475 : Software Engineering for Industry - Coursework 2

AcmeTelecom (Coursework should be done in groups of 4-5)

You are working at Acme Telecom. Having successfully completed your previous project, you are asked to look at an important legacy codebase which is at the core of Acme's business. This application is currently running the pricing and billing calculations for all of Acme's mobile phone customers. Mobile customers generate a lot of revenue for Acme, so this system is vital to the business.

You are called in to a meeting with Stephen, the head of engineering, who explains that due to a change in regulations on the industry, a change needs to be made to the way that customers' bills are calculated.

Currently if a customer begins a call during peak time, but the call overlaps until off-peak time, then the whole call is billed at the the peak rate.

Similarly, if a customer begins a call during the off-peak period, but the call doesn't finish until after peak time has started, then the whole call is billed as being a peak time call.

Any call that overlaps the peak time period is charged as a peak call.

The regulators have specified that to be fairer to customers, mobile operators must only charge the peak rate for the duration of the call that is actually during the advertised peak period. If peak time ends at 7pm, and a call goes from 6:50pm to 7:20pm, ten minutes should be billed at peak rate, but the remaining twenty minutes must be billed at the off-peak rate.

Stephen doesn't seem very happy about the whole business, presumably as being fairer to the customer will mean a reduction in Acme's revenues. He repeats that it is an important project, and that apart from this change the system should continue to behave as is does now. As you leave his office, he adds that he is sure you will make a good job of it.

When you get back to your desk, you have an email from Stephen, saying that although everyone who developed the billing system originally has since left the company (somehow he had neglected to mention this in the meeting), he has been able to recover the source code. Attached to the email is a tar ball of what Stephen assures you is the latest code.

Examining the codebase you are given, you find the code shown in the Appendix (you can also download the code from http://www.doc.ic.ac.uk/~rbc/475/coursework/AcmeTelecom.zip). The system doesn't appear to have any test code, but it is written in Java, so at least you can understand it. You set to work...

The Code

You can find the code you receive in the Appendix of this document, but you can download it from the URL noted above.

Included in the download is acme-externals.jar which includes binary versions of some of the components that you will need to build and run the system. You don't have the source code for these components though as they are built and maintained by other departments.

As a starting point, you may want to try running the system as it is now. For example, you could implement something like the following:

```
public class Runner {

    public static void main(String[] args) throws Exception {

        System.out.println("Running...");
        BillingSystem billingSystem = new BillingSystem();

        billingSystem.callInitiated("447722113434", "447766511332");
        sleepSeconds(20);
        billingSystem.callCompleted("447722113434", "447766511332");

        billingSystem.callInitiated("447722113434", "447711111111");
        sleepSeconds(30);
        billingSystem.callCompleted("447722113434", "447711111111");

        billingSystem.callInitiated("447777765432", "447711111111");
        sleepSeconds(60);
        billingSystem.callCompleted("447777765432", "447711111111");

        billingSystem.createCustomerBills();
    }

    private static void sleepSeconds(int n) throws InterruptedException {
        Thread.sleep(n * 1000);
    }
}
```

this, combined with reading the source code, should show you what the system does.

Assessment Details

For this coursework you should write a report (one per group) describing how you approached the project.

*In the report, describe the steps that you take in completing this work. Show **how** and **why** (if you did) you:*

a) added acceptance tests covering the existing code, and any new code you write

b) added unit tests

c) refactored the existing code in order to achieve the above

d) wrote new code

e) anything else you did

f) how would you manage adding further features to this code in future

If you don't have space in the report to describe every aspect of what you did, focus on the most interesting parts.

Maximum length of report 7 pages.

Assessment Critera

This assignment is designed to allow you to demonstrate use of the tools and techniques that we covered in the course.

For each technique that you choose to use, write about your reasoning for applying it, and how it helps you in completing the project. Describe the steps you have to go through to apply them.

