Criterion C-Development

Word Count:956

Table of Contents

Inheritance

Inheritance is a mechanism in object-oriented programming that allows a class to inherit the properties and methods of another class. The class CaptureAct is the best example of inheritance I found in my project. By extending CaptureActivity, the class CaptureAct is able to use all the methods and properties of CaptureActivity.

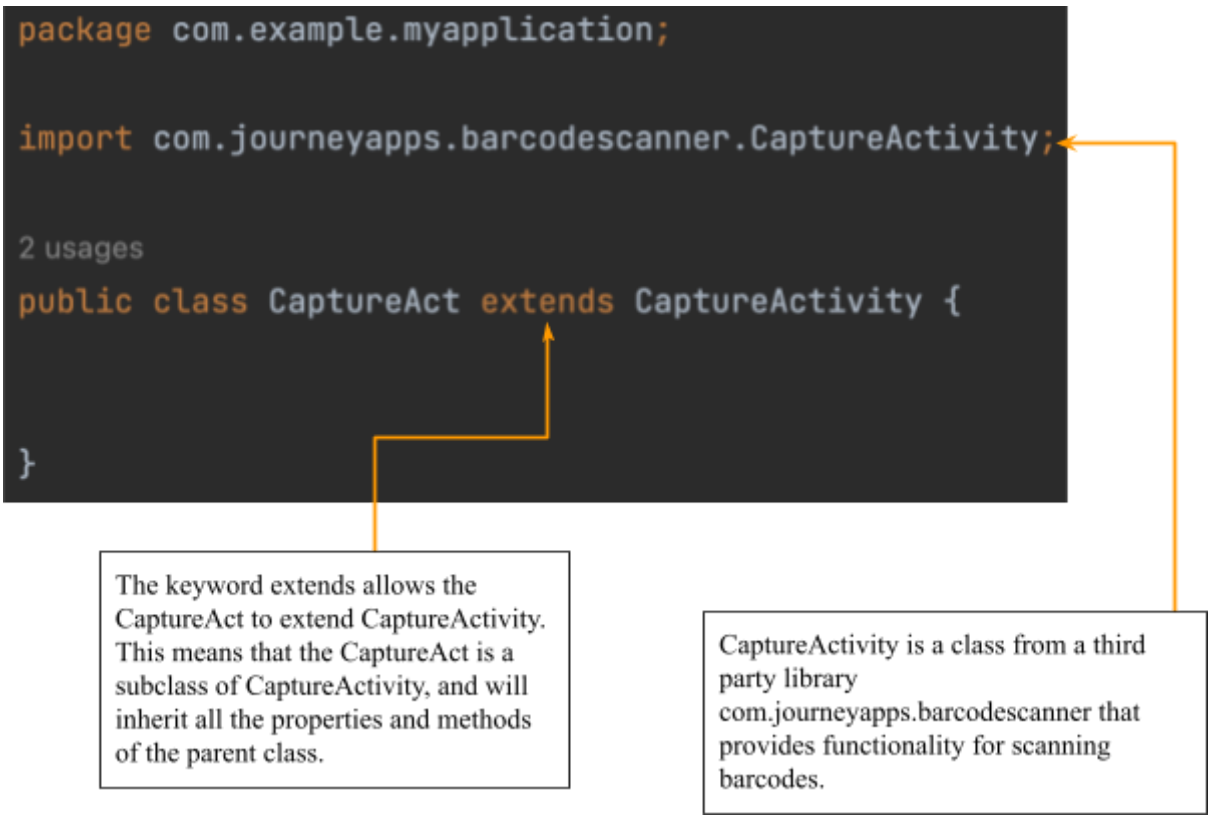

Image 1: Inheritance

In this case, it means that I am able to use all of CaptureActivity's features without having to code the class myself. The use of Inheritance here allows for code reuse and reduces the work I need to do.

Database operation

The classes use the DatabaseHelper class with the SQLite database which allows the classes to perform CRUD operations on the database.

```
// Constructor
    6 usages
    public DatabaseHelper(Context context) {
        super(context, DATABASE_NAME,  factory: null, DATABASE_VERSION);
    }
```

Image 2: DatabaseHelper Constructor

```
// Create tables
@Override
public void onCreate(SQLiteDatabase db) {
    db.execSQL(CREATE_FOOD_TABLE);
    db.execSQL(CREATE_USER_TABLE);
    db.execSQL(CREATE_LOGIN_TABLE);
    db.execSQL(CREATE_DATES_TABLE);
}
```

Creates three tables in the database for storing food, user, and date data, respectively.

