10.4 Structural Patterns

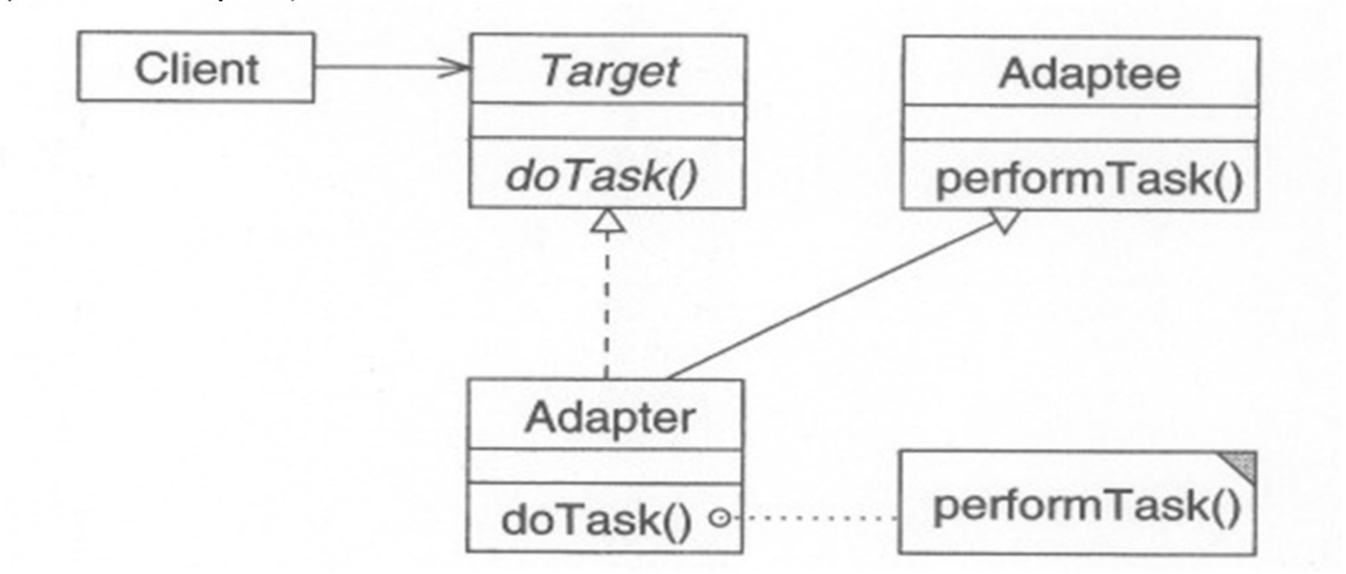
- 10.4.1 Design Pattern : Adapter
  - We often find some classes that provide functionality that can be reused. However, the interface providing the functionality is not the interface expected by the client.
  - Adapter pattern : addresses the issue of adapting an interface to accommodate the needs of a client that hopes to reuse the functionality but excepts a different interface.

10.4 Structural Patterns

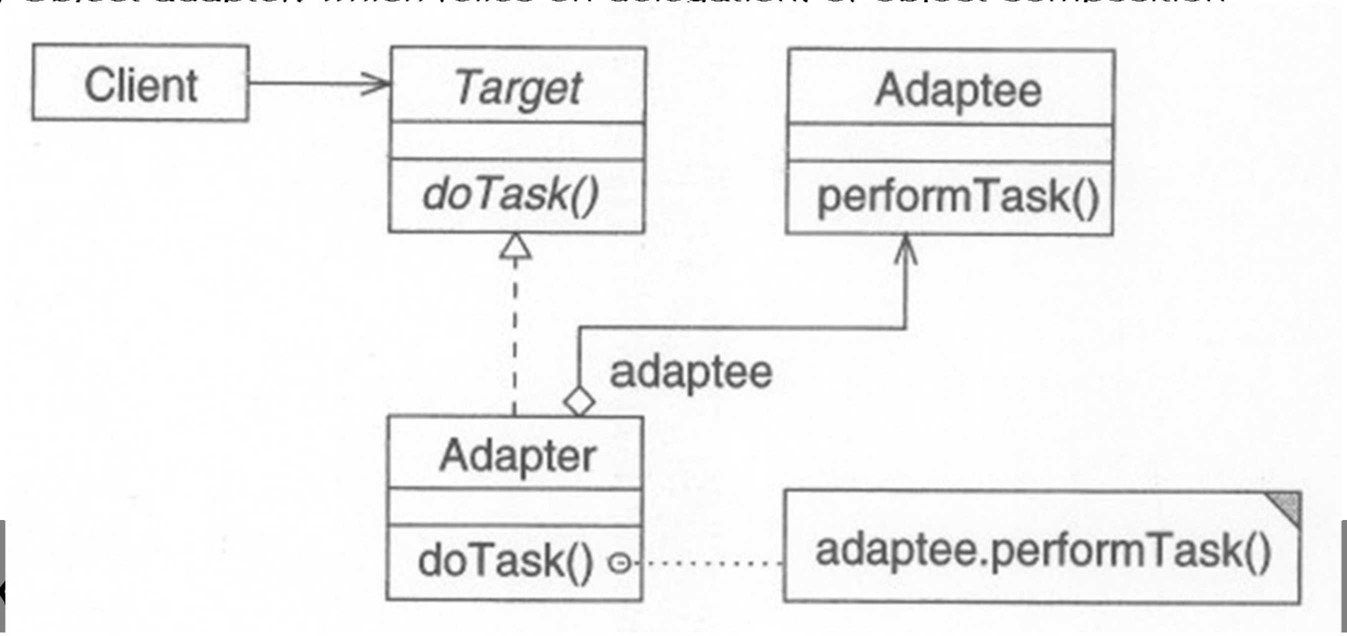
- Design Pattern : Adapter
  - Category : Structural design pattern.
  - Intent : To convert the interface of a class into another interface that clients expect.
  - Also Known As : Wrapper.
  - Applicability : Use the Adapter design pattern
    - to use an existing class with an interface different from the desired interface.

Page 514. Two forms of the Adapter Pattern.

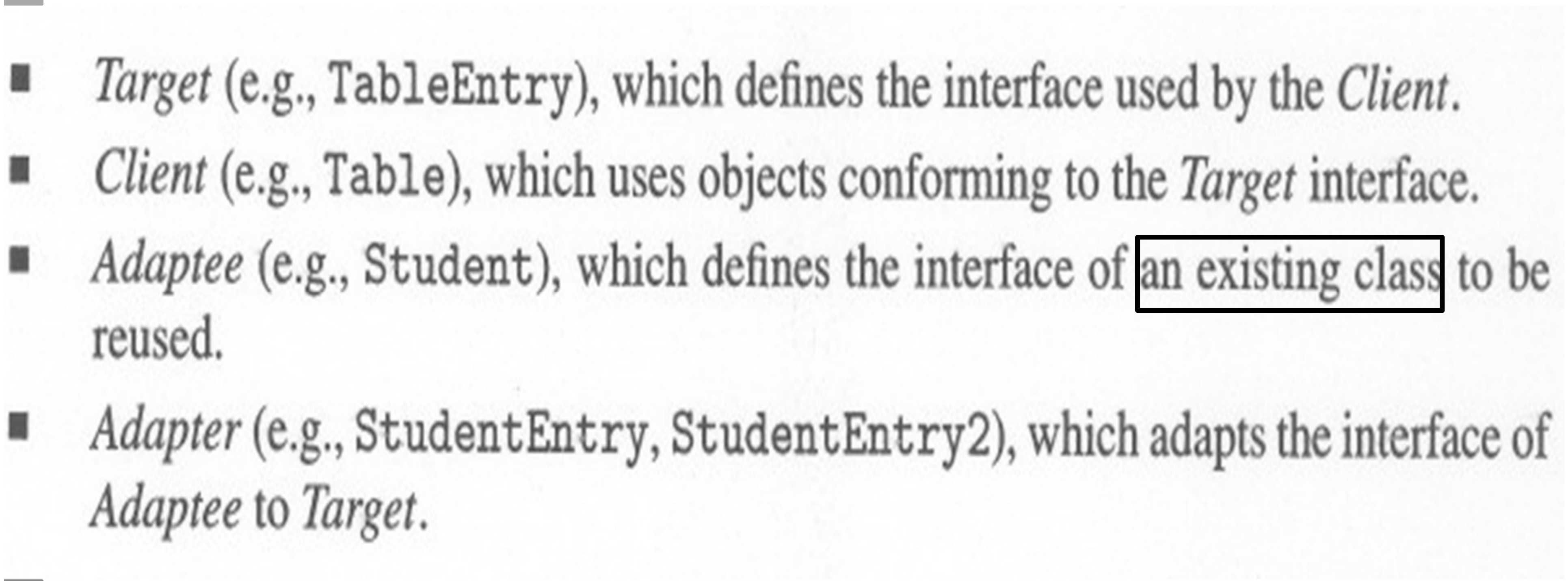
- 1) Class adapter, which relies on inheritance.



