**Exercise 11: Implementing the Dependency Injection**

**Scenario:**

You are developing a customer management application where the service class depends on a repository class. Use Dependency Injection to manage these dependencies.

1. Create a new Java Project:

* Create a new Java Project named DependencyInjectionExample.
* A Java project named DependencyInjectionExample is created in eclipse IDE.

2. Define Repository Interface:

* Create an interface **CustomerRepository** with methods like **findCustomerById()**.

public interface CustomerRepository {

String findCustomerById(int id);

void addCustomer(int id, String name);

}

* Defines methods for adding and finding customers.

3. Implement Concrete Repository:

* Create a class **CustomerRepositoryImpl** that implements **CustomerRepository**.

import java.util.HashMap;

import java.util.Map;

public class CustomerRepositoryImpl implements CustomerRepository {

private final Map<Integer, String> customerDatabase = new HashMap<>();

@Override

public String findCustomerById(int id) {

return customerDatabase.getOrDefault(id, "Customer not found");

}

@Override

public void addCustomer(int id, String name) {

customerDatabase.put(id, name);

}

}}

* Implements the interface and manages customer data using a Map.

4. Define Service Class:

* Create a class **CustomerService** that depends on **CustomerRepository**.

public class CustomerService {

private final CustomerRepository customerRepository;

public CustomerService(CustomerRepository customerRepository) {

this.customerRepository = customerRepository;

}

public String getCustomerNameById(int id) {

return customerRepository.findCustomerById(id);

}

public void addCustomer(int id, String name) {

customerRepository.addCustomer(id, name);

}

}

* Depends on CustomerRepository and provides methods to interact with it..

5. Test the Dependency Injection Implementation:

* Create a main class to demonstrate creating a **CustomerService** with **CustomerRepositoryImpl** and using it to find a customer.

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

CustomerRepositoryImpl customerRepository = new CustomerRepositoryImpl();

CustomerService customerService = new CustomerService(customerRepository);

System.out.println("Enter the number of customers to add:");

int numberOfCustomers = scanner.nextInt();

scanner.nextLine();

for (int i = 0; i < numberOfCustomers; i++) {

System.out.println("Enter Customer ID:");

int id = scanner.nextInt();

scanner.nextLine();

System.out.println("Enter Customer Name:");

String name = scanner.nextLine();

customerService.addCustomer(id, name);

System.out.println("Customer added successfully.");

}

System.out.println("Enter Customer ID to find:");

int customerId = scanner.nextInt();

String customerName = customerService.getCustomerNameById(customerId);

System.out.println("Customer Name: " + customerName);

scanner.close();

}

}}

* Demonstrates dependency injection by creating CustomerService with CustomerRepositoryImpl, adding customers, and searching for them.

Output:

