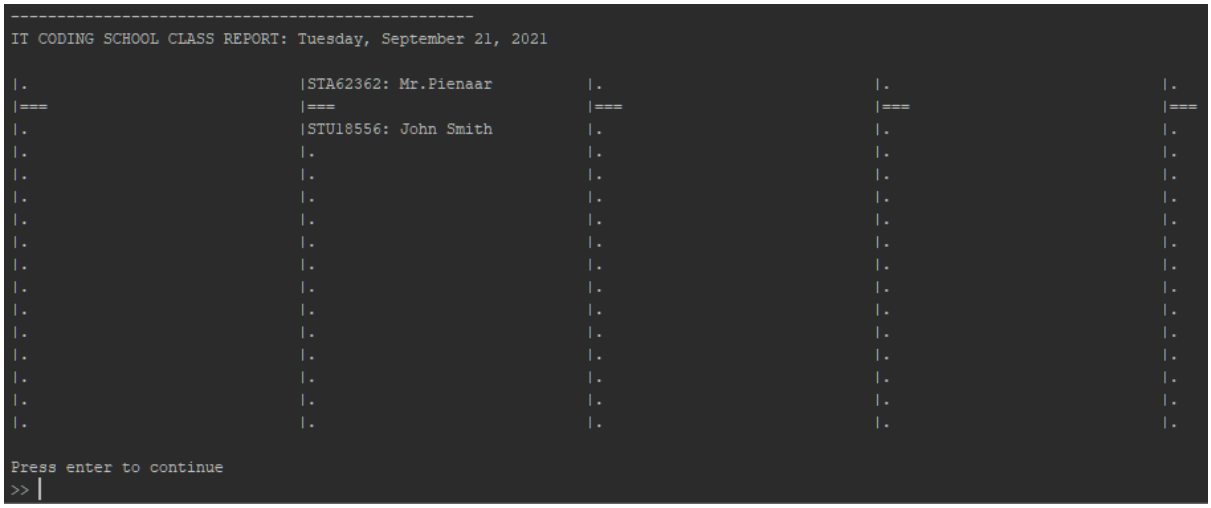

Figure 12: The table showing the 5 classes. Empty spaces are represented using a ".".

The date seen at the top of the table in figure 12 was formatted using this code (Baeldung. 2021):

```
//Date header for table
//Code adapted from:
//https://www.baeldung.com/java-datetimeformatter
//Accessed 21 September 2021
LocalDate myDateObj = LocalDate.now();
String formattedDate = DateTimeFormatter.ofLocalizedDate(FormatStyle.FULL).format(myDateObj);
System.out.println("IT CODING SCHOOL CLASS REPORT: " + formattedDate + "\n");
```

Figure 13: The DateTimeFormatter.

Here we can see the information of a student after their details are searched. Some of the details here can be changed using other menu options or deleted. If a student is deleted, the rest of the class gets shifted up. Students are always added after one another, but only one staff member can be added or overwritten at the top of the class.

```
Enter the ID of the person you want to search for >> STU90193
Daniel Pienaar - Student
===
GENDER:                 Male
ID:                     STU90193
CLASS:                  Group 3
CURRENT GRADE:          A (90.0%)
CURRENT ATTENDANCE:     80.0%

Press enter to continue
>>
```

Figure 14: Details of a student displayed.

4. CONCLUSION

These applications make use of various types of arrays to store and manipulate data. This proved to be quite effective, however the main issue here is having to re-enter data every time the program starts again. If I were to remake these programs, I would use a text file, or preferably a database to store the data.

The method of input used (console), also proved to be much less intuitive than GUI input. However, a menu that uses input to call methods with a switch case proved to be an effective solution.

REFERENCE LIST

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

W3Schools. 2021. Java Exceptions (Try...Catch) (n.d.). [Online]. Available at: https://www.w3schools.com/java/java_try_catch.asp [Accessed 21 September 2021].

JavaTPoint. 2021. ConcurrentModificationException in Java (n.d.). [Online]. Available at: https://www.javatpoint.com/concurrentmodificationexception-in-java [Accessed 21 September 2021].

polygenelubricants. (2010) Java: Infinite loop using Scanner in.hasNextInt(), 4 May 2010. [Online]. Available at: https://stackoverflow.com/questions/1794281/java-infinite-loop-using-scanner-in-hasnextint [Accessed 21 September 2021].

Ligios, A. (2021) A Practical Guide to DecimalFormat, 15 May 2021. [Online]. Available at: https://www.baeldung.com/java-decimalformat [Accessed 21 September 2021].

Baeldung. 2021. Guide to DateTimeFormatter, 19 May 2021. [Online]. Available at: https://www.baeldung.com/java-datetimeformatter [Acessed 21 September 2021].

JavaTPoint. 2021. How to Print Table in Java Using Formatter (n.d.). [Online]. Available at: https://www.javatpoint.com/how-to-print-table-in-java-using-formatter [Accessed 21 September 2021].