Basic Bill Generator IN JAVA

Dissertation submitted in fulfilment of the requirements for the Degree of

Bachelor of Technology

in

Computer Science and Engineering

By

DAULAT KUMAR

Registration number: 12203838

Supervisor

MR. SHUBHAM SHARMA - 64339

School of Computer Science and Engineering

Lovely Professional University

Phagwara, Punjab (India)

Month: October, Year 2023

Abstract

Basic Bill Generator is a console application designed to help users manage their tasks, deadlines, and priorities efficiently. This application has been developed in Java and operates without the use of a traditional database system. It utilizes data structures and algorithms to provide users with a simple and effective way to keep track of their tasks and deadlines, ensuring a streamlined and organized approach to task management.

Introduction

The Bill Generator Project is a comprehensive software application designed to simplify the billing process for businesses. It leverages cutting-edge technology to automate the generation of bills and invoices, ensuring accuracy and efficiency in financial transactions. The project encompasses various features and functionalities to cater to the diverse needs of businesses across different sectors.

Related Work

Studying how taxation systems work, especially in countries with complex tax structures, provides insights into compliance requirements. Understanding how tax-related information is processed and integrated into billing systems is crucial for any comprehensive bill generating project.

Problem Statement

WorkItWise aims to tackle the persistent problem of maintaining consistency in exercise routines and a lack of insights into personal fitness progress. Individuals often struggle to stay motivated and informed about their exercise journey. Fitness trainers require efficient tools to manage multiple clients' data effectively. WorkItWise addresses these challenges with its user-centric approach and multifaceted functionality.

Approach

Data Storage

The primary challenge in this project is data storage without a traditional database. We opted to use in-memory data structures like ArrayLists and HashMaps to store and manage task-related data.

Algorithms

Hashing :

Purpose: Hashing can be used to create fast data lookup tables, such as mapping product IDs to their details.

Graph Algorithms:

Purpose: Graph algorithms can be used in scenarios where you have complex relationships between entities, for instance, finding the shortest path between different customers.

Tree Structures:

- **Purpose:** Tree structures can be useful for hierarchical data representation, for example, categorizing products into different categories.
- **Example:** Binary Search Trees (BST) or Balanced Binary Search Trees (like AVL trees) can be used for efficient search and retrieval operations.

String Matching Algorithms:

- **Purpose:** String matching algorithms can be applied for tasks like searching for a product by its name or description.
- **Example:** Algorithms like Knuth-Morris-Pratt (KMP) or Boyer-Moore can be used for efficient string searching.

Functionalities:

- **Item Input:** Users can input item names, prices, and quantities.
- **Cost Calculation:** The system calculates the total cost of items based on their prices and quantities.
- **Tax Calculation:** A fixed tax rate (10%) is applied to the total cost to compute the tax amount.
- **Final Bill Generation:** The system generates a final bill report, displaying the total cost, tax amount, and the overall amount to be paid.

Implementation Details:

- **Java Language:** The project is implemented in Java, ensuring platform independence and ease of development.
- **User Interaction:** The system interacts with users through the command line, providing a straightforward input mechanism.
- **Modularity:** The code is organized into functions, ensuring readability, maintainability, and ease of future enhancements.
- **Error Handling:** The system incorporates error handling to manage invalid inputs, ensuring a seamless user experience.

For generating the shopping bill, we require the product ID, name, quantity, price per item, and total price of the product, and grand total. Besides the product details, we can also add some other details like date of purchase, discount (if any), SGST, CGST, POS name, address, contact details, bill number, etc.

