CS 2510 Exam 3 – Fall 2009

Name: ________________________

Student Id (last 4 digits): ________________________

• Write down the answers in the space provided.
• Your programs should work in standard Java 1.5. If you need a method and you don't know whether it is provided, define it.
• For tests you only need to provide the expression that computes the actual value, connecting it with an arrow to the expected value. For example `s.method() -> true` is sufficient.
• Remember that the phrase "develop a class" or "develop a method" means more than just providing a definition. It means to design them according to the **design recipe**. You are *not* required to provide a method template unless the problem specifically asks for one. However, be prepared to struggle if you choose to skip the template step.
• We will not answer *any* questions during the exam.

Good luck.

Problem	Points	/
1		/23
2		/27
Total		/50

23 Points

Problem 1

A bad apple spoils the whole bunch. So, we are given a `Bag` of `Apple`s and want to help the customer to select only the good apples.

How can you tell what is a bad apple? Well, different customers have different ways of deciding which apples are bad.

One customer thinks all bruised apples are bad. Another customer thinks that green apples are bad and want to select apples of any other color.

The class `Apple` and the classes that represent a bag of apples have been defined as follows:

```
+--------------------+
| Bag                |
+--------------------+
| boolean isEmpty()  |
| void add(Apple)    |
| Apple remove()     |
+--------------------+

+------------------+
| Apple            |
+------------------+
| String kind      |
| boolean bruised  |
| String color     |
+------------------+
```

The methods for the `Bag` are defined to do the following:

```
// is this bag empty?
boolean isEmpty()...

// EFFECT: add the given apple to this bag
void add(Apple)...

// produce one apple from the bag
// throws RuntimeEception if the bag has no apples
// EFFECT: the bag no longer has the apple that has been removed
Apple remove()...
```

We only sell three kinds of apples, `"mac"`, `"fuji"`, and `"jona"` - each can come in `"red"`, `"green"`, or `"gold"` color.

A. **4 points**

Make examples of at least four apples. Add a `setup` method that creates examples of three different `Bag`s of apples.

B. **19 points**

Design the method `sortpick` that collects all **good** apples from the given `Bag` into an `ArrayList<Apple>` is such way that all `"fuji"` apples are first, then all `"jona"` apples, then all `"mac"` apples.

Note 1: Remember to use helper methods as needed.

Note 2: If this is too hard, select always only the apples that are not bruised.

Note 3: If this is still hard, don't worry about making `"fuji"` apples first, then `"jona"` apples, then `"mac"` apples.

27 Points

Problem 2

You are in charge of scheduling meeting for the members of your group at work. To avoid schedule overlaps, you have decided to record the meetings using the data definitions shown on the next page. (The `import java.util.*;` statement has been omitted.)

There are two options for how to represent a *schedule*. Choose the one that you find easier to work with. We think the first one is easier.

Read the whole problem before you start working on any part of it.

Hint: Once you decide which data definition you choose, stick with it.

A. **7 points**

Make examples for all classes shown.

Include in your examples a schedule for Tom with a staff meeting from 9:00 am till 10:30 am in the room 23, a group meeting from 2:00 pm till 3:00 pm in the room 35, and a reception from 5:30 pm until 6:00 pm in the room 12.

B. **20 points**

Design the method `isValid` for the class that represents the schedule for one person. The meetings scheduled for this person may not overlap, they must be given with the times ordered from the earliest to the latest, and if the next meeting is in another room, there must be at least 5 minutes time to switch the rooms.

Note 1: As you define methods (and helper methods) make sure you start by specifying in which class the method is defined.

Note 2: Once you have given the purpose and the header for a method, you may assume in subsequent problems that it has been defined, even if you do not complete the rest of the method design.

```
// to represent a time on a 24 hour clock
class ClockTime{
  int hour;
  int minute;

  ClockTime(int hour, int minute){
    this.hour = hour;
    this.minute = minute;
  }
}

// to represent one meeting in a schedule
class Meeting{
  String purpose;
  int room;
  ClockTime start;
  int duration; // in minutes

  Meeting(String purpose, int room,
          ClockTime start, int duration){
    this.purpose = purpose;
    this.room = room;
    this.start = start;
    this.duration = duration;
  }
}

// to represent a list of meetings
interface ILoM{}

// to represent an empty list of meetings
class MtLoM implements ILoM{
  MtLo(){}
}

//to represent a nonempty list of meetings
class ConsLoM{
  Meeting first;
  ILoM rest;

  ConsLoM(Meeting first, ILoM rest){
    this.first = first;
    this.rest = rest;
  }
}
```

```
//to represent a schedule for one manager for one day
// Note: uses a recursively defined list
class ScheduleList{
  String name;
  ILoM list;

  ScheduleList(String name, ILoM list){
    this.name = name;
    this.list = list;
  }
}

//to represent a schedule for one manager for one day
//Note: uses an ArrayList
class ScheduleArList{
  String name;
  ArrayList<Meeting> list;

  ScheduleArList(String name, ArrayList<Meeting> list){
    this.name = name;
    this.list = list;
  }
}
```