- 2) Object adapter, which relies on delegation, or object composition



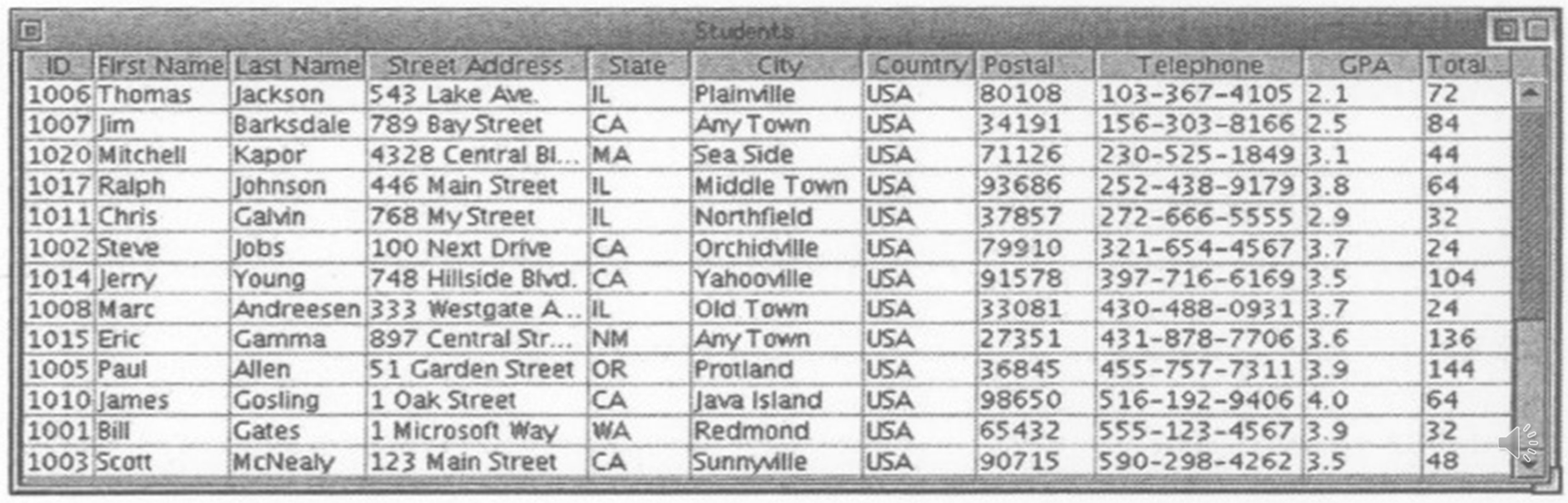
■ The participants of the Adapter design pattern are the following:



- *Target* (e.g., `TableEntry`), which defines the interface used by the *Client*.
- *Client* (e.g., `Table`), which uses objects conforming to the *Target* interface.
- *Adaptee* (e.g., `Student`), which defines the interface of an existing class to be reused.
- *Adapter* (e.g., `StudentEntry`, `StudentEntry2`), which adapts the interface of *Adaptee* to *Target*.

Fig 10.9. The Table of entries of student information

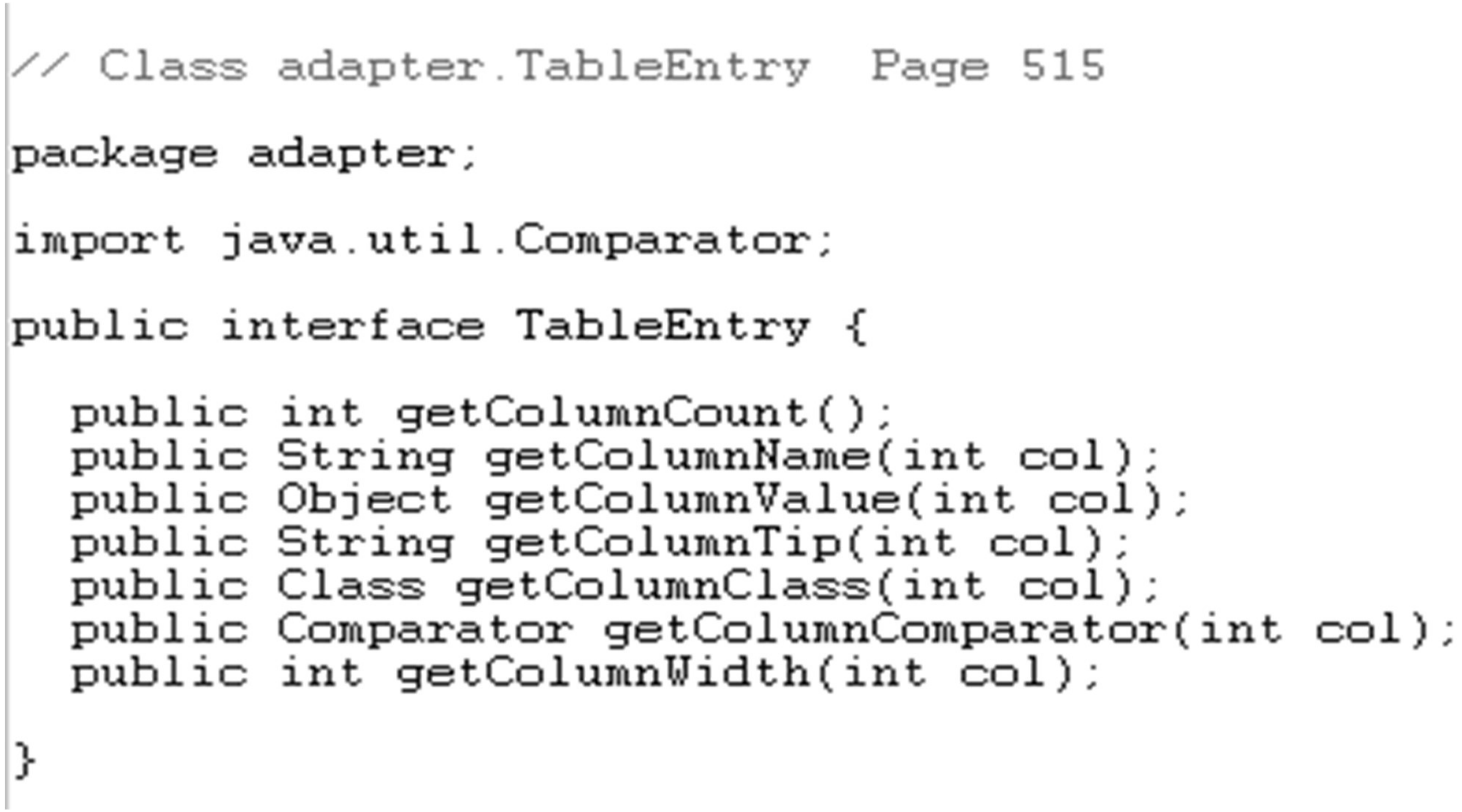
- A generic table to display a list of objects of any class in tabular form
  - when you click on the header of a column, the list of objects will be sorted based on the values in that column.



