# UML Diagram

The UML diagram reflects the following relationships:

1.LegacyCustomer:

Represents the old system's customer data.

2.NewCustomerInterface:

Defines methods for the new system.

3.CustomerAdapter:

Implements *NewCustomerInterface* and adapts *LegacyCustomer* to the new system.

4.Main:

Demonstrates the use of the adapter.

# Design Pattern Used: Adapter Pattern

**Description:**

The Adapter Pattern bridges the gap between two incompatible interfaces, allowing objects from a legacy system (*LegacyCustomer*) to interact with a new system interface (*NewCustomerInterface*).

**Implementation:**

The *CustomerAdapter* acts as the bridge, implementing the new interface and internally adapting data from the legacy system.

This ensures smooth integration without altering existing systems.

**Advantages of the Pattern:**

- Enables reusability of existing code.

- Provides a clean interface for future extensions.

- Adheres to the Open/Closed Principle, supporting system scalability.

```
CustomerAdapterTest
+ testFullName(): void
+ testContactDetailsWithEmptyEmail(): void
```

```
Main
+ main(String[]): void
```

tests

uses

```
CustomerAdapter
- legacyCustomer: LegacyCustomer
+ getFullName(): String
+ getContactDetails(): String
```

uses

implements

```
LegacyCustomer
- firstName: String
- lastName: String
- phone: String
- email: String
+ getFirstName(): String
+ getLastName(): String
+ getPhone(): String
+ getEmail(): String
```

```
<<interface>>
NewCustomerInterface
+ getFullName(): String
+ getContactDetails(): String
```