FACULTY OF ENGINEERING AND APPLIED SCIENCE

Software Design and Architecture

Tutorial #4 Design Patterns

Course: SOFE 3650

Group 16

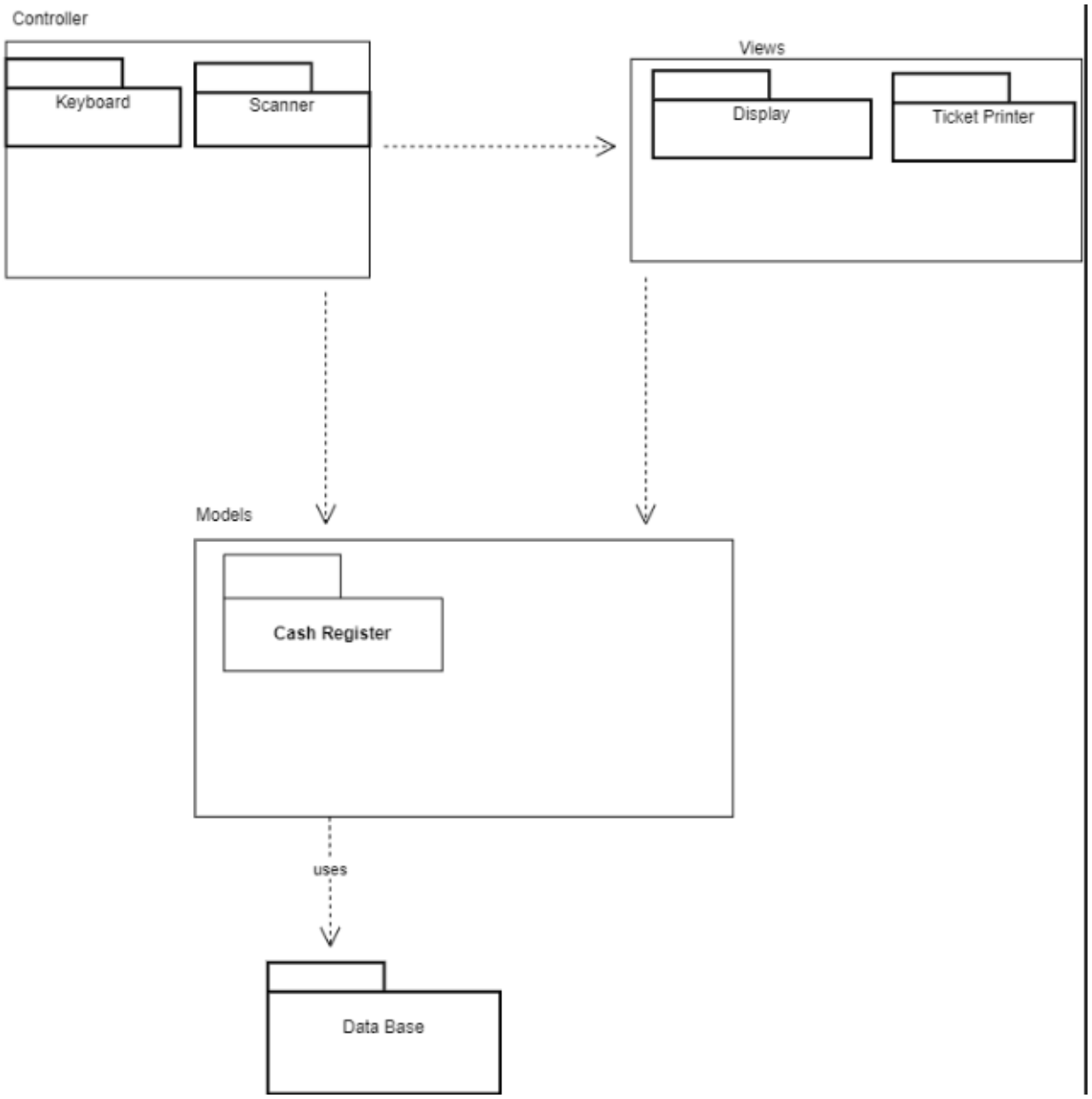
CRN: 43510

Due Date: Sept 15,2023

Name:	Student number:
Shekina Hien	100807845
Naftanan Mamo	100822222
Fernando Chan Qui	100844946