Students

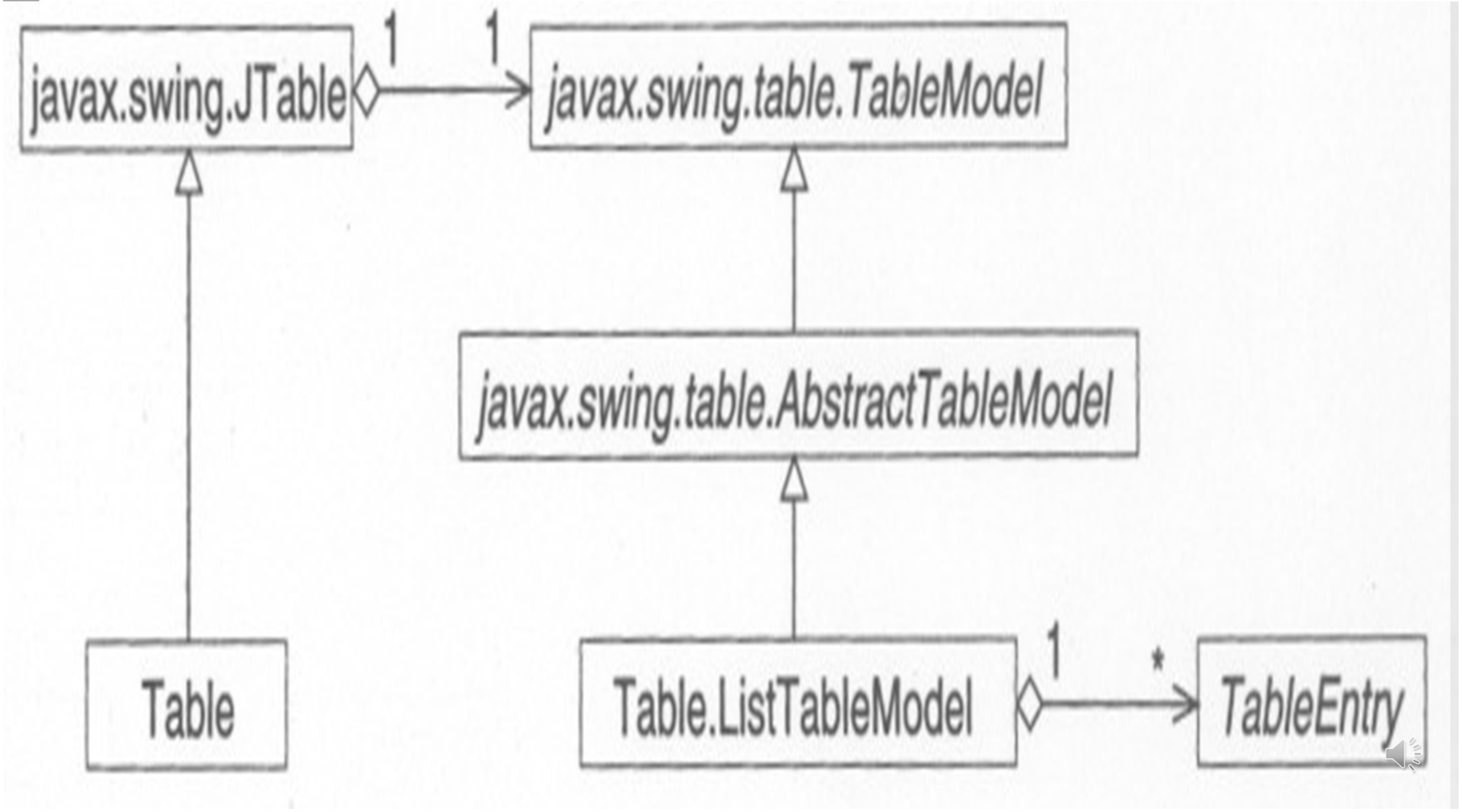
ID	First Name	Last Name	Street Address	State	City	Country	Postal ...	Telephone	GPA	Total...
1006	Thomas	Jackson	543 Lake Ave.	IL	Plainville	USA	80108	103-367-4105	2.1	72
1007	Jim	Barksdale	789 Bay Street	CA	Any Town	USA	34191	156-303-8166	2.5	84
1020	Mitchell	Kapor	4328 Central Bl...	MA	Sea Side	USA	71126	230-525-1849	3.1	44
1017	Ralph	Johnson	446 Main Street	IL	Middle Town	USA	93686	252-438-9179	3.8	64
1011	Chris	Galvin	768 My Street	IL	Northfield	USA	37857	272-666-5555	2.9	32
1002	Steve	Jobs	100 Next Drive	CA	Orchidville	USA	79910	321-654-4567	3.7	24
1014	Jerry	Young	748 Hillside Blvd.	CA	Yahooville	USA	91578	397-716-6169	3.5	104
1008	Marc	Andreesen	333 Westgate A...	IL	Old Town	USA	33081	430-488-0931	3.7	24
1015	Eric	Gamma	897 Central Str...	NM	Any Town	USA	27351	431-878-7706	3.6	136
1005	Paul	Allen	51 Garden Street	OR	Protland	USA	36845	455-757-7311	3.9	144
1010	James	Gosling	1 Oak Street	CA	Java Island	USA	98650	516-192-9406	4.0	64
1001	Bill	Gates	1 Microsoft Way	WA	Redmond	USA	65432	555-123-4567	3.9	32
1003	Scott	McNealy	123 Main Street	CA	Sunnyville	USA	90715	590-298-4262	3.5	48

- Adapter.Table class
  - extends the Jtable in swing
  - the design of the generic table is shown in Figure 10.10
  - the key to the design of the generic table is that the data entries shown in the table must be instances of a class that conforms to the TableEntry interface.

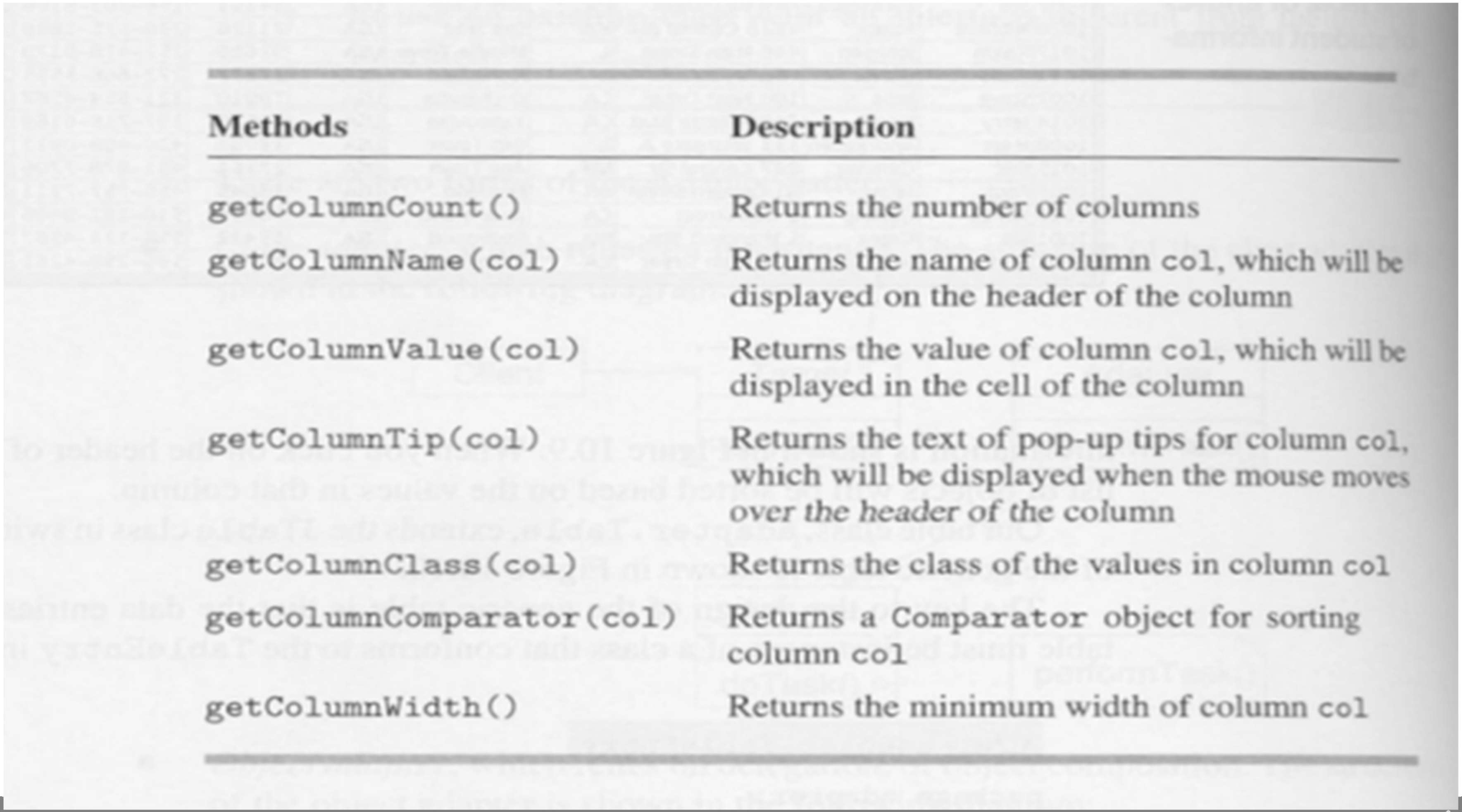


```
// Class adapter.TableEntry  Page 515

package adapter;

import java.util.Comparator;

public interface TableEntry {

  public int getColumnCount();
  public String getColumnName(int col);
  public Object getColumnValue(int col);
  public String getColumnTip(int col);
  public Class getColumnClass(int col);
  public Comparator getColumnComparator(int col);
  public int getColumnWidth(int col);

}
```