Show selected snippets of code that demonstrate the changes that you make, rather than pasting in large chunks, or just the finished product.

Hints

If you find it makes the problem easier, you can assume that no phone call lasts more than 24 hours.

If you find using Java's java.util.Date cumbersome, you might like to look at the JodaTime project and try using that in your code.

Administration

Due date: 5th December 2012

Submit your report as a pdf file, and a zip file containing your code, electronically via CATE. Assessment will mostly be done on the content of the report, but we may want to refer to the code for more detail on how you did certain things that you describe in the report.

Appendix: The code you receive to work with

```
package com.acmetelecom;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.*;

public class BillingSystem {

    private List<CallEvent> callLog = new ArrayList<CallEvent>();

    public void callInitiated(String caller, String callee) {
        callLog.add(new CallStart(caller, callee));
    }

    public void callCompleted(String caller, String callee) {
        callLog.add(new CallEnd(caller, callee));
    }

    public void createCustomerBills() {
        List<Customer> customers = CentralCustomerDatabase.getInstance().getCustomers();
        for (Customer customer : customers) {
            createBillFor(customer);
        }
        callLog.clear();
    }

    private void createBillFor(Customer customer) {
        List<CallEvent> customerEvents = new ArrayList<CallEvent>();
        for (CallEvent callEvent : callLog) {
            if (callEvent.getCaller().equals(customer.getPhoneNumber())) {
                customerEvents.add(callEvent);
            }
        }

        List<Call> calls = new ArrayList<Call>();

        CallEvent start = null;
        for (CallEvent event : customerEvents) {
            if (event instanceof CallStart) {
                start = event;
            }
            if (event instanceof CallEnd && start != null) {
                calls.add(new Call(start, event));
                start = null;
            }
        }

        BigDecimal totalBill = new BigDecimal(0);
        List<LineItem> items = new ArrayList<LineItem>();

        for (Call call : calls) {

            Tariff tariff = CentralTariffDatabase.getInstance().tarriffFor(customer);

            BigDecimal cost;

            DaytimePeakPeriod peakPeriod = new DaytimePeakPeriod();
            if (peakPeriod.offPeak(call.startTime()) && peakPeriod.offPeak(call.endTime())
                && call.durationSeconds() < 12 * 60 * 60) {
                cost = new BigDecimal(call.durationSeconds()).multiply(tariff.offPeakRate());
            } else {
                cost = new BigDecimal(call.durationSeconds()).multiply(tariff.peakRate());
            }

            cost = cost.setScale(0, RoundingMode.HALF_UP);
            BigDecimal callCost = cost;
            totalBill = totalBill.add(callCost);
            items.add(new LineItem(call, callCost));
        }
        new BillGenerator().send(customer, items, MoneyFormatter.penceToPounds(totalBill));
    }

    static class LineItem {
        private Call call;
        private BigDecimal callCost;

        public LineItem(Call call, BigDecimal callCost) {
            this.call = call;
            this.callCost = callCost;
        }

        public String date() {
            return call.date();
        }

        public String callee() {
            return call.callee();
        }

        public String durationMinutes() {
          return "" + call.durationSeconds() / 60 + ":" + String.format("%02d", call.durationSeconds
() % 60);
        }

        public BigDecimal cost() {
            return callCost;
        }
    }
}



package com.acmetelecom;

import java.util.List;

public class BillGenerator {

  public void send(Customer customer, List<BillingSystem.LineItem> calls, String totalBill) {

   Printer printer = HtmlPrinter.getInstance();
   printer.printHeading(customer.getFullName(), customer.getPhoneNumber(), customer.getPricePlan());
       for (BillingSystem.LineItem call : calls) {
           printer.printItem(call.date(), call.callee(), call.durationMinutes(),
           MoneyFormatter.penceToPounds(call.cost()));
        }
        printer.printTotal(totalBill);
    }

}



package com.acmetelecom;

