Consider the problem of representing a filing cabinet with drawers of student records. A filing cabinet is implemented using a linked list of drawers. Each drawer is implemented using a linked list of student records. All student records in a drawer have a student ID less than or equal to the drawer's maximum student ID, and student records are stored in a drawer in ascending order by student ID.

The diagram below illustrates the structure of a filing cabinet as implemented by the class `FilingCabinet`. The data member `myDrawerList` is an instance of `ListNode` that implements a linked list of `Drawer` objects in ascending order by maximum ID.

```
```

The class `Student` is declared as follows.
public class Student
{
    // constructor and data members not shown

    // returns id of this student
    public int getID()
    {
        // not shown
    }

    // returns name of this student
    public String getName()
    {
        // not shown
    }

    // precondition: o is an instance of student
    // postcondition: returns true if o equals this student
    // otherwise returns false
    public boolean equals(Object o)
    {
        // not shown
    }
}

The class **Drawer** is declared as follows.

public class Drawer
{
    private int myMaxID;            // maximum ID in this drawer
    private ListNode myStudents;    // all students in this drawer

    // constructor and some methods not shown

    // returns maximal ID for this drawer
    public int getMaxID()
    {
        return myMaxID;
    }

    // remove Student equal to s from this drawer
    public void removeStudent(Student s)
    {
        // you will write this
    }

    // return first node in this drawer's linked list
    public ListNode getFirst()
    {
        return myStudents;
    }
}

The class **FilingCabinet** is declared as follows.
public class FilingCabinet
{
    private ListNode myDrawerList;

    // precondition: this filing cabinet has at least one Drawer;
    // studentID is less than or equal to maximum ID
    // of last Drawer
    // postcondition: returns the first Drawer d such that
    // d.getMaxID() >= studentID

    public Drawer findDrawer(int studentID)
    {
        // you will write this
    }

    // precondition: student.getID() is less than or equal to
    // maximum ID of last Drawer
    // postcondition: if there is a Student s in this filing cabinet
    // equal to student, then s is removed from the
    // drawer in which it is located; otherwise this
    // FilingCabinet is unchanged

    public void removeStudent(Student student)
    {
        Drawer d = findDrawer(student.getID());
        d.removeStudent(student);
    }
}

Part A

Write FilingCabinet method findDrawer, which is described as follows. Method findDrawer returns the Drawer object in which studentID would be found. Method findDrawer returns the first Drawer in the list myDrawerList for which studentID is less than or equal to the maximum student ID number that can be filed in the drawer.

Complete findDrawer below.

    // precondition: this filing cabinet has at least one Drawer;
    // studentID is less than or equal to maximum ID
    // of last Drawer
    // postcondition: returns the first Drawer d such that
    // d.getMaxID() >= studentID

    public Drawer findDrawer(int studentID)
    {
        // you will write this
    }

Part B

Write the Drawer method removeStudent, which is described as follows. Method removeStudent removes the Student object equal to student from the Drawer if there is such an object. If there is no such object then the Drawer is unchanged.
Complete method `removeStudent` below.

```java
// precondition: student.getID() is less than or equal to
//               maximum ID of this Drawer
// postcondition: if there is a Student s in this drawer
//                equal to student, then s is removed from the
data; otherwise this drawer is unchanged

public void removeStudent(Student student)
{

}
```

---

Owen L. Astrachan

Last modified: Wed May 14 10:15:37 EDT 2003