Figure 10.10 page 515
The design of the generic table with sorting capability


javax.swing.JTable
1
1
javax.swing.table.TableModel
javax.swing.table.AbstractTableModel
Table
Table.ListTableModel
1
*
TableEntry

Methods of the TableEntry interface (Page 516)



Methods	Description
`getColumnCount()`	Returns the number of columns
`getColumnName(col)`	Returns the name of column `col`, which will be displayed on the header of the column
`getColumnValue(col)`	Returns the value of column `col`, which will be displayed in the cell of the column
`getColumnTip(col)`	Returns the text of pop-up tips for column `col`, which will be displayed when the mouse moves over the header of the column
`getColumnClass(col)`	Returns the class of the values in column `col`
`getColumnComparator(col)`	Returns a `Comparator` object for sorting column `col`
`getColumnWidth()`	Returns the minimum width of column `col`

- Class adapter.Table (p. 516)
  - the implementation of the Table Class is immaterial to the discussion of the Adapter pattern.
-

```
package adapter;

import java.awt.Color;
import java.awt.Point;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.table.*;
import javax.swing.event.*;
import java.util.*;

public class Table extends JTable {
  public Table() {
    this(null);
  }
```

```
public Table(List entries) {
    super(new ListTableModel(entries));
    model = (ListTableModel) dataModel;

    getTableHeader().addMouseListener(new MouseAdapter() {
            public void mousePressed(MouseEvent e) {
              Point p = e.getPoint();
              JTableHeader header = (JTableHeader) e.getSource();
              int column = header.columnAtPoint(p);
              if (model.sort(column)) {
                clearSelection();
                updateUI();
              }
            }
      });
```

```
setSelectionMode(ListSelectionModel.SINGLE_SELECTION);

  int columnCount = model.getColumnCount();
  for (int i = 0; i < columnCount; i++) {
    TableColumn column = getColumnModel().getColumn(i);
    DefaultTableCellRenderer renderer = new DefaultTableCellRenderer();
    String tip = model.getColumnTip(i);
    renderer.setToolTipText(tip);
    column.setCellRenderer(renderer);
    int w = model.getColumnWidth(i);
    if (w > 0) {
            column.setPreferredWidth(w);
    }
  }
}

protected ListTableModel model;

static class ListTableModel extends AbstractTableModel {

  public ListTableModel(List entries) {
    if (entries != null &&
             entries.size() > 0) {
            Object obj = entries.get(0);
            if (obj != null &&
               obj instanceof TableEntry) {
             this.prototype = (TableEntry) obj;
             setData(entries);
            }
    }
  }
```

```
public ListTableModel(TableEntry prototype) {
    this.prototype = prototype;
  }

  public int getColumnCount() {
    if (prototype != null) {
            return prototype.getColumnCount();
    }
    return 0;
  }

  public int getRowCount() {
    if (entries != null) {
            return entries.size();
    } else {
            return 0;
    }
  }

  public String getColumnName(int col) {
    if (prototype != null) {
            return prototype.getColumnName(col);
    }
    return null;
  }

  public Object getValueAt(int row, int col) {
    if (entries != null) {
            TableEntry entry = getTableEntry(row);
            if (entry != null) {
              return entry.getColumnValue(col);
            }
    }
    return null;
  }
```

```
 public Class getColumnClass(int col) {
    if (prototype != null) {
            return prototype.getColumnClass(col);
    }
    return String.class;
  }

  public String getColumnTip(int col) {
    if (prototype != null) {
            return prototype.getColumnTip(col);
    }
    return null;
  }

  public Comparator getColumnComparator(int col)
{
    if (prototype != null) {
            return
prototype.getColumnComparator(col);
    }
    return null;
  }

  public int getColumnWidth(int col) {
    if (prototype != null) {
            return prototype.getColumnWidth(col);
    }
    return -1;
  }

  public boolean isCellEditable(int row, int col) {
    return false;
  }
 public void setValueAt(Object value, int row, int col)
{
   }

   public void clearData() {
     entries = null;
   }

   public void setData(List entries) {
     this.entries = entries;
   }

   public boolean sort(int col) {
     if (entries != null &&
                 col >=0 &&
                 col < getColumnCount()) {
               Comparator c =
getColumnComparator(col);
               if (c != null) {
                 Collections.sort(entries, c);
                 return true;
               }
     }
     return false;
   }
```

```
public TableEntry getTableEntry(int i) {
      if (entries != null &&
                   i >= 0 &&
                   i < entries.size()) {
                 return (TableEntry) entries.get(i);
      }
      return null;
    }

    protected TableEntry prototype;
    protected List entries; // elements are instance of
TableEntry

  }

}
```

- Class adapter.Student(p.520)
  - given the Student Class, we want to display the student information using the generic table
- The problem
  - the Table Class expects the entries to be instances of a class that conforms to the TableEntry interface, while the Student class does not conform to the TableEntry interface

