In exercises 1-4, you and another programmer will use a Java interface to implement two interacting classes at the same time. Before continuing, please choose another person to work with, and form a programming team.

1. Your team will implement an appointment scheduling system. Specifically, you are two implement two main classes, Appointment and Schedule. Each programmer should implement a different class. The Schedule class will schedule Appointment objects. Since the Schedule class uses the Appointment class, you might think that the programmer who is responsible for implementing Schedule cannot write code that compiles until the Appointment class has been implemented. Actually, the two programmers need only define an interface for Appointment to implement so that the person who is implementing Schedule can write code that compiles. Note that this interface has already been defined for you below.

You are to implement the following two classes.

(a) The Appointment class.

An appointment has the following attributes: a day of the week, a start hour, an end hour, a location, and a description. Location and description can be Strings, and you should represent days of the week as integers. This class must implement the following interface.

```java
public interface Schedulable {
    public int getDayOfWeek();
    public int getStartHour();
    public int getEndHour();
}
```

In addition, implement a public String toString() method that returns a string representation of the appointment.

(b) The Schedule class. This class does the actual scheduling. Specifically, you should implement the following public methods.

i. public boolean scheduleAppointment(Schedulable appointment)

This method should return true if the appointment could be scheduled and false if there was a time conflict. For simplicity, appointment times are defined by days of the week and whole number hours.

HINT: This class should have private arrays of Schedulable objects, one for each day of the week. The positions in the array represent the hours in the day. Instead of having 7 arrays, it would be a good idea to use a 2-D array.
ii. public Schedulable getAppointmentAtTime(int day, int hour)
This method returns the appointment that is scheduled at the given
time. If no appointment is scheduled for that time, then this method
returns null.

2. Once your team has implemented these two classes, write a driver program to
test your scheduling system.

3. Both team members can work together on this exercise. Implement a new kind
of appointment, called SecretAppointment. The difference between a secret
appointment and a regular appointment is that we don’t want to include the
time and location of a secret appointment in its string representation. You have
already implemented many of the features of SecretAppointment. Do not write
duplicate code. (HINT: Make SecretAppointment a subclass of Appointment)

4. Write a comprehensive test driver to demonstrate the features of your schedul-
ing system. When you have finished, please demonstrate your system to the
instructor. Feel free to add features that you think should be included.