

abc RETAIL SHOP jAVA aPPLICATION

OBJECT ORIENTED DESIGN AND PROGRAMMING

Project



k200810 k210302 k240384 K240260

Anesu D. Pasipanodya Luiziane Branta Borkotur RahmanLabib Hossain Limon

Kent Institute f Australia

Contents

[Introduction 2](#_Toc174567722)

[The PURPOSE OF THE SYSTEM 2](#_Toc174567723)

[Class diagram 3](#_Toc174567724)

[Coding Process 4](#_Toc174567725)

[AbcShopDriver 5](#_Toc174567726)

[Person 7](#_Toc174567727)

[Customer 8](#_Toc174567728)

[Supplier 8](#_Toc174567729)

[Employee 9](#_Toc174567730)

[Test Cases and Documentation 11](#_Toc174567731)

# Introduction

Our Team has been approached to design a java application for a local retailer The ABC Retail Shop. It is envisioned that this java application will track the details of employees, customers, and suppliers. The CEO wants us to make a design, accompanied by documentation, and code of a simple version of the java application.

# The PURPOSE OF THE SYSTEM

The Java application that we are going to produce for The ABC Retail Shop, is to be designed to manage and track the details of all stakeholders of the shop. This includes employees, customers, and suppliers of the lolly shop. This Java application is being designed to simplify and streamline the management of information about employees, customers, and suppliers. After implementation, the application will participate in the following:

*Employee Management:*

Store and organize employee details such as names, roles, contact information, and employment status.

Allow managers to easily update employee records when needed.

*Customer Management:*

Maintain a database of customer information, including names, contact details, and purchase history.

Enable the shop to provide personalized service and track customer preferences and loyalty.

*Supplier Management:*

Keep track of supplier details, such as company names, contact information, and supplied products.

Help in managing orders, deliveries, and supplier relationships efficiently.

Overall, the application will enhance the efficiency of The ABC Retail Shop's operations by providing a centralized system to manage key information, reduce manual record-keeping, and support better decision-making with organized data.

# Class diagram

Here is a class diagram of the system we are going to design.

|  |
| --- |
|  |

Fig 1.1 Class Diagram

The Person class represents a general individual with basic attributes. Employee, Customer, and Supplier are specialized subclasses of Person. Employee adds job-specific details, Customer tracks purchase history, and Supplier includes company-related information. Each subclass inherits from Person, showcasing an ‘is-a' relationship.

# Coding Process

We started by using Eclipse to Create a new project called Assesssment 2.

|  |
| --- |
|  |

Fig 1.2 Eclipse IDE: New Java Project Setup Window

We then made a package called AbcShopSystem, with 5 “classes”; Supplier, Employee, Customer and Person and finally AbcShopDriver. AbcShopDriver is the class that we will use to run the entire package.

|  |
| --- |
|  |

Fig 1.3 Java Project Structure for "AbcShopSystem" in Eclipse IDE

## AbcShopDriver

We wrote the code for AbcShopDriver.java where we added the man method, and added all the constructors and set methods for the lolly shop. In it we created two objects of each of our child classes with different details. One object using parametrised constructor and one using setMethods(). We then added code which helps to display the details of all objects and amount of discount they receive from lolly shop in the following format.

In this code we added the two types of constructors. These will print out the parameters for the user to add their relevant information.

|  |
| --- |
|  |

Fig. 1.4 Employee Object with 2 constructors

In this code we added the two types of constructors. These will print out the parameters for the user to add their relevant information.

|  |
| --- |
|  |

Fig.1.5 Customer Object with 2 objects

In this code we added the two types of constructors. These will print out the parameters for the user to add their relevant information.

|  |
| --- |
|  |

Fig.1.6 Supplier Object with 2 objects

This section of code is the one that will display the discount after it has been calculated by the application.

|  |
| --- |
|  |

Fig.1.7 Details of Discount display

## Person

The Person class is the parent class representing an individual with basic attributes like firstName, surname, phoneNumber, and discount. It includes constructors, getters, setters, and a toString() method for object representation. Below is how we presented basic attributes that are going to be called by the Child classes, so they do not have to be repeated in each classes.

|  |
| --- |
|  |

Fig 1.7 Class Definition with Private Fields

Here are the constructors.

It will assign all the values entered by the user/stakeholder to the relevant attributes e.g. Phone number

|  |
| --- |
| A close-up of a white background  Description automatically generated |

Fig 1.8 Parameterized Constructor for Initializing Object Fields

Getters, setters, and a toString() method

The getter will provide controlled access to an objects attributes, and then the setter will return the string (words)

Representation of the object. The **toString()** provides a readable string representation of an object.

|  |  |
| --- | --- |
| Fig 1.9 Java Getters and Setters for the Person Class | Fig 1.10 Custom toString() Method in Java for Person Class |

## Customer

The Customer class extends Person, adding attributes like emailAddress and membershipLevel. It includes constructors, getters, setters, and overrides the setDiscount() method to set a 0% discount for customers.

|  |
| --- |
| Fig 1.11 Customer Class: Default and Parameterized Constructors |
| Fig 1.12 Customer Class: Constructor with Parameters and Getters/Setters |
| Fig 1.13 Overriding setDiscount Method for Customer Class |