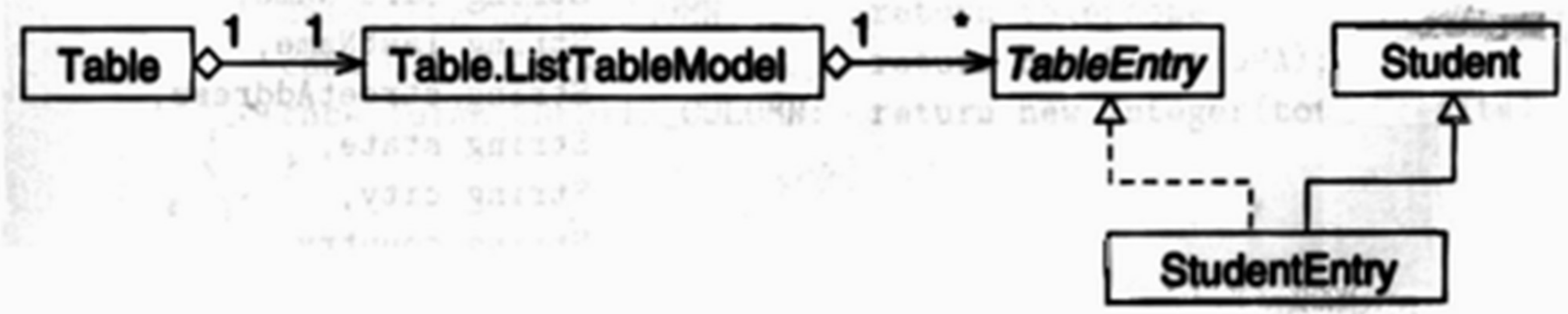
```
package adapter;

public class Student implements Cloneable {

  public Student() {}

  public Student(String ID,
                                  String firstName,
                                  String lastName,
                                  String streetAddress,
                                  String state,
                                  String city,
                                  String country,
                                  String postalCode,
                                  String telephone,
                                  float GPA,
                                  int totalCredits) {
    this.ID = ID;
    this.firstName = firstName;
    this.lastName = lastName;
    this.streetAddress = streetAddress;
    this.state = state;
    this.city = city;
    this.country = country;
    this.postalCode = postalCode;
    this.telephone = telephone;
    this.GPA = GPA;
    this.totalCredits = totalCredits;
  }

  public Object clone()
      throws CloneNotSupportedException {
    return super.clone();
  }
 }
```

```
public String getCity() {
   return city;
 }

 public String getCountry() {
   return country;
 }

 public String getFirstName() {
   return firstName;
 }

 public float getGPA() {
   return GPA;
 }

 public String getID() {
   return ID;
 }

 public String getLastName() {
   return lastName;
 }

 public String getPostalCode() {
   return postalCode;
 }

 public String getState() {
   return state;
 }
```

```
public String getStreetAddress() {
    return streetAddress;
  }

  public String getTelephone() {
    return telephone;
  }

  public int getTotalCredits() {
    return totalCredits;
  }

  public void setCity(String city) {
    this.city = city;
  }

  public void setCountry(String country) {
    this.country = country;
  }

  public void setFirstName(String firstName) {
    this.firstName = firstName;
  }

  public void setGPA(float GPA) {
    this.GPA = GPA;
  }

  public void setID(String ID) {
    this.ID = ID;
  }
```

```
public void setLastName(String lastName) {
    this.lastName = lastName;
  }

  public void setPostalCode(String postalCode) {
    this.postalCode = postalCode;
  }

  public void setState(String state) {
    this.state = state;
  }

  public void setStreetAddress(String streetAddress)
{
    this.streetAddress = streetAddress;
  }

  public void setTelephone(String telephone) {
    this.telephone = telephone;
  }

  public void setTotalCredits(int totalCredits) {
    this.totalCredits = totalCredits;
  }
```

```java
public String toString() {
   StringBuffer s = new StringBuffer();
   s.append("Student [");
   s.append("totalCredits=");
   s.append(totalCredits);
   s.append("; ");
   s.append("GPA=");
   s.append(GPA);
   s.append("; ");
   s.append("telephone=");
   s.append(telephone);
   s.append("; ");
   s.append("postalCode=");
   s.append(postalCode);
   s.append("; ");
   s.append("country=");
   s.append(country);
   s.append("; ");
   s.append("city=");
   s.append(city);
   s.append("; ");
   s.append("state=");
   s.append(state);
   s.append("; ");
   s.append("streetAddress=");
   s.append(streetAddress);
   s.append("; ");
   s.append("lastName=");
   s.append(lastName);
   s.append("; ");
s.append("firstName=");
   s.append(firstName);
   s.append("; ");
   s.append("ID=");
   s.append(ID);
   s.append("]");
   return s.toString();
 }

protected int totalCredits;

 protected float GPA;

 protected String telephone;

 protected String postalCode;

 protected String country;

 protected String city;

 protected String state;

 protected String streetAddress;

 /** The last name of the student */
 protected String lastName;

 /** The first name of the student */
 protected String firstName;

 /** The ID of the student */
 protected String ID;

}
```



- Figure 10.11 page 523
  class adapter : <u>adapt by inheritance</u>
- Instead of rewriting the Student class to make it conform to the TableEntry interface, we can use the Adapter pattern
  - We can use either form of the Adapter to accomplish the task.
  - The design using the class adapter form is shown in Figure 10.11
- the StudentEntry class is the adapter, the Student class is the adaptee
- The tableEntry interface is the target.

- Class adapter.StudentEntry (p. 523)

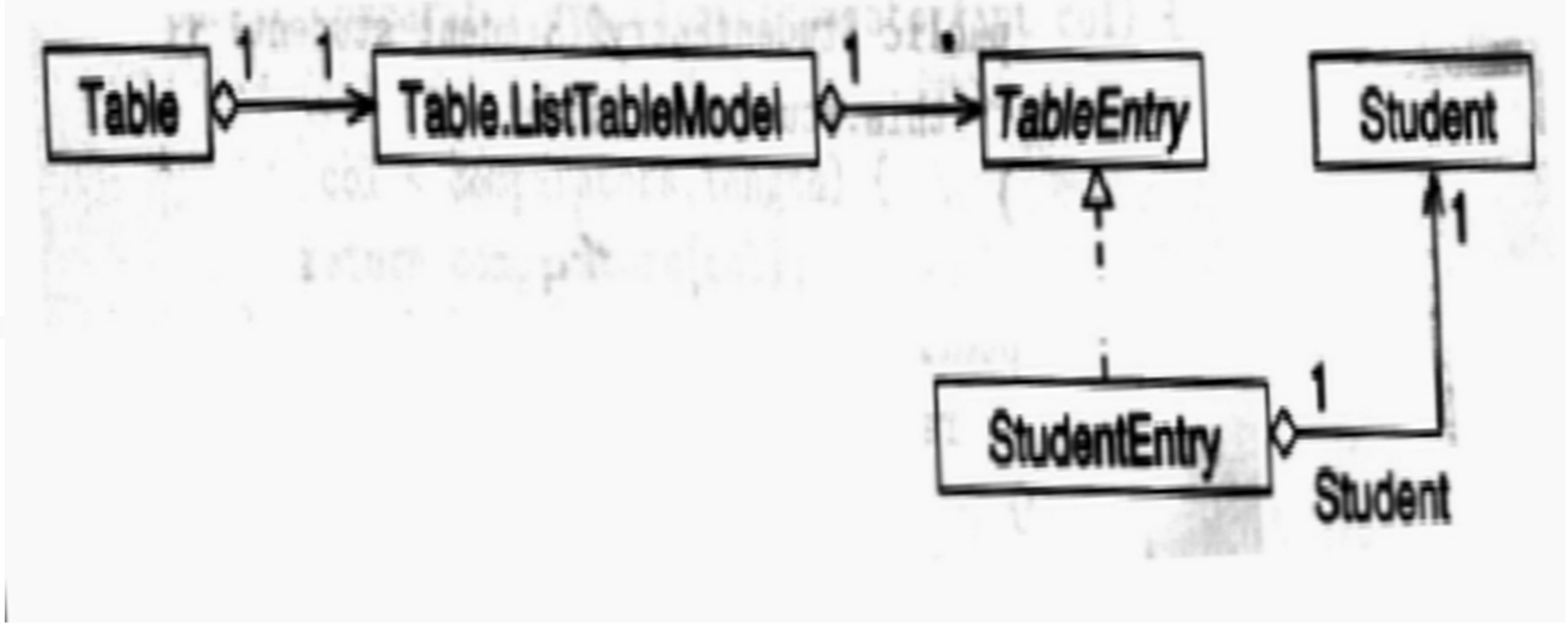
```java
package adapter;

import java.util.Comparator;

/**
 *  Adapter design pattern
 */
public class StudentEntry extends Student implements TableEntry {

  // position of each column
  public static final int ID_COLUMN             = 0;
  public static final int FIRST_NAME_COLUMN     = 1;
  public static final int LAST_NAME_COLUMN      = 2;
  public static final int STREET_ADDRESS_COLUMN = 3;
  public static final int STATE_COLUMN          = 4;
  public static final int CITY_COLUMN           = 5;
  public static final int COUNTRY_COLUMN        = 6;
  public static final int POSTAL_CODE_COLUMN    = 7;
  public static final int TELEPHONE_COLUMN      = 8;
  public static final int GPA_COLUMN            = 9;
  public static final int TOTAL_CREDITS_COLUMN  = 10;
public static final String[] columnNames = {
   "ID",
   "First Name",
   "Last Name",
   "Street Address",
   "State",
   "City",
   "Country",
   "Postal Code",
   "Telephone",
   "GPA",
   "Total Credits",
 };

 public static final String[] columnTips = {
   "ID",
   "First Name",
   "Last Name",
   "Street Address",
   "State",
   "City",
   "Country",
   "Postal Code",
   "Telephone",
   "GPA",
   "Total Credits",
 };
```

```java
public static final Comparator[] comparators = {
   new StudentEntryComparator(ID_COLUMN),
   new StudentEntryComparator(FIRST_NAME_COLUMN),
   new StudentEntryComparator(LAST_NAME_COLUMN),
   new
StudentEntryComparator(STREET_ADDRESS_COLUMN),
   new StudentEntryComparator(STATE_COLUMN),
   new StudentEntryComparator(CITY_COLUMN),
   new StudentEntryComparator(COUNTRY_COLUMN),
   new StudentEntryComparator(POSTAL_CODE_COLUMN),
   new StudentEntryComparator(TELEPHONE_COLUMN),
   new StudentEntryComparator(GPA_COLUMN),
   new
StudentEntryComparator(TOTAL_CREDITS_COLUMN),
 };

 public StudentEntry(String ID,
                     String firstName,
                     String lastName,
                     String streetAddress,
                     String state,
                     String city,
                     String country,
                     String postalCode,
                     String telephone,
                     float GPA,
                     int totalCredits) {
   super(ID, firstName, lastName, streetAddress, state, city,
country, postalCode,
         telephone, GPA, totalCredits);
 }
```

```
public StudentEntry(Student student) {
    if (student != null) {
      this.ID = student.ID;
      this.firstName = student.firstName;
      this.lastName = student.lastName;
      this.streetAddress = student.streetAddress;
      this.state = student.state;
      this.city = student.city;
      this.country = student.country;
      this.postalCode = student.postalCode;
      this.telephone = student.telephone;
      this.GPA = student.GPA;
      this.totalCredits = student.totalCredits;
    }
  }

  public int getColumnCount() {
    return columnNames.length;
  }

  public String getColumnName(int col) {
    if (col >= 0 &&
              col < columnNames.length) {
      return columnNames[col];
    }
    return null;
  }
```

```
public Object getColumnValue(int col) {
  if (col >= 0 &&
            col < columnNames.length) {
    switch (col) {
    case ID_COLUMN:             return ID;
    case FIRST_NAME_COLUMN:     return firstName;
    case LAST_NAME_COLUMN:      return lastName;
    case STREET_ADDRESS_COLUMN: return streetAddress;
    case STATE_COLUMN:          return state;
    case CITY_COLUMN:           return city;
    case COUNTRY_COLUMN:        return country;
    case POSTAL_CODE_COLUMN:    return postalCode;
    case TELEPHONE_COLUMN:      return telephone;
    case GPA_COLUMN:            return new Float(GPA);
    case TOTAL_CREDITS_COLUMN:  return new Integer(totalCredits);
    }
  }
  return null;
}

public String getColumnTip(int col) {
  if (col >= 0 &&
            col < columnTips.length) {
    return columnTips[col];
  }
  return null;
}
```

```
public Class getColumnClass(int col) {
    if (col == GPA_COLUMN) {
      return Float.class;
    } else if (col == TOTAL_CREDITS_COLUMN) {
      return Integer.class;
    } else {
      return String.class;
    }
  }

  public Comparator getColumnComparator(int col) {
    if (col >= 0 &&
              col < comparators.length) {
      return comparators[col];
    }
    return null;
  }

  public int getColumnWidth(int col) {
    return -1;
  }

  static public class StudentEntryComparator implements Comparator {

    public StudentEntryComparator(int col) {
      this.col = col;
    }
    public int compare(Object o1, Object o2) {
      if (o1 != null &&
              o2 != null &&
              o1 instanceof StudentEntry &&
              o2 instanceof StudentEntry) {
            StudentEntry e1 = (StudentEntry) o1;
            StudentEntry e2 = (StudentEntry) o2;
            if (col == GPA_COLUMN) {
              return (int) (e1.getGPA() * 1000 - e2.getGPA() * 1000);
            } else if (col == TOTAL_CREDITS_COLUMN) {
              return (e1.getTotalCredits() - e2.getTotalCredits());
            } else {
              return ((String) e1.getColumnValue(col)).compareTo(e2.getColumnValue(col));
            }
      }
      return 0;
    }

    protected int col;
  }

}
```