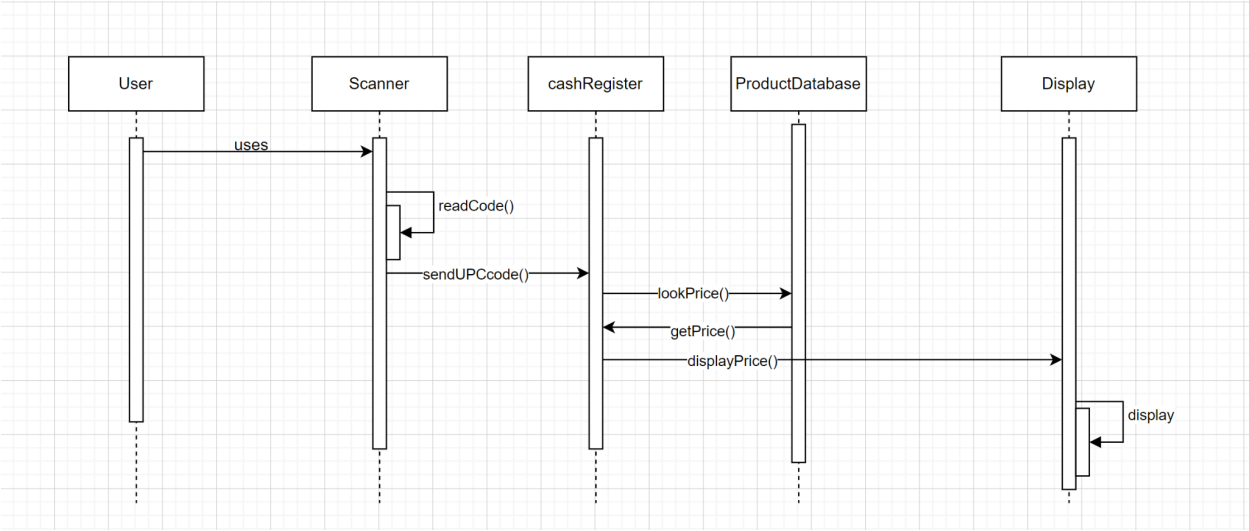
A. MVC Design Pattern

https://app.diagrams.net/#G1SkM1IdznqqAUdiHS7Y_OKGIHZnyzbaoT (LINK FOR BOTH DIAGRAMS IN A AND B)



The diagram above illustrates the relationship among the MVC (Model-View-Controller) components in this scenario. In this context, the keyboard and scanner act as controllers that users manipulate, while the ticket printer and display serve as views that the controllers interact with. Both controllers and views depend on the cash register. The core of the business logic lies within the cash register, which acts as the model, overseeing the interaction with the database.

B. Sequence Diagram



This diagram illustrates how the price of the scanned product is displayed for the user. First, the user employs the scanner to scan a product. Afterward, the scanner reads the UPC and transmits it to the cash register. Once the cash register receives the UPC code, it queries the product database to locate the price. When the product is identified with its price in the database, the price is subsequently relayed back to the cash register, which will then forward it to the display device. Finally, the display device will undertake the required operations to present the price to the user.

C. Java Code Implementation

The Java code down below shows the screenshot of the test code, and the tasks for each class.

- The ***Keyboard*** class lets the user enter a product ID.
- The ***CashRegister*** class manages a ***product database*** (represented as a HashMap in this example) and retrieves product information based on the entered product ID from a file.
- The ***Display*** class displays the product's name and price.
- The ***Product*** class stores product details.
- In the ***Test class***, a user enters a product ID, the system looks up the product, and the Display class shows the product's name and price.

Output

```
"C:\Users\fcq11\AppData\Local\Programs\Eclipse
Enter Product ID: 100844946
Product Name: Product #3
Product Price: $255.99

Process finished with exit code 0
```

Keyboard.java

```java
import java.util.Scanner;

2 usages
public class Keyboard {
    2 usages
    private final CashRegister cashRegister;

    1 usage
    public Keyboard(CashRegister cashRegister) {
        this.cashRegister = cashRegister;
    }

    1 usage
    public void enterProductID() {
        int i;
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter Product ID: ");
        int productId = scanner.nextInt();
        cashRegister.processProduct(productId);
    }
}
```

CashRegister.java

```java
public class CashRegister {
    2 usages
    private ProductDatabase productDatabase;
    3 usages
    private Display display;

    1 usage
    public CashRegister(ProductDatabase productDatabase, Display display) {
        this.productDatabase = productDatabase;
        this.display = display;
    }

    1 usage
    public void processProduct(int productId) {
        Product product = productDatabase.lookupProduct(productId);
        if (product != null) {
            display.showProductInfo(product);
        } else {
            display.showErrorMessage("Product not found.");
        }
    }
}
```

Product.java

```java
public class Product {
    2 usages
    private String name;
    2 usages
    private double price;

    4 usages
    public Product(String name, double price) {
        this.name = name;
        this.price = price;
    }

    1 usage
    public String getName() {
        return name;
    }

    1 usage
    public double getPrice() {
        return price;
    }
}
```

Display.java

```java
public class Display {
    1 usage
    public void showProductInfo(Product product) {
        System.out.println("Product Name: " + product.getName());
        System.out.println("Product Price: $" + product.getPrice());
    }

    1 usage
    public void showErrorMessage(String message) {
        System.out.println("Error: " + message);
    }
}
```

ProductDatabase.java

```java
public class ProductDatabase {
    3 usages
    private Map<Integer, Product> products;
    1 usage
    public ProductDatabase(String filename) {
        products = new HashMap<>();
        loadProductsFromFile(filename);
    }
    1 usage
    private void loadProductsFromFile(String filename) {
        try (BufferedReader reader = new BufferedReader(new FileReader(filename))) {
            String line;
            while ((line = reader.readLine()) != null) {
                String[] parts = line.split( regex: ",");
                if (parts.length == 3) {
                    int productId = Integer.parseInt(parts[0]);
                    String productName = parts[1];
                    double productPrice = Double.parseDouble(parts[2]);
                    products.put(productId, new Product(productName, productPrice));
                }
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    1 usage
    public Product lookupProduct(int productId) {
        return products.get(productId);
    }
}
```

TestCode.java

```java
public class TestCode {
    public static void main(String[] args) {
        ProductDatabase productDatabase = new ProductDatabase( filename: "product_data.txt");
        Display display = new Display();
        CashRegister cashRegister = new CashRegister(productDatabase, display);
        Keyboard keyboard = new Keyboard(cashRegister);
        keyboard.enterProductID();
    }
}
```