## Supplier

The Supplier class extends Person, introducing attributes such as companyName, supplierStatus, and deliveryTime. It overrides the setDiscount() method to assign discounts based on the supplier's status (e.g., Past = 5%, Future=10%, or Active = 15%)

|  |
| --- |
| Fig 1.14 Supplier Class: Default and Parameterized Constructors |
| Fig 1.15 Supplier Class: Getters and Setters |
| Fig 1.16 Overriding setDiscount Method for Supplier Class |

## Employee

The Employee class extends Person, adding attributes like hourlyWorked and jobTitle. It overrides the setDiscount() method to calculate discounts based on the number of hours worked, ranging from 5% to 15%.  
If an employee:   
- Worked < 20 they will get a 5% discount.   
- Worked >= 21 or <= 30 they will get 10% discount.   
- Worked >30 they will get 15% discount;

|  |
| --- |
| Fig 1.17 Employee Class: Default and Parameterized Constructors |
| Fig 1.17 Employee Class: Getters, Setters, and Overridden setDiscount Method |

# Test Cases and Documentation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test ID | Objective | Preconditions | Test Steps | Expected Result | Actual Result | Status |
| TC001 | Verify that the Employee class compiles and runs without errors.  (To test if our default constructor, parameterized constructors and getters and setters are working) | Ensure the Employee class file is working as expected without compilation or runtime issues. | Employee class file is complete and included in the project.  1. Compile the Employee class.  2. Run a simple test or main method that instantiates Employee objects.  3. Check for compilation errors or runtime issues. | The Employee class should compile successfully and run without errors. | Here we ran the Employee class as a Java application It had 0 errors, and we could add values | Pass |
| TC002 | Verify that the Supplier class compiles and runs without errors.  (To test if our default constructor, parameterized constructors and getters and setters are working) | Supplier class file is complete and included in the project. | 1. Compile the Supplier class.  2. Run a simple test or main method that instantiates Supplier objects.  3. Check for compilation errors or runtime issues. | The Supplier class should compile successfully and run without errors. | This is the compilation of the code with 0 errors. It also allows us to add values. | Pass |
| TC003 | Verify that the Customer class compiles and runs without errors.  (To test if our default constructor, parameterized constructors and getters and setters are working) | Ensure the Customer class file is working as expected without compilation or runtime issues. | Customer class file is complete and included in the project.  1. Compile the Customer class.  2. Run a simple test or main method that instantiates Customer objects.  3. Check for compilation errors or runtime issues. | The Customer class should compile successfully and run without errors. | The document runs with no issues and allows us to test. | Pass |
| TC004 | Test discount calculation for Employee based on hours worked. | Verify that discount is calculated correctly for Employee objects. | Employee class is implemented and compiled.  1. Create Employee objects with different hours worked (e.g., <20, 21-30, >30).  2. Verify the discount applied. | Discounts should match the defined rules for Employee. | We entered values to see if they would receive a discount. Because employee 1 only worked 20 hours they received 0 hours discount. Everyone else received larger discounts because of their hours worked. | Pass |
| TC005 | Test discount calculation for Supplier based on status. | Verify that discount is calculated correctly for Supplier objects. | Supplier class is implemented and compiled.  1. Create Supplier objects with different statuses.  2. Verify the discount applied. | Discounts should match the defined rules for Supplier. | We entered values to see if they would receive a discount. Because the person is a customer they received 0 percent discount. | Pass |
| TC006 | Test discount calculation for Customer (should be 0). | Verify that discount is 0 for Customer objects. | Verify that discount is 0 for Customer objects. | Discounts should be 0% for all Customer objects. | We entered values to see if they would receive a discount. Discounts had been added in conjunction with delivery efficiency | Pass |
| TC007 | Verify that AbcShopDriver opens. | Ensure that the AbcShopDriver class runs correctly. | AbcShopDriver class is implemented and compiled. | Verify that AbcShopDriver opens. | The code compiled of the code with 0 errors | Pass |
| TC008 | Verify that Person class works within AbcShopDriver. | Ensure that the Person class functionalities are accessible and working as expected within the driver. | AbcShopDriver and Person classes are implemented and compiled. | The Person class should work correctly within the AbcShopDriver application. | The code compiled of the code with 0 errors | Pass |

***TC001***

*Here we ran the Employee class as a Java application It had 0 errors and we could add values.*

|  |
| --- |
|  |

**This is the output of the Employee class**

|  |
| --- |
|  |

***TC002***

*This is the compilation of the code with 0 errors. It also allows us to add values it shows 0 problems in its description.*

|  |
| --- |
|  |

**This is the output of the Supplier class**

|  |
| --- |
|  |

***TC003***

*The document runs with no issues and allows us to test.*

|  |
| --- |
|  |

**This is the output of the Customer class**

|  |
| --- |
|  |

***TC004***

*We entered values to see if they would receive a discount. Because employee 1 only worked 20 hours they received 0 % discount. Everyone else received larger discounts as a result of their hours worked.*

|  |
| --- |
|  |

***TC005***

*We entered values to see if they would receive a discount. Discounts were dispersed efficiently.*

|  |
| --- |
|  |

***TC006***

*We entered values to see if a supplier would receive a discount based on their attributes. Because the Person in question is a customer they received 0 percent discount.*

|  |
| --- |
|  |

***TC007***

*This is the compilation of the code with 0 errors*

|  |
| --- |
|  |

***TC008***

*This is the compilation of the code with 0 errors*

|  |
| --- |
|  |