Image 3: DatabaseHelper onCreate Method

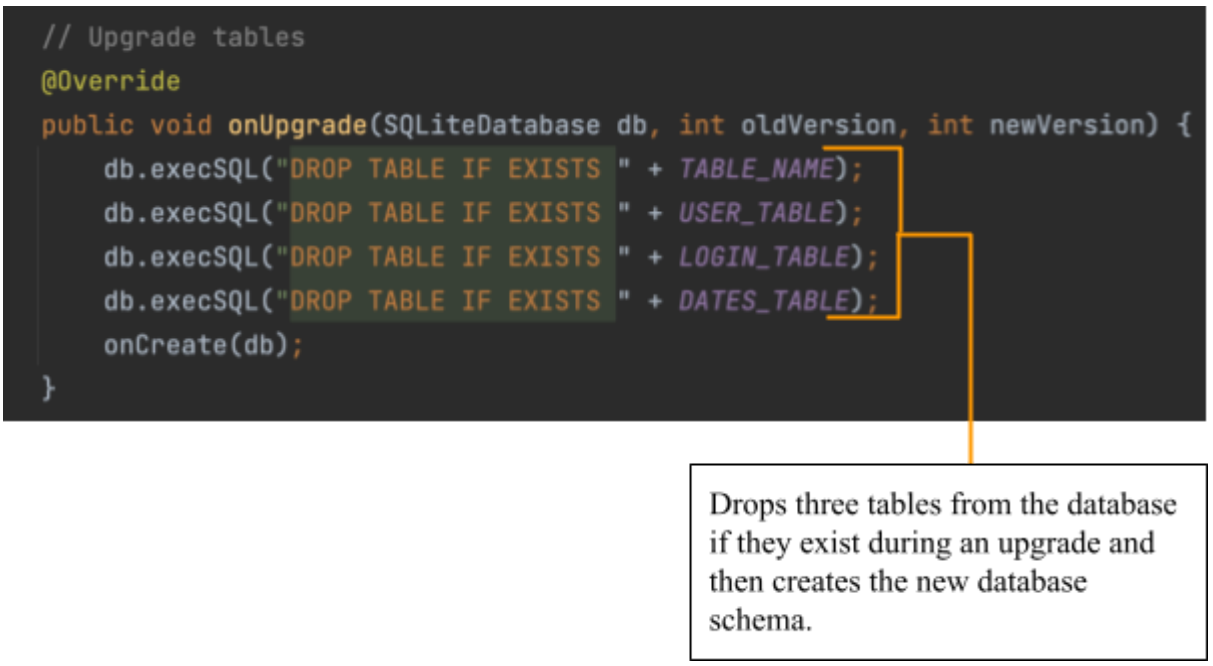

Image 4: DatabaseHelper onUpgrade Method

The DatabaseHelper class is used to perform the actual database operations, such as opening and closing the database connection, creating the table for storing user data, and inserting, updating and deleting data from the table. The UserInput class uses an instance of the DatabaseHelper class, named dbHelper, to perform these operations.

```
// creating a user object and setting the user input data to it
User user = new User();
user.setName(name);
user.setAge(age);
user.setWeight(weight);
user.setHeight(height);

// inserting the user input data into the SQLite database
boolean isInserted = dbHelper.addUserData(user);
```

Image 5: Implementation of the DatabaseHelper Class

Error handling

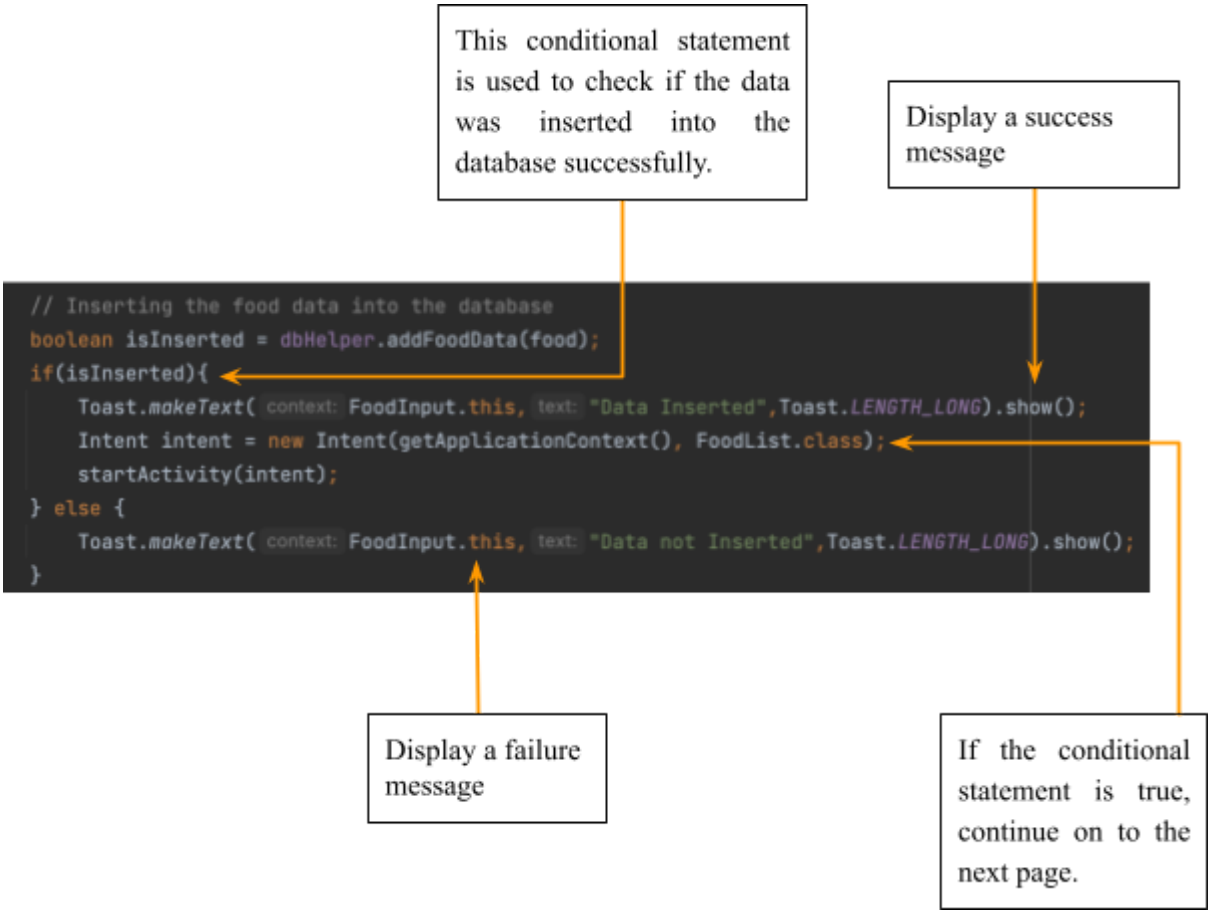

Image 6: Error handling Statements

Cursor

In Android, a Cursor is a class that provides read-write access to the results of a database query.

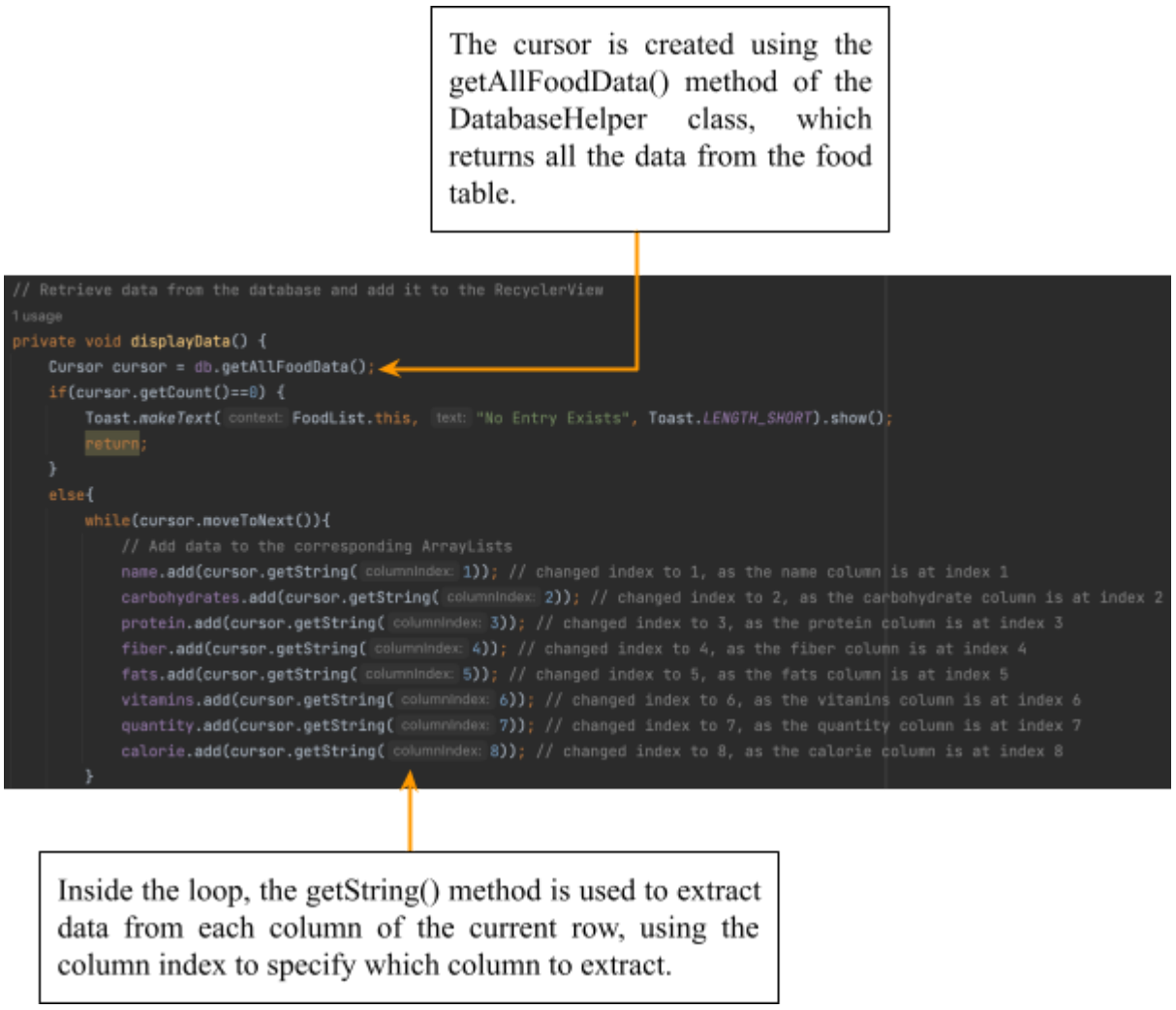

Image 7: Use of Cursor

In this class, the cursor is used to retrieve data from the database and add it to the RecyclerView.

ArrayList

An ArrayList is a class in Java that implements the List interface and provides a way to store and manipulate a collection of elements in a dynamic array. In this class, ArrayLists are used to store the nutritional values of food items.

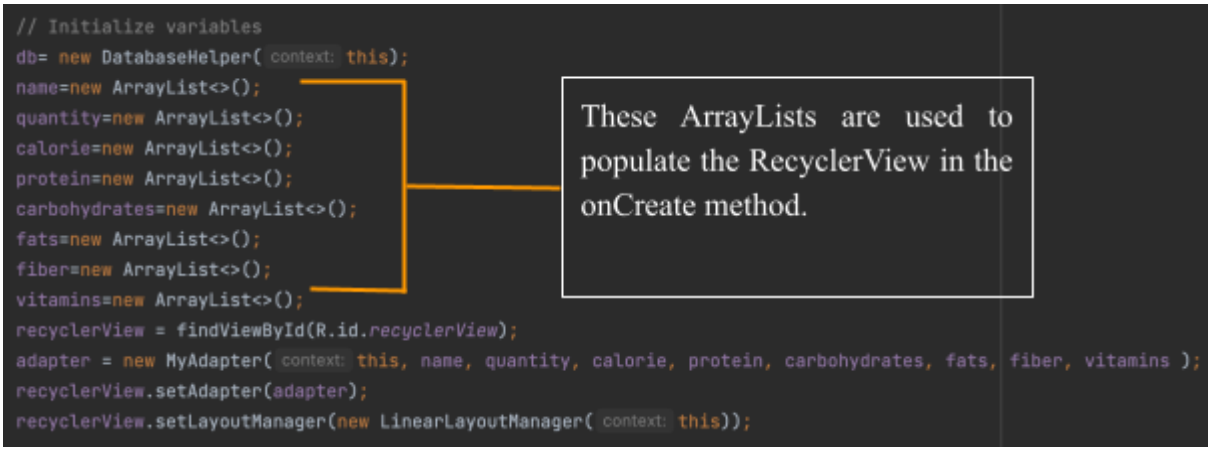

Image 8: Use of ArrayLists

ArrayLists are useful in this context because they provide a way to store a collection of items of the same type. They allow for dynamic resizing of the collection, which is important when the number of food items in the database can change.

Libraries imported

This is a list of notable libraries I used in my project and their functions.

Intent Class

This library is used to communicate between different components of an Android application. It enables the passing of data or messages between activities, services, and broadcast receivers.

```
import android.content.Intent;
```

Image 9: Intent Library

```
btnFoodInput.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        // Create an intent to navigate to the FoodInput activity
        Intent intent = new Intent(getApplicationContext(), FoodInput.class);
        // Start the FoodInput activity
        startActivity(intent);
    }
});
```

Image 10: Implementation of the Intent Library

These libraries are used for formatting and parsing dates and times in Java. They provide various methods for converting dates and times to different formats.

Simple Date Format and Calendar Classes

Image 11: SimpleDateFormat Library

```
// Add food name, date, quantity, and meal to dates table
String date = new SimpleDateFormat( pattern: "yyyy-MM-dd", Locale.getDefault()).format(Calendar.getInstance().getTime());
ContentValues values = new ContentValues();
values.put("food", foodName);
values.put("date", date);
values.put("quantity_eaten", quantity);
values.put("total_calories", totalCalories);
values.put("meal", meal);
database.insert( table: "dates_table",  nullColumnHack: null, values);
```

Image 12: Implementation of the SimpleDateFormat Library

	ID	quantity_eaten	meal	food	date	total_calories
	Fi...	Filter	Filter	Filter	Filter	Filter
1	1	2.0	lunch	pizza	2023-02-19	2.0
2	2	2.0	lunch	pizza	2023-02-19	2.0
3	3	2.0	lunch	pizza	2023-02-19	2.0
4	4	1.0	lunch	pizza	2023-02-19	1.0
5	5	1.0	lunch	pizza	2023-02-19	1.0
6	6	1.0	lunch	pizza	2023-02-19	1.0

Image 13: Date Format

The import statement import java.util.Calendar; is used to import the Calendar class from the java.util package in Java. The Calendar class provides methods for working with dates, times, and time zones, and is used to convert between the different date and time representations.

```
import java.util.Calendar;
```

Image 14: Calendar Library

This library is used for formatting and parsing decimal numbers in Java. It provides methods for converting decimal numbers to different formats.

DecimalFormat class

The import statement "import java.text.DecimalFormat;" is used to import the DecimalFormat class from the java.text package. This class provides functionality to format numbers as strings, specifying the number of digits after the decimal point and other formatting options. In the given code, it is used to format the BMI value with one decimal place before displaying it in the Toast message.

Image 15: DecimalFormat Library

```
// Format the BMI value with one decimal place
DecimalFormat decimalFormat = new DecimalFormat( pattern: "#.#");
String bmiValue = decimalFormat.format(bmi);
```

Image 16: Implementation of the DecimalFormat Library

Alert Dialog Class

This library is used to display alert dialogs in Android applications. It provides a pre-built UI for displaying messages, prompts, and notifications to the user. The AlertDialog is used in the scanCode() function to display the result of the barcode scan in an alert dialog.

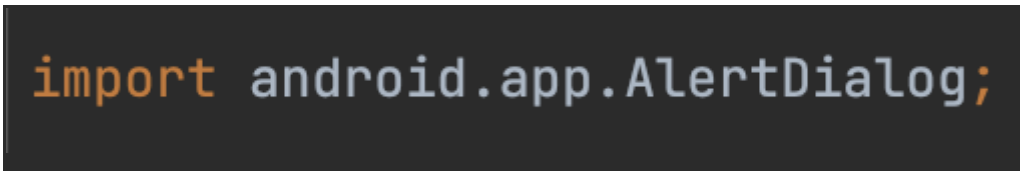

Image 17: AlertDialog Library

```
AlertDialog.Builder builder = new AlertDialog.Builder( context: FoodInput.this);
```