- Figure 10.12 page 527
  Object adapter : adapt by delegation
- The design using the object adapter form
  - the StudentEntry2 class is the adapter
  - the Student class is the adaptee
  - the TableEntry interface is the target.

- Class adapter.StudentEntry2 (P. 527)

```java
package adapter;

import java.util.Comparator;

/**
 *  Adapter design pattern
 */
public class StudentEntry2 implements TableEntry {

  // position of each column
  public static final int ID_COLUMN              = 0;
  public static final int FIRST_NAME_COLUMN      = 1;
  public static final int LAST_NAME_COLUMN       = 2;
  public static final int STREET_ADDRESS_COLUMN = 3;
  public static final int STATE_COLUMN           = 4;
  public static final int CITY_COLUMN            = 5;
  public static final int COUNTRY_COLUMN         = 6;
  public static final int POSTAL_CODE_COLUMN     = 7;
  public static final int TELEPHONE_COLUMN       = 8;
  public static final int GPA_COLUMN             = 9;
  public static final int TOTAL_CREDITS_COLUMN  = 10;
public static final String[] columnNames = {
  "ID",
  "First Name",
  "Last Name",
  "Street Address",
  "State",
  "City",
  "Country",
  "Postal Code",
  "Telephone",
  "GPA",
  "Total Credits",
};

public static final String[] columnTips = {
  "ID",
  "First Name",
  "Last Name",
  "Street Address",
  "State",
  "City",
  "Country",
  "Postal Code",
  "Telephone",
  "GPA",
  "Total Credits",
};
```

```
public static final Comparator[] comparators = {
  new StudentEntryComparator(ID_COLUMN),
  new StudentEntryComparator(FIRST_NAME_COLUMN),
  new StudentEntryComparator(LAST_NAME_COLUMN),
  new StudentEntryComparator(STREET_ADDRESS_COLUMN),
  new StudentEntryComparator(STATE_COLUMN),
  new StudentEntryComparator(CITY_COLUMN),
  new StudentEntryComparator(COUNTRY_COLUMN),
  new StudentEntryComparator(POSTAL_CODE_COLUMN),
  new StudentEntryComparator(TELEPHONE_COLUMN),
  new StudentEntryComparator(GPA_COLUMN),
  new StudentEntryComparator(TOTAL_CREDITS_COLUMN),
};

public StudentEntry2(String ID,
                     String firstName,
                     String lastName,
                     String streetAddress,
                     String state,
                     String city,
                     String country,
                     String postalCode,
                     String telephone,
                     float GPA,
                     int totalCredits) {
  student = new Student(ID, firstName, lastName,          //////
                        streetAddress, state, city, country, postalCode,
                        telephone, GPA, totalCredits);
}
```

```
public StudentEntry2(Student student) {
    this.student= student;
  }

  public int getColumnCount() {
    return columnNames.length;
  }

  public String getColumnName(int col) {
    if (col >= 0 &&
                col < columnNames.length) {
      return columnNames[col];
    }
    return null;
  }
```

```
public Object getColumnValue(int col) {
  if (student != null &&
            col >= 0 &&
            col < columnNames.length) {
    switch (col) {
    case ID_COLUMN:             return student.getID();       ///////////
    case FIRST_NAME_COLUMN:     return student.getFirstName();  ///////////
    case LAST_NAME_COLUMN:      return student.getLastName();  ///////////
    case STREET_ADDRESS_COLUMN: return student.getStreetAddress();  ///////////
    case STATE_COLUMN:          return student.getState();
    case CITY_COLUMN:           return student.getCity();
    case COUNTRY_COLUMN:        return student.getCountry();
    case POSTAL_CODE_COLUMN:    return student.getPostalCode();
    case TELEPHONE_COLUMN:      return student.getTelephone();
    case GPA_COLUMN:            return new Float(student.getGPA());
    case TOTAL_CREDITS_COLUMN:  return new Integer(student.getTotalCredits());
    }
  }
  return null;
}

public String getColumnTip(int col) {
  if (col >= 0 &&
            col < columnTips.length) {
    return columnTips[col];
  }
  return null;
}
```

```
public Object getColumnValue(int col) {
  if (col >= 0 &&
            col < columnNames.length) {
    switch (col) {
    case ID_COLUMN:             return ID;
    case FIRST_NAME_COLUMN:     return firstName;
    case LAST_NAME_COLUMN:      return lastName;
    case STREET_ADDRESS_COLUMN: return streetAddress;
    case STATE_COLUMN:          return state;
    case CITY_COLUMN:           return city;
    case COUNTRY_COLUMN:        return country;
    case POSTAL_CODE_COLUMN:    return postalCode;
    case TELEPHONE_COLUMN:      return telephone;
    case GPA_COLUMN:            return new Float(GPA);
    case TOTAL_CREDITS_COLUMN:  return new Integer(totalCredits);
    }
  }
  return null;
}
```

