#3. An appointment scheduling system is represented by the following three classes: `TimeInterval`, `Appointment`, and `DailySchedule`. In this question, you will implement one method in the `Appointment` class and two methods in the `DailySchedule` class.

A `TimeInterval` object represents a period of time. The `TimeInterval` class provides a method to determine if another time interval overlaps with the time interval represented by the current `TimeInterval` object. An `Appointment` object contains a time interval for the appointment and a method that determines if there is a time conflict between the current appointment and another appointment. The declarations of the `TimeInterval` and `Appointment` classes are shown below.

```
public class TimeInterval
{
  // returns true if interval overlaps with this TimeInterval;
  // otherwise, returns false
  public boolean overlapsWith(TimeInterval interval)
  {  /* implementation not shown */  }

  // There may be fields, constructors, and methods that are not shown.
}


public class Appointment
{
  // returns the time interval of this Appointment
  public TimeInterval getTime()
  {  /* implementation not shown */  }

  // returns true if the time interval of this Appointment
  // overlaps with the time interval of other;
  // otherwise, returns false
  public boolean conflictsWith(Appointment other)
  {  /* to be implemented in part (a) */  }

  // There may be fields, constructors, and methods that are not shown.
}
```

(a) Write the `Appointment` method `conflictsWith`. If the time interval of the current appointment overlaps with the time interval of the appointment `other`, method `conflictsWith` should return `true`, otherwise, it should return `false`.

Complete method `conflictsWith` below.

```
  // returns true if the time interval of this Appointment
  // overlaps with the time interval of other;
  // otherwise, returns false
  public boolean conflictsWith(Appointment other)
```

(b) A `DailySchedule` object contains a list of nonoverlapping `Appointment` objects. The `DailySchedule` class contains methods to clear all appointments that conflict with a given appointment and to add an appointment to the schedule.

```
public class DailySchedule
{
  // contains Appointment objects, no two Appointments overlap
  private ArrayList apptList;


  public DailySchedule()
  {  apptList = new ArrayList();  }


  // removes all appointments that overlap the given Appointment
  // postcondition: all appointments that have a time conflict with
  //                appt have been removed from this DailySchedule
  public void clearConflicts(Appointment appt)
  {  /* to be implemented in part (b) */  }


  // if emergency is true, clears any overlapping appointments and adds
  // appt to this DailySchedule; otherwise, if there are no conflicting
  // appointments, adds appt to this DailySchedule;
  // returns true if the appointment was added;
  // otherwise, returns false
  public boolean addAppt(Appointment appt, boolean emergency)
  {  /* to be implemented in part (c) */  }


  // There may be fields, constructors, and methods that are not shown.
}
```

Write the `DailySchedule` method `clearConflicts`. Method `clearConflicts` removes all appointments that conflict with the given appointment.

In writing method `clearConflicts`, you may assume that `conflictsWith` works as specified, regardless of what you wrote in part (a).

Complete method `clearConflicts` below.

```
// removes all appointments that overlap the given Appointment
// postcondition: all appointments that have a time conflict with
//                appt have been removed from this DailySchedule
public void clearConflicts(Appointment appt)
```

(c) Write the `DailySchedule` method `addAppt`. The parameters to method `addAppt` are an appointment and a `boolean` value that indicates whether the appointment to be added is an emergency. If the appointment is an emergency, the schedule is cleared of all appointments that have a time conflict with the given appointment and the appointment is added to the schedule. If the appointment is not an emergency, the schedule is checked for any conflicting appointments. If there are no conflicting appointments, the given appointment is added to the schedule. Method `addAppt` returns `true` if the appointment was added to the schedule; otherwise, it returns `false`.

In writing method `addAppt`, you may assume that `conflictsWith` and `clearConflicts` work as specified, regardless of what you wrote in parts (a) and (b).

Complete method `addAppt` below.

```
// if emergency is true, clears any overlapping appointments and adds
// appt to this DailySchedule; otherwise, if there are no conflicting
// appointments, adds appt to this DailySchedule;
// returns true if the appointment was added;
// otherwise, returns false
public boolean addAppt(Appointment appt, boolean emergency)
```