Image 18: Implementation of the AlertDialog Library

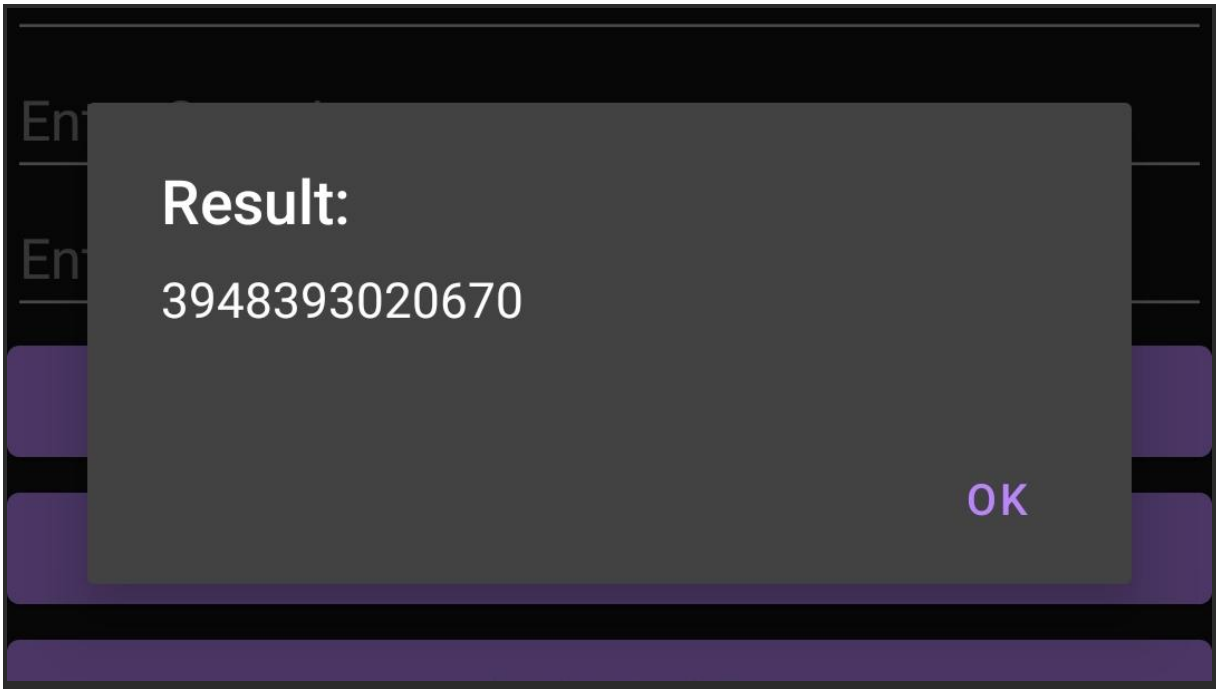

Image 19: Result of Barcode Scan

Dependencies Used

This implementation line is used to add the ZXing ("Zebra Crossing") barcode scanning library to an Android project.

com.journeyapps:zxing-android-embedded:4.3.0 is the dependency that specifies the library's name, version, and location.

```
implementation 'com.journeyapps:zxing-android-embedded:4.3.0'
```

Image 20: ZXing Library

This particular library, ZXing Android Embedded, provides a simple way to integrate barcode scanning functionality into an Android app by embedding the ZXing library, which is a popular open-source barcode scanning library.

SQL commands used

This SQL command selects the sum of the total_calories column from the dates_table table where the date column matches the date variable. The date variable is specified as a

parameter in the new String[]{date} part of the query, which makes the query safe from SQL injection attacks. The result is returned in a Cursor object named cursor.

```
Cursor cursor = database.rawQuery(
        sql: "SELECT SUM(total_calories) FROM dates_table WHERE date=?",
        new String[]{date}
);
```

Image 21: SQL Command

Searching and Sorting

Searching and sorting of data is used in the form of SQL commands.

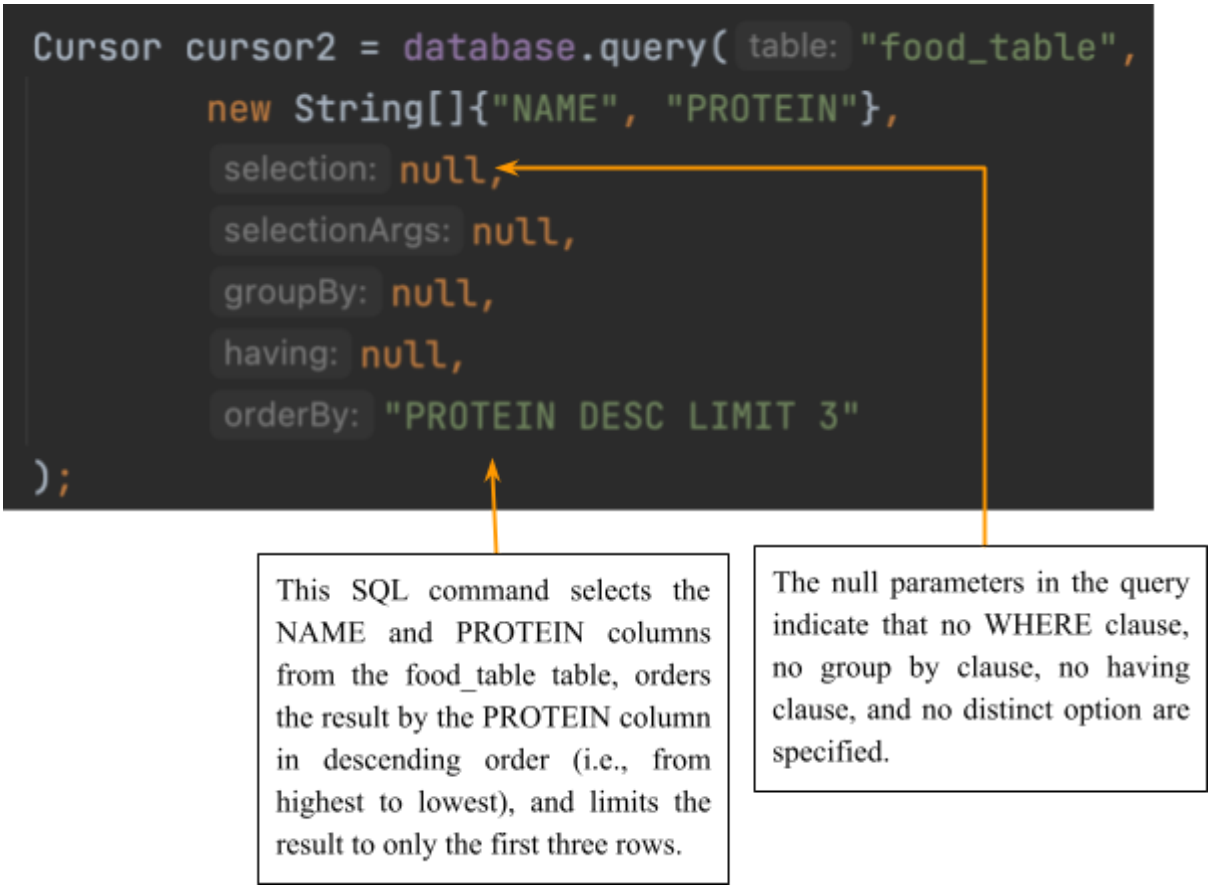

Image 22: Searching and Sorting SQL Command

```
// Store the protein-rich food names and their nutrient values in an array.
if (cursor2.moveToFirst()) {
    ProteinfoodNames = new String[3];
    int index = 0;
    do {
        String food_name = cursor2.getString(cursor2.getColumnIndex( columnName: "NAME"));
        float protein_count = cursor2.getFloat(cursor2.getColumnIndex( columnName: "PROTEIN"));
        ProteinfoodNames[index] = food_name + " (" + protein_count + " g)";
        index++;
    } while (cursor2.moveToNext());
}
```

Image 23: Implementation of the Searching and Sorting SQL Command

This SQL command selects the NAME and PROTEIN columns from the food_table table, orders the result by the PROTEIN column in descending order (i.e., from highest to lowest), and limits the result to only the first three rows. The result is returned in a Cursor object named cursor2. The null parameters in the query indicate that no WHERE clause, no group by clause, no having clause, and no distinct option are specified.

Privacy and Security

This line of code is an example of a permission declaration in an Android app's manifest file. Specifically, it is declaring that the app requires permission to access the device's camera.

```
<uses-permission android:name="android.permission.CAMERA"/>
```

Image 24: Permission Declaration

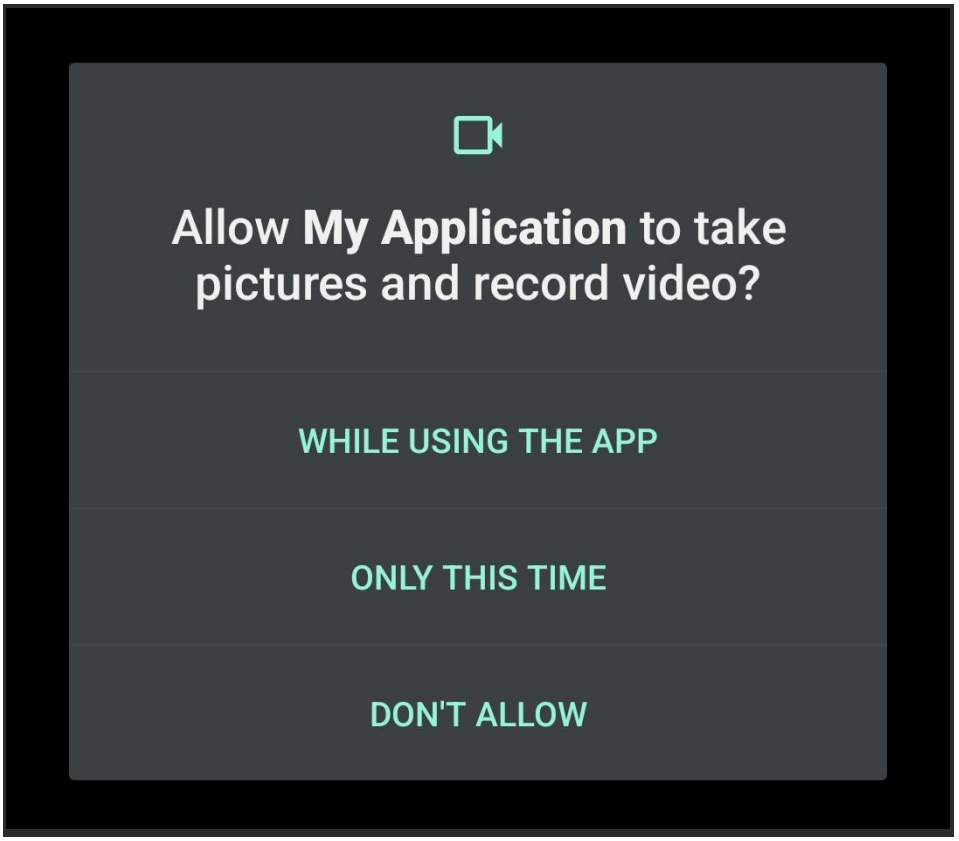

Image 25: Permission Pop up

In Android, permissions are used to protect the user's privacy and security by limiting the actions that an app can take. The Android system requires that an app declare the permissions it needs in its manifest file, and then requests the user's approval to grant those permissions when the app is installed.

