

CST8284

Object-Oriented Programming (in Java)

Midterm Test: Part 2 Possible Solution

Professor: Rex Woollard

Part 1: 24 Marks

Part 2: 29 Marks

Name: _____

Programming Problem (25 marks)

Write a working Java program that manages monthly billing information for a cell phone company's customers. In crafting your solution, you can use the *Input* class that is supplied (the *JavaDoc* API is shown below). The class containing *main()* is also supplied, and your solution must work properly with my *main()*. Your solution must be complete except for the following, which are not required:

- Since the class containing *main()* is supplied, you do not need to write it out again.
- You do not need to add any *import* statements.
- You do not need to add any comments.

The following will be required:

- Variable names, class names and method names adhere to the Java standard.

The following are optional approaches:

- The *PhoneCompany* class will hold a collection of *Customer* objects. You can choose to implement the collection with either an array or with one of the Java *Collection* classes (such as *ArrayList*). There are no grading benefits or penalties associated with either choice. If you use an array, remember to ensure that the addition of customers does not exceed the capacity of the array.

Overview of the Requirements

All customers are charged a base rate for their phone plan, but they will face additional charges based on their choice of plan. There are two types of plans:

- *Per Minute Plan*: In addition to the *base rate*, customers will pay a per-minute charge of 25 cents.
- *Flat Rate Plan*: In addition to the *base rate*, customers will pay an extra premium fee which can vary for each customer.

Details: Hierarchy of Phone Classes

There are 3 class: *Phone*, *PerMinutePhone* and *FlatRatePhone*.

The *Phone* Class

The *Phone* class will be *abstract*. It will have the following instance variables all of which must be *private*:

- *phoneNumber*: holds the phone number in a *String* object. There is no need for any validation of this field during input.
- *baseRate*: holds a floating-point number. This value is entered during keyboard input.
- *taxAmt*: holds the calculated tax. This value is calculated when all customer bills are displayed.

The *Phone* class will also hold a constant for the HST tax (which is now going to be 13%).

The *PerMinutePhone* and *FlatRatePhone* Classes

Both *PerMinutePhone* and *FlatRatePhone* are subclasses of *Phone*. The *PerMinutePhone* class has the following *private*

instance variable:

- *minutes*: holds a floating-point number which represents the total number of minutes used during that month.

The *FlatRatePhone* class has the following *private* instance variable:

- *flatRatePremium*: holds the additional cost of having a flat-rate plan.

The Class Methods

The organization of methods is largely up to you, with the following requirements. You must implement overridden methods (virtual methods) to enter values from the keyboard, and to calculate the before-tax bill total. The tax calculation can be left as an exclusive responsibility of the *Phone* class.

Details: Customer Class

The *Customer* class will hold the customer's name in a *String* object, and have a reference to a *Phone* object. It will require suitable methods to perform:

- input
- processing / calculations
- display of output

Details: PhoneCompany Class

As you can see in *main()*, the *PhoneCompany* class is used to create an object, then the *addCustomer()* and *calcBills()* methods are called as a result of a user's menu choice.

The *addCustomer()* method will create a new *Customer* object and create the correct type of *Phone* object based on the user's choice. The *PhoneCompany* class must ensure that the addition of new customers will fit in the collection that is used to manage all the added customers.

The *calcBills()* method will determine the billing total for each customer, then display the results.

The following *main()* routine has already been completed. Your program solution must work with this.

Methods Available in Input Class

Memory Map (4 marks)

On a separate sheet of paper, draw a representative memory map that shows the organization of objects in memory. Your memory map must show the sample data that was entered during the sample program execution shown on the right.

The following screen capture demonstrates a sample run of the program. You can see that two customers have been added, then the bills are displayed for those two customers. Of course, more customers could have been added.