In the following Java program, we have created two classes first is the **Product** class and the second is the **ShoppingBill** class.

```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Calendar;
class Product
    {
        // properties
        private String id;
        private String pname;
        private int qty;
        private double price;
        private double totalPrice;

        // constructor
        Product(String id, String pname, int qty, double price, double totalPric
 e)
{
            this.id=id;
          this.pname = pname;
            this.qty = qty;
            this.price = price;
            this.totalPrice = totalPrice;
        }
            // getter methods
```

```
public String getId()
        {
       return id;
        }
        public String getPname()
        {
           return pname;
        }
        public int getQty()
        {
           return qty;
        }
        public double getPrice()
        {
           return price;
    }
        public double getTotalPrice()
        {
           return totalPrice;
        }
    //displayFormat
       public static void displayFormat()
        {
```

```
System.out.format(" welcome on my shop");
                    System.out.print("\nProduct ID \t\tName\t\tQuantity\t\tRate \t\t\t\to
    tal Price\n");
                    System.out.format(" \n");
                }

                // display
                public void display()
                {
                    System.out.format("   %9s    %5d       %9.2f       %14.2f\n" ,id, pname,
    qty, price, totalPrice);
                }
    }
    public class ShoppingBill
    {
        public static void main(String args[])
          {
            // variables
          String id = null;
              String productName = null;
      int quantity = 0;
           double price = 0.0;
           double totalPrice = 0.0;
       double overAllPrice = 0.0;
              double cgst, sgst, subtotal=0.0, discount=0.0;
              char choice = '\0';
           System.out.println("\t\t\t\t----------------------Invoice-----------------");
              System.out.println("\t\t\t\t\t "+"  "+"Metro Mart Grocery Shop");
              System.out.println("\t\t\t\t\t3/98 Mecrobertganj New Mumbai");
              System.out.println("\t\t\t\t\t"  +"    " +"Opposite Metro Walk");
              System.out.println("GSTIN: 03AWBPP8756K592"+"\t\t\t\t\t\t\tContact:
    +91) 9988776655");
           //format of date and time
```

```
            SimpleDateFormat formatter = new SimpleDateFormat("dd/MM/yyyy
HH:mm:ss");
            Date date = new Date();
            Calendar calendar = Calendar.getInstance();
            String[] days = new String[] { "Sunday", "Monday", "Tuesday", "Wedned
ay", "Thursday", "Friday", "Saturday" };
            //prints current date and time
            System.out.println("Date: "+formatter.format(date)+"  "+days[calendarg
et(Calendar.DAY_OF_WEEK) - 1]+"\t\t\t\t\t\t (+91) 9998887770");
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter Customer Name: ");
            String customername=scan.nextLine();
            //create Scanner class object
            //creating an ArrayList to store the product
        List<Product> product = new ArrayList<Product>();
            do
              {
                // read input values
                System.out.println("Enter the product details: ");
                System.out.print("Product ID: ");
                id = scan.nextLine();
                System.out.print("Product Name: ");
                productName = scan.nextLine();
                System.out.print("Quantity: ");
                quantity = scan.nextInt();
                System.out.print("Price (per unit): ");
                price = scan.nextDouble();
                //calculate total price for a particular product
                totalPrice = price * quantity;
                //calculates overall price
                overAllPrice = overAllPrice + totalPrice;
                //creates Product class object and add it to the List
                product.add( new Product(id, productName, quantity, price, totalPi
ce) );
```

```
                // ask for continue shopping?
                System.out.print("Want to add more items? (y or n): ");
            //reads a character Y or N
                choice = scan.next().charAt(0);
                //read remaining characters, don't store (no use)
            scan.nextLine();
             }
        while (choice == 'y' || choice == 'Y');
        //display all product with its properties
        Product.displayFormat();
        for (Product p : product)
        {
           p.display();
}
    //price calculation
        System.out.println("Total Amount (Rs.) " +overAllPrice);
        //calculating discount
        discount = overAllPrice*2/100;
    System.out.println("Discount (Rs.) " +discount);
        //total amount after discount
        subtotal = overAllPrice-discount;
        System.out.println("         Subtotal "+subtotal);
        //calculating tax
```


```
System.out.println("Discount (Rs.) " +discount);
//total amount after discount
subtotal = overAllPrice-discount;
System.out.println("         Subtotal "+subtotal);
//calculating tax
```

```
            sgst=overAllPrice*12/100;
            System.out.println("SGST (%) "+sgst);
            cgst=overAllPrice*12/100;
            System.out.println("CGST (%) "+cgst);
            //calculating amount to be paid by buyer
            System.out.println("Invoice Total " +(subtotal+cgst+s
gst));
            System.out.println("Thank You for Shopping");
            System.out.println("Visit Again");
            // close Scanner
        scan.close();
        }
```

```
<terminated> ShoppingBill [Java Application] C:\Program Files\Java\jdk1.8.0_05\bin\javaw.exe (Nov 23, 2021, 11:56:25 PM)
                                   -------------------Invoice----------------
                                         Metro Mart Grocery Shop
                                      3/98 Mecrobertganj New Mumbai
                                          Opposite Metro Walk
GSTIN: 03AWBPP8756K592                                                Contact: (+91) 9988776655
Date: 23/11/2021 23:56:25  Tuesday                                             (+91) 9998887770
Enter Customer Name: DEVID
Enter the product details:
Product ID: 321
Product Name: Rice
Quantity: 4
Price (per unit): 40
Want to add more items? (y or n): y
Enter the product details:
Product ID: 764
Product Name: Salt
Quantity: 2
Price (per unit): 20
Want to add more items? (y or n): y
Enter the product details:
Product ID: 984
Product Name: Cooking Oil
Quantity: 1
Price (per unit): 180
Want to add more items? (y or n): y
Enter the product details:
Product ID: 109
Product Name: Sugar
Quantity: 5
Price (per unit): 44
Want to add more items? (y or n): y
Enter the product details:
Product ID: 098
Product Name: Eggs
Quantity: 12
Price (per unit): 7
Want to add more items? (y or n): y
Enter the product details:
Product ID: 554
Product Name: Butter
Quantity: 2
Price (per unit): 50
Want to add more items? (y or n): y
Enter the product details:
Product ID: 781
Product Name: Tea
Quantity: 1
Price (per unit): 360
Want to add more items? (y or n): y
Enter the product details:
Product ID: 671
Product Name: Cheese
Quantity: 70
Price (per unit): 50
Want to add more items? (y or n): n
----------------------------------------------------------------------------------------------------------
Product ID          Name           Quantity            Rate                              Total Price
----------------------------------------------------------------------------------------------------------
  321               Rice           4                   40.00                             160.00
  764               Salt           2                   20.00                             40.00
  984               Cooking Oil      1                  180.00                            180.00
  109               Sugar          5                   44.00                             220.00
```

Conclusion

In the dynamic landscape of business, a reliable and efficient billing system is paramount. The Bill Generator System stands as a fundamental solution that simplifies financial transactions, enhances accuracy, and fosters professionalism within enterprises. As we reflect on the core attributes and impact of a Bill Generator System, several key points come to light:

1. Accuracy and Efficiency:

- The Bill Generator System ensures accurate calculations of invoices, eliminating errors associated with manual processes. Automation leads to efficiency, saving time and resources for businesses.

2. Professionalism and Brand Image:

- Customizable invoice templates lend a professional appearance to business transactions. Consistent branding and structured invoices enhance the credibility and trustworthiness of the business in the eyes of clients and partners.