QR code Scanner

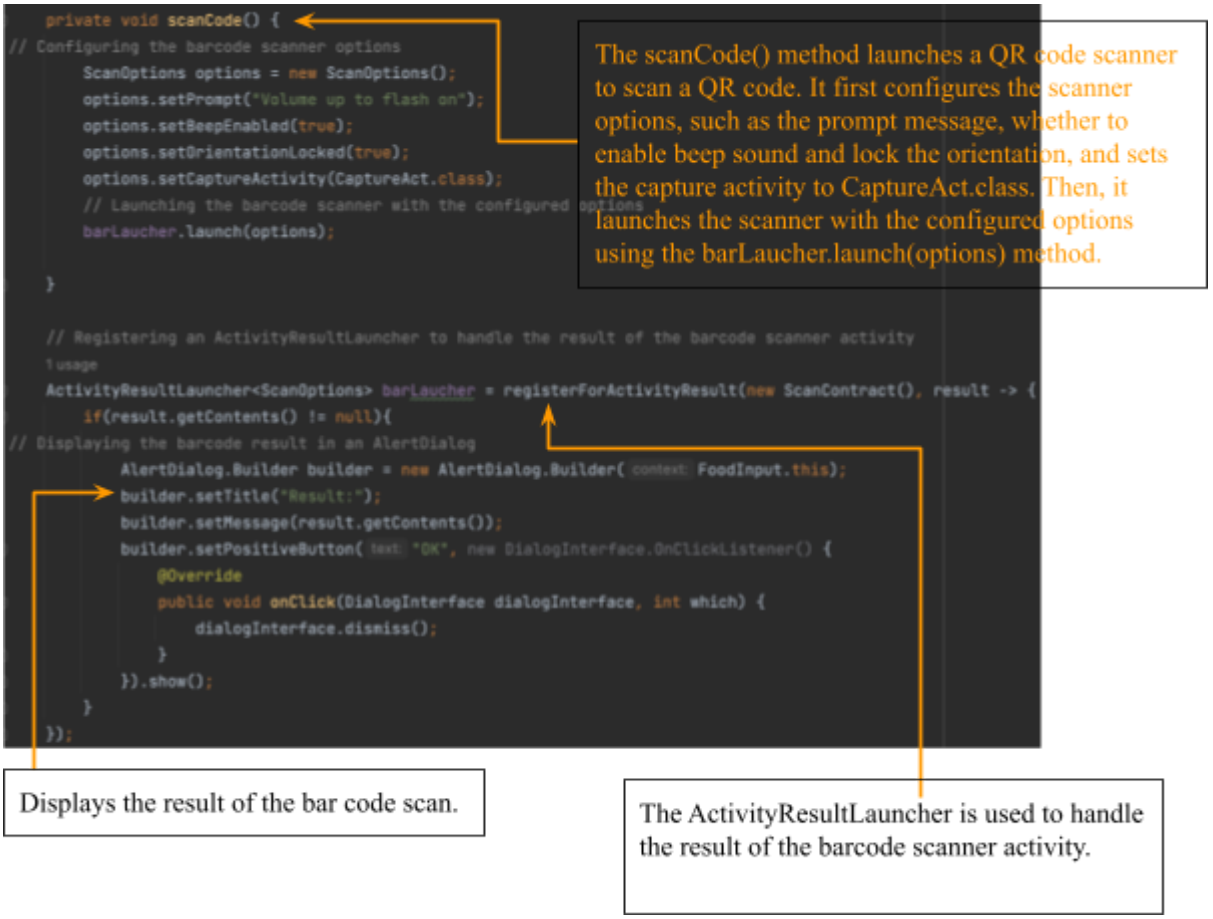

Image 26: scanCode() Method

```
// Setting an OnClickListener on the QRscanner button to launch the scanCode function
QRscanner.setOnClickListener(v -> {
    scanCode();
});
```

Image 27:Implementation of the scanCode() Method

The setOnClickListener method is used to call the scanCode method when the button is clicked.

Nested if else

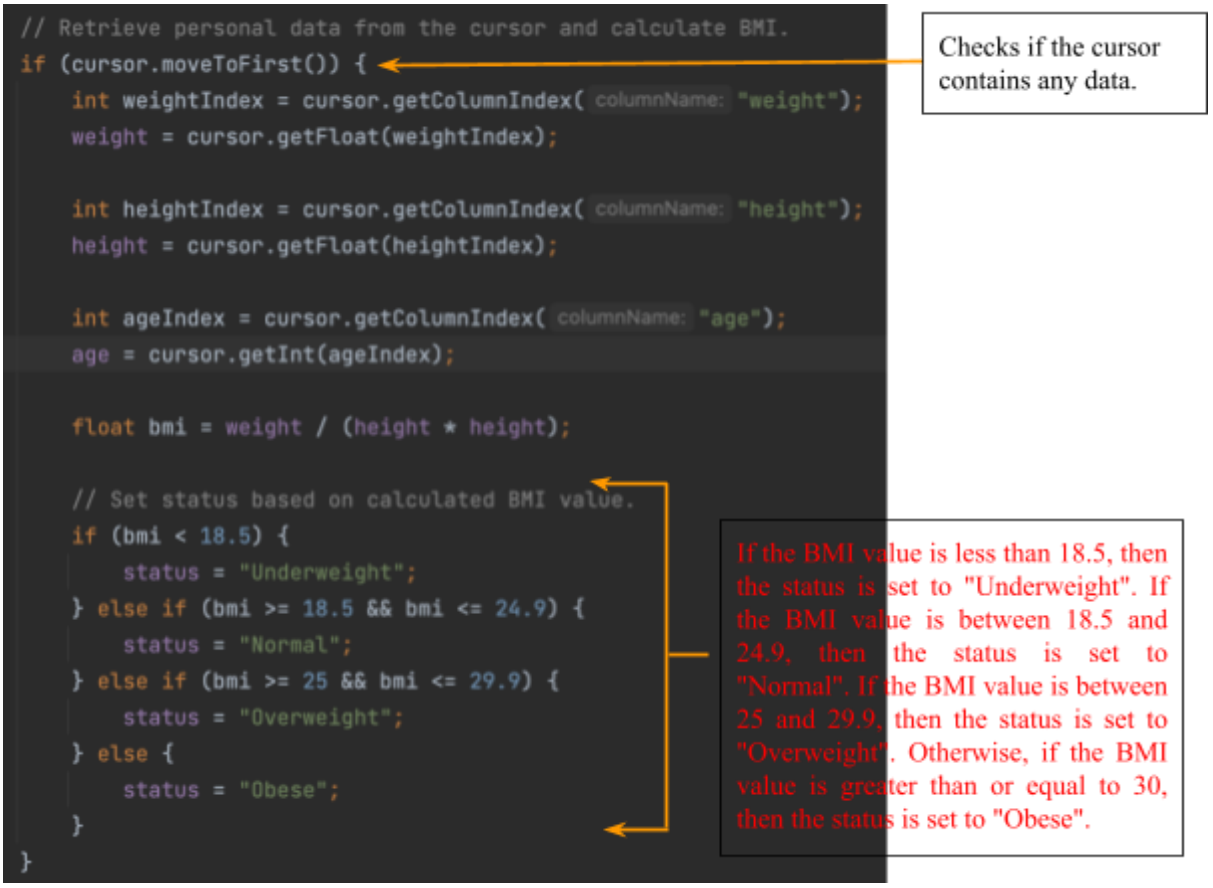

Image 28: Nested if-else Statment

The nested if-else statements are used to determine the user's status based on their BMI value calculated from the weight, height, and age values retrieved from the database.