CST8284: Object-Oriented Programming (in Java)

Possible Solution

Class Supplied in Midterm Question

```
public class MidFall2013 {
    public static void main(String[] args) {
        PhoneCompany phoneCo = new PhoneCompany();
        System.out.println(phoneCo);

        int menuChoice;
        do {
            menuChoice = Input.getInt("n1. Add customer"
                                     + "n2. Display Customer Bills"
                                     + "n0. Quit: "
                                     + "nEnter Choice:");

            switch (menuChoice) {
                case 1: phoneCo.addCustomer(); break;
                case 2: phoneCo.calcBills(); break;
            }
        } while (menuChoice != 0);
    }
}
```

Class PhoneCompany

```
public class PhoneCompany {
    private List<Customer> database;

    public PhoneCompany() {
        database = new ArrayList<Customer>();
    }

    public void addCustomer() {
        Customer newCustomer = new Customer();
        newCustomer.inputAllFields();
        database.add(newCustomer);
    }

    public void calcBills() {
        System.out.println(this);
        double totalBill = 0.0;
        for (Customer iter : database) {
            iter.calcBill();
            System.out.println(iter);
            totalBill += iter.getBillTotal();
        }
        System.out.printf("TOTAL: $%5.2f\n", totalBill);
    }

    @Override
    public String toString() {
        return "AirHead Phone Company";
    }
}
```

Class Phone

```
public abstract class Phone {
    private static double HSTRATE = 0.13;

    private String phoneNo;
    private double baseRate;
    private double taxAmt;

    public void inputAllFields() {
        phoneNo = Input.getString("Enter Phone Number:");
        baseRate = Input.getDouble("Base Rate:", 0.0, 100.0);
    }

    public double getTaxAmt() { return taxAmt; }
    public String getPhoneNo() { return phoneNo; }

    public void calcTaxAmt() {
        taxAmt = getBillAmt() * HSTRATE;
    }

    public double getBillAmt() { return baseRate; }

    @Override
    public String toString() {
        return String.format(" Number:%12s\n Amt:$%6.2f\n Tax:$%5.2f\n TOTAL:$%6.2f\n ",
            phoneNo, getBillAmt(), getTaxAmt(), getBillAmt()+getTaxAmt());
    }
}
```

Class PerMinutePhone

```
public class PerMinutePhone extends Phone {
    private static double RATEPERMINUTE = 0.25;
    private double minutes;

    public void inputAllFields() {
        super.inputAllFields();
        minutes = Input.getDouble("Minutes Used:", 0.0, 500.0);
    }

    @Override
    public double getBillAmt() {
        return super.getBillAmt() + minutes*RATEPERMINUTE;
    }

    @Override
    public String toString() {
        return super.toString() + String.format(" PER MINUTE PLAN: Rate:%4.2f Minutes
            Used:%5.1f", RATEPERMINUTE, minutes);
    }
}
```

Class *FlatRatePhone*

```
public class FlatRatePhone extends Phone {
    private double flatRatePremium;

    public FlatRatePhone(double flatRatePremium) {
        this.flatRatePremium = flatRatePremium;
    }

    public FlatRatePhone() {}

    public void inputAllFields() {
        super.inputAllFields();
        flatRatePremium = Input.getDouble("Flat Rate Premium:", 0.0, 100.0);
    }

    @Override
    public double getBillAmt() {
        return super.getBillAmt() + flatRatePremium;
    }

    @Override
    public String toString() {
        return super.toString() + String.format(" FLAT RATE PLAN: Premium:%5.2f",
flatRatePremium);
    }
}
```

Class *Customer*

```
public class Customer {
    private String name;
    private Phone phone;

    public String getName() return name; }
    public Phone getPhone() { return phone; }

    public void setName(String name) { this.name = name; }

    public void inputAllFields() {
        name = Input.getString("\nEnter name: ");

        switch (Input.getInt("1:FLAT RATE 2:PER-MINUTE Enter Choice:")){
            case 1: phone = new FlatRatePhone(); break;
            case 2: phone = new PerMinutePhone(); break;
        }
        phone.inputAllFields();
    }

    public void calcBill() {
        phone.getBillAmt();
        phone.calcTaxAmt();
    }

    @Override
    public String toString() {
        return String.format("%8s %s ", name, phone);
    }

    public double getBillTotal() {
        return (phone.getBillAmt() + phone.getTaxAmt());
    }
}
```