import javax.swing.text.DateFormatter;
import java.math.BigDecimal;
import java.math.RoundingMode;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Calendar;
import java.util.Date;

public class Call {
    private CallEvent start;
    private CallEvent end;

    public Call(CallEvent start, CallEvent end) {
        this.start = start;
        this.end = end;
    }
```

```java
    public String callee() {
        return start.getCallee();
    }

    public int durationSeconds() {
        return (int) (((end.time() - start.time()) / 1000));
    }

    public String date() {
        return SimpleDateFormat.getInstance().format(new Date(start.time()));
    }

    public Date startTime() {
        return new Date(start.time());
    }

    public Date endTime() {
        return new Date(end.time());
    }
}
```

```java
package com.acmetelecom;

public abstract class CallEvent {
    private String caller;
    private String callee;
    private long time;

    public CallEvent(String caller, String callee, long timeStamp) {
        this.caller = caller;
        this.callee = callee;
        this.time = timeStamp;
    }

    public String getCaller() {
        return caller;
    }

    public String getCallee() {
        return callee;
    }

    public long time() {
        return time;
    }
}
```

```java
package com.acmetelecom;

public class CallStart extends CallEvent {
    public CallStart(String caller, String callee) {
        super(caller, callee, System.currentTimeMillis());
    }
}
```

```java
package com.acmetelecom;

public class CallEnd extends CallEvent {
    public CallEnd(String caller, String callee) {
        super(caller, callee, System.currentTimeMillis());
    }
}
```

```java
package com.acmetelecom;

import java.util.Calendar;
import java.util.Date;

class DaytimePeakPeriod {

    public boolean offPeak(Date time) {
        Calendar calendar = Calendar.getInstance();
        calendar.setTime(time);
        int hour = calendar.get(Calendar.HOUR_OF_DAY);
        return hour < 7 || hour >= 19;
    }
}
```

```java
package com.acmetelecom;

import java.math.BigDecimal;

class MoneyFormatter {
    public static String penceToPounds(BigDecimal pence) {
        BigDecimal pounds = pence.divide(new BigDecimal(100));
        return String.format("%.2f", pounds.doubleValue());
    }
}
```

```java
package com.acmetelecom;

public interface Printer {

    void printHeading(String name, String phoneNumber, String pricePlan);

    void printItem(String time, String callee, String duration, String cost);

    void printTotal(String total);
}
```

```java
package com.acmetelecom;

class HtmlPrinter implements Printer {

    private static Printer instance = new HtmlPrinter();

    private HtmlPrinter() {
    }

    public static Printer getInstance() {
        return instance;
    }

    public void printHeading(String name, String phoneNumber, String pricePlan) {
        beginHtml();
        System.out.println(h2(name + "/" + phoneNumber + " - " + "Price Plan: " + pricePlan));
        beginTable();
    }

    private void beginTable() {
        System.out.println("<table border=\"1\">");
        System.out.println(tr(th("Time") + th("Number") + th("Duration") + th("Cost")));
    }

    private void endTable() {
        System.out.println("</table>");
    }

    private String h2(String text) {
        return "<h2>" + text + "</h2>";
    }

    public void printItem(String time, String callee, String duration, String cost) {
        System.out.println(tr(td(time) + td(callee) + td(duration) + td(cost)));
    }

    private String tr(String text) {
        return "<tr>" + text + "</tr>";
    }

    private String th(String text) {
        return "<th width=\"160\">" + text + "</th>";
    }

    private String td(String text) {
        return "<td>" + text + "</td>";
    }

    public void printTotal(String total) {
        endTable();
        System.out.println(h2("Total: " + total));
        endHtml();
    }

    private void beginHtml() {
        System.out.println("<html>");
        System.out.println("<head></head>");
        System.out.println("<body>");
        System.out.println("<h1>");
        System.out.println("Acme Telecom");
        System.out.println("</h1>");
    }

    private void endHtml() {
        System.out.println("</body>");
        System.out.println("</html>");
    }
}
```