Encapsulation

Encapsulation is a concept used to tie up data and methods that work within a single unit like a class. This allows for the construction of a safer code since access to the internal state of an object can be monitored. Read and write privileges are provided to the developer by methods called getters and setters respectively and can be accessed by any unit outside the class in which the object who is subject to modification/reading was written as long as the respective methods were implemented in the first place

Image 29:Private Variables

BMI Calculations

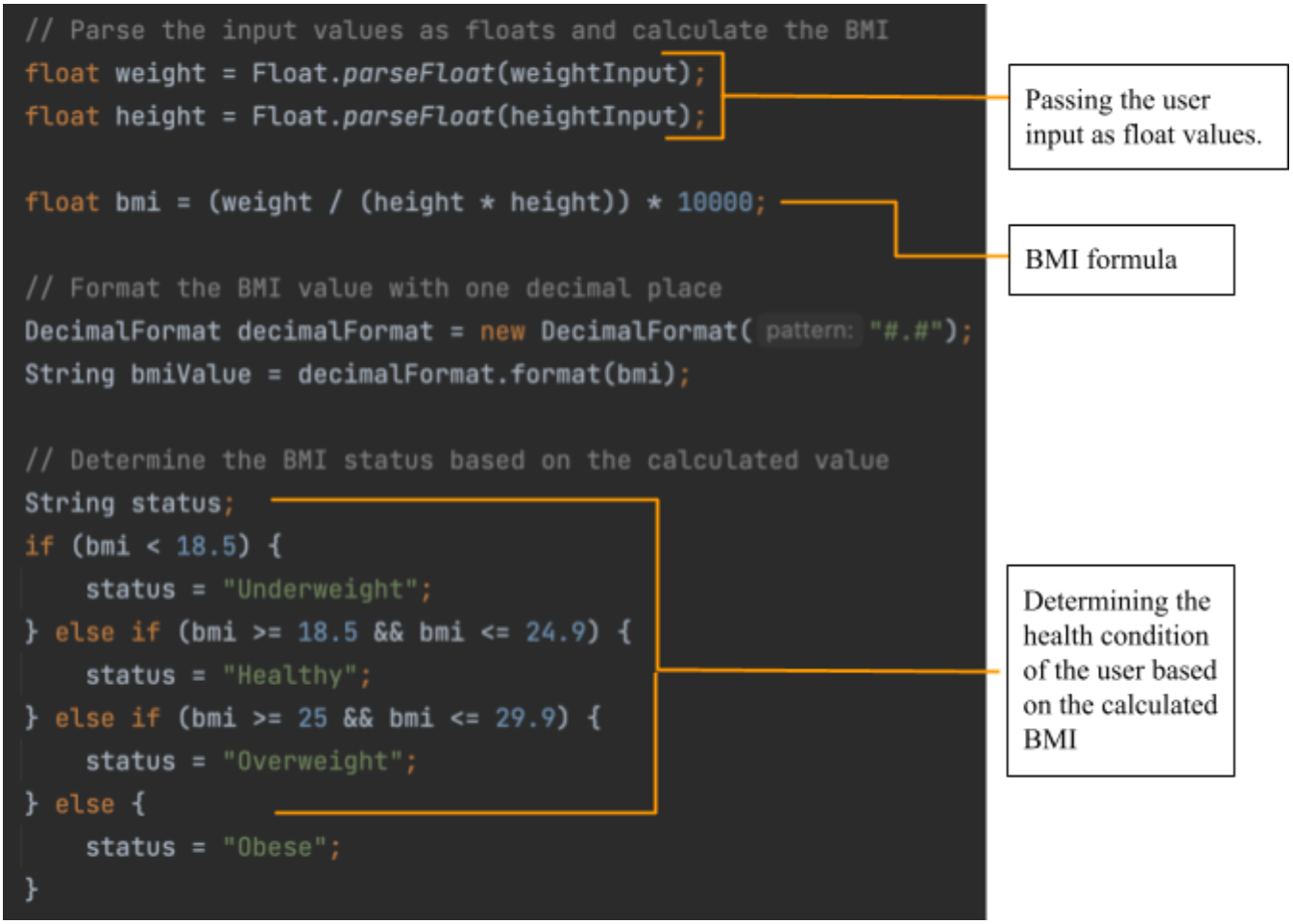

Image 30: BMI Calculations

Bibliography

"Encapsulation - Definition & Overview | Sumo Logic." *Sumo Logic*, 2022, www.sumologic.com/glossary/encapsulation/. Accessed 28 Feb. 2023.

Cambo Tutorial. "Implement Barcode QR Scanner in Android Studio Barcode Reader | Cambo Tutorial." *YouTube*, 18 Mar. 2022, www.youtube.com/watch?v=jtT60yFPelI&t=69s. Accessed 28 Feb. 2023.

freeCodeCamp.org. "SQLite Database for Android - Full Course." *YouTube*, 13 Oct. 2020, www.youtube.com/watch?v=312RhjfetP8&t=3301s. Accessed 28 Feb. 2023.

"Android Developers." *Android Developers*, 2023, developer.android.com/. Accessed 28 Feb. 2023.

CodeWithHarry. "Android Development Tutorial in Hindi." *YouTube*, 29 Aug. 2019, www.youtube.com/watch?v=qK0QNA0sMGc. Accessed 28 Feb. 2023.