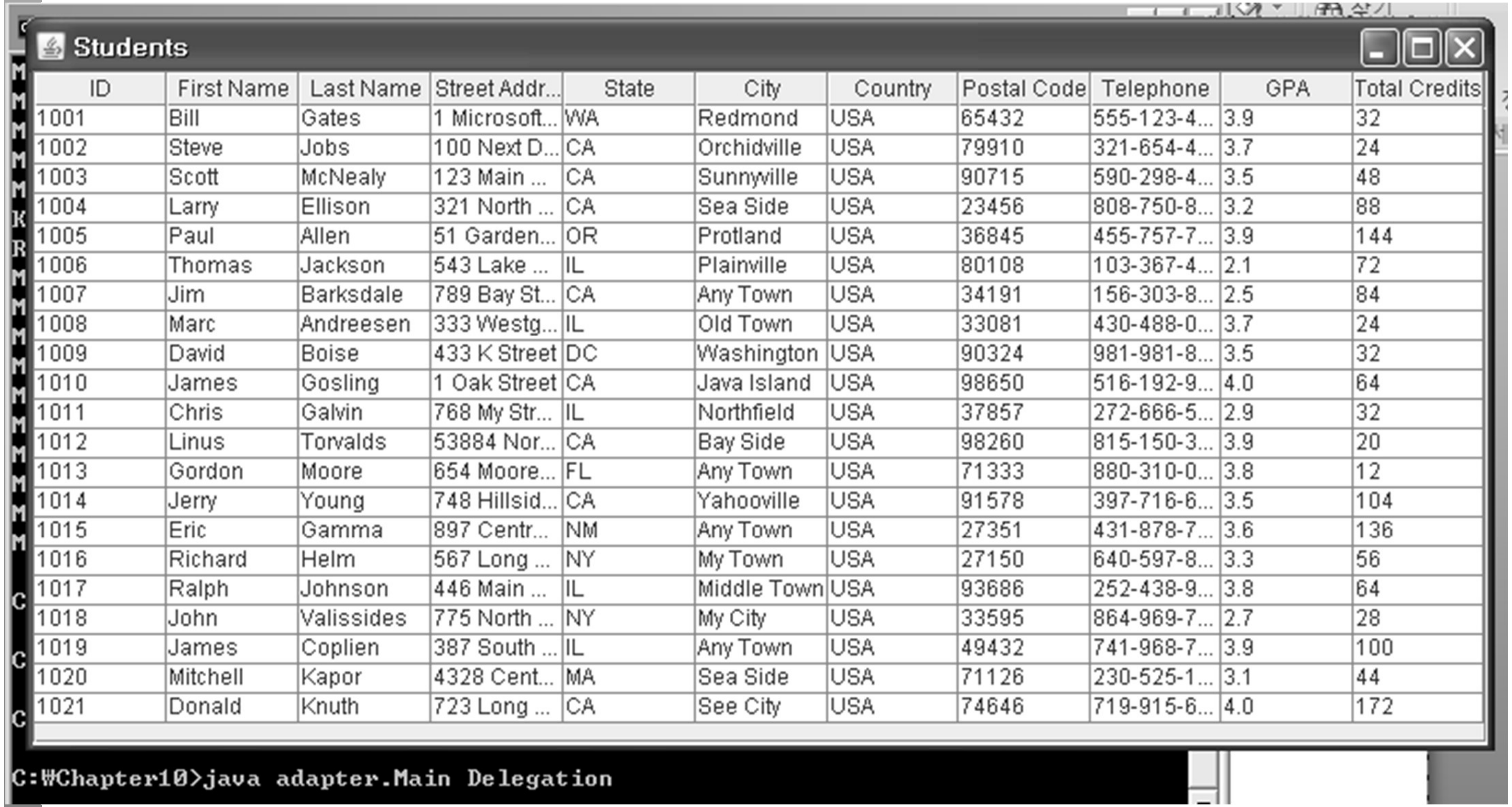
```
public Class getColumnClass(int col) {
   if (col == GPA_COLUMN) {
     return Float.class;
   } else if (col == TOTAL_CREDITS_COLUMN) {
     return Integer.class;
   } else {
     return String.class;
   }
 }

 public Comparator getColumnComparator(int col) {
   if (col >= 0 &&
              col < comparators.length) {
     return comparators[col];
   }
   return null;
 }

 public int getColumnWidth(int col) {
   return -1;
 }
```

```
protected Student student;

  static public class StudentEntryComparator implements Comparator {

    public StudentEntryComparator(int col) {
      this.col = col;
    }

    public int compare(Object o1, Object o2) {
      if (o1 != null &&
                o2 != null &&
                o1 instanceof StudentEntry2 &&
                o2 instanceof StudentEntry2) {
              StudentEntry2 e1 = (StudentEntry2) o1;
              StudentEntry2 e2 = (StudentEntry2) o2;
              if (col == GPA_COLUMN) {
                return (int) (e1.student.getGPA() * 1000 - e2.student.getGPA() * 1000);
              } else if (col == TOTAL_CREDITS_COLUMN) {
                return (e1.student.getTotalCredits() - e2.student.getTotalCredits());
              } else {
                return ((String) e1.getColumnValue(col)).compareTo(e2.getColumnValue(col));
              }
      }
      return 0;
    }

    protected int col;
  }

}
```

- The Main Class (p. 530)
  - creates a list of instances of the Student class and displays the instances using the generic table via either of the adapters.
  - The adapters are selected by an optional command-line argument.
  - If the command-line argument Delegation is present, the object adapter will be used; otherwise, the class adapter will be used.

```
package adapter;

import java.awt.Dimension;
import java.awt.Toolkit;
import java.util.*;
import javax.swing.*;

public class Main {

  static Student[] students = {
    new Student("1001", "Bill", "Gates",
                                   "1 Microsoft Way", "WA",
"Redmond", "USA", "65432",
                                   "555-123-4567", 3.9f, 32),
    new Student("1002", "Steve", "Jobs",
                                   "100 Next Drive", "CA",
"Orchidville", "USA", "79910",
                                   "321-654-4567", 3.7f, 24),
    new Student("1003", "Scott", "McNealy",
                                   "123 Main Street", "CA",
"Sunnyville", "USA", "90715",
                                   "590-298-4262", 3.5f, 48),
    new Student("1004", "Larry", "Ellison",
                                   "321 North Blvd.", "CA", "Sea
Side", "USA", "23456",
                                   "808-750-8955", 3.2f, 88),
    new Student("1005", "Paul", "Allen",
                                   "51 Garden Street", "OR",
"Protland", "USA", "36845",
                                   "455-757-7311", 3.9f, 144),
    new Student("1006", "Thomas", "Jackson",
                                   "543 Lake Ave.", "IL",
"Plainville", "USA", "80108",
                                   "103-367-4105", 2.1f, 72),
 new Student("1007", "Jim", "Barksdale",
                                   "789 Bay Street", "CA", "Any
Town", "USA", "34191",
                                   "156-303-8166", 2.5f, 84),
    new Student("1008", "Marc", "Andreesen",
                                   "333 Westgate Ave.", "IL",
"Old Town", "USA", "33081",
                                   "430-488-0931", 3.7f, 24),
    new Student("1009", "David", "Boise",
                                   "433 K Street", "DC",
"Washington", "USA", "90324",
                                   "981-981-8493", 3.5f, 32),
 new Student("1010", "James", "Gosling",
                                   "1 Oak Street", "CA", "Java
Island", "USA", "98650",
                                   "516-192-9406", 4.0f, 64),
    new Student("1011", "Chris", "Galvin",
                                   "768 My Street", "IL",
"Northfield", "USA", "37857",
                                   "272-666-5555", 2.9f, 32),
    new Student("1012", "Linus", "Torvalds",
                                   "53884 Norht Sea Drive",
"CA", "Bay Side", "USA", "98260",
                                   "815-150-3179", 3.9f, 20),
    new Student("1013", "Gordon", "Moore",
                                   "654 Moore Street", "FL",
"Any Town", "USA", "71333",
                                   "880-310-0516", 3.8f, 12),
    new Student("1014", "Jerry", "Young",
                                   "748 Hillside Blvd.", "CA",
"Yahooville", "USA", "91578",
                                   "397-716-6169", 3.5f, 104),
    new Student("1015", "Eric", "Gamma",
                                   "897 Central Street", "NM",
"Any Town", "USA", "27351",
                                   "431-878-7706", 3.6f, 136),
    new Student("1016", "Richard", "Helm",
                                   "567 Long Blvd.", "NY", "My
Town", "USA", "27150",
                                   "640-597-8608", 3.3f, 56),
    new Student("1017", "Ralph", "Johnson",
                                   "446 Main Street", "IL",
"Middle Town", "USA", "93686",
                                   "252-438-9179", 3.8f, 64),
 new Student("1018", "John", "Valissides",
                                   "775 North Blvd.", "NY", "My
City", "USA", "33595",
                                   "864-969-7578", 2.7f, 28),
    new Student("1019", "James", "Coplien",
                                   "387 South Street", "IL", "Any
Town", "USA", "49432",
                                   "741-968-7355", 3.9f, 100),
    new Student("1020", "Mitchell", "Kapor",
                                   "4328 Central Blvd.", "MA",
"Sea Side", "USA", "71126",
                                   "230-525-1849", 3.1f, 44),
    new Student("1021", "Donald", "Knuth",
                                   "723 Long Blvd.", "CA", "See
City", "USA", "74646",
                                   "719-915-6393", 4.0f, 172),
  };
```

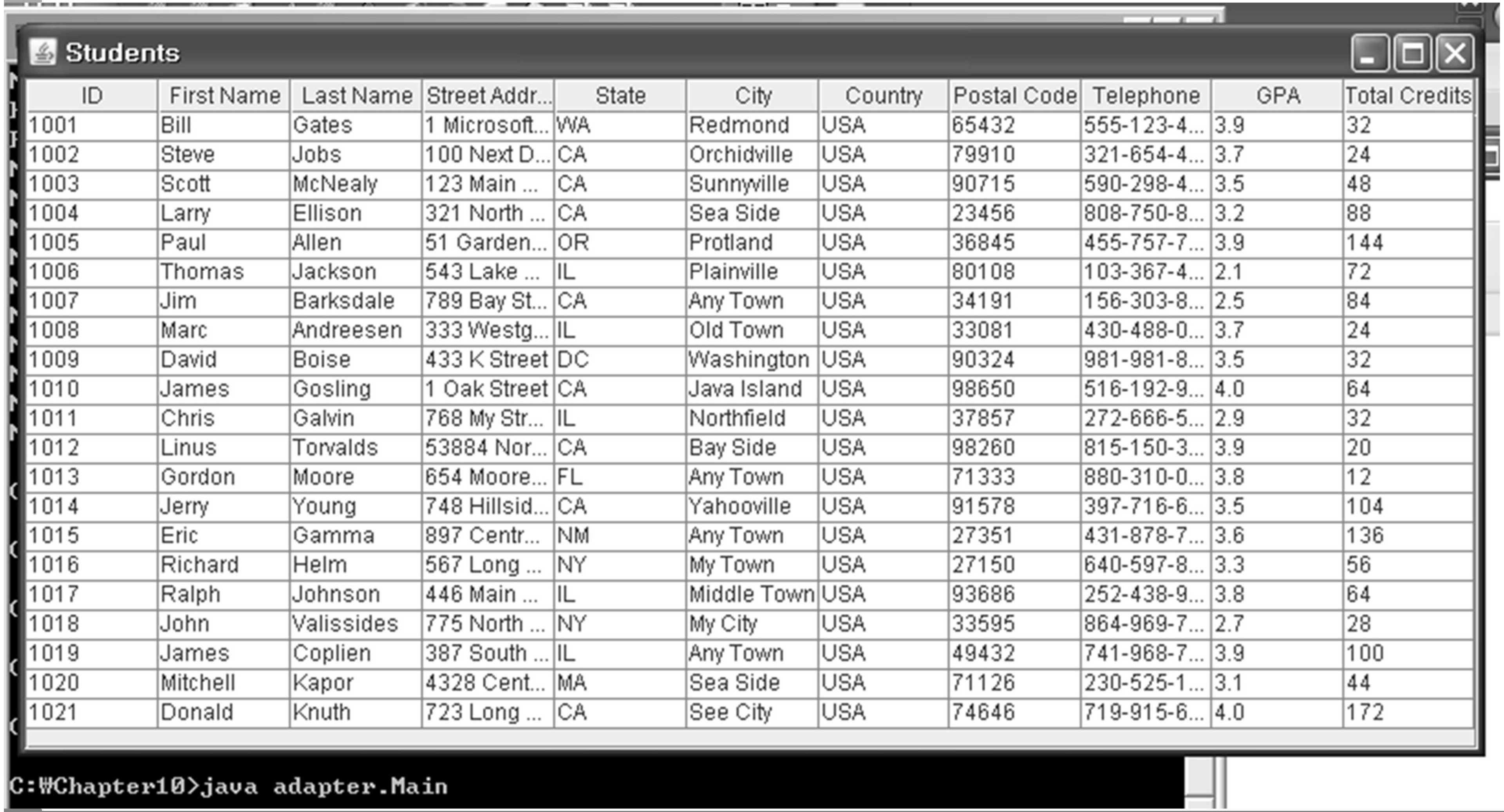
```
 public static final int INITIAL_FRAME_WIDTH = 800;
  public static final int INITIAL_FRAME_HEIGHT =
400;

  public static void main(String[] args) {
    boolean useDelegation = false;
    if (args.length > 0 &&
               "Delegation".equals(args[0])) {
      useDelegation = true;
    }

    List entries = new ArrayList(students.length);
    for (int i = 0; i < students.length; i++) {
      if (useDelegation) {
               entries.add(new
StudentEntry2(students[i]));
      } else {
               entries.add(new
StudentEntry(students[i]));
      }
    }
 Table table = new Table(entries);

   JFrame frame = new JFrame("Students");
   frame.setContentPane(new JScrollPane(table));
   frame.setSize(INITIAL_FRAME_WIDTH, INITIAL_FRAME_HEIGHT);
   Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
   frame.setLocation(screenSize.width / 2 - INITIAL_FRAME_WIDTH / 2,
                                 screenSize.height / 2 - INITIAL_FRAME_HEIGHT / 2);
   frame.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
   frame.setVisible(true);
 }

}
```



Students

ID	First Name	Last Name	Street Addr...	State	City	Country	Postal Code	Telephone	GPA	Total Credits
1001	Bill	Gates	1 Microsoft...	WA	Redmond	USA	65432	555-123-4...	3.9	32
1002	Steve	Jobs	100 Next D...	CA	Orchidville	USA	79910	321-654-4...	3.7	24
1003	Scott	McNealy	123 Main ...	CA	Sunnyville	USA	90715	590-298-4...	3.5	48
1004	Larry	Ellison	321 North ...	CA	Sea Side	USA	23456	808-750-8...	3.2	88
1005	Paul	Allen	51 Garden...	OR	Protland	USA	36845	455-757-7...	3.9	144
1006	Thomas	Jackson	543 Lake ...	IL	Plainville	USA	80108	103-367-4...	2.1	72
1007	Jim	Barksdale	789 Bay St...	CA	Any Town	USA	34191	156-303-8...	2.5	84
1008	Marc	Andreesen	333 Westg...	IL	Old Town	USA	33081	430-488-0...	3.7	24
1009	David	Boise	433 K Street	DC	Washington	USA	90324	981-981-8...	3.5	32
1010	James	Gosling	1 Oak Street	CA	Java Island	USA	98650	516-192-9...	4.0	64
1011	Chris	Galvin	768 My Str...	IL	Northfield	USA	37857	272-666-5...	2.9	32
1012	Linus	Torvalds	53884 Nor...	CA	Bay Side	USA	98260	815-150-3...	3.9	20
1013	Gordon	Moore	654 Moore...	FL	Any Town	USA	71333	880-310-0...	3.8	12
1014	Jerry	Young	748 Hillsid...	CA	Yahooville	USA	91578	397-716-6...	3.5	104
1015	Eric	Gamma	897 Centr...	NM	Any Town	USA	27351	431-878-7...	3.6	136
1016	Richard	Helm	567 Long ...	NY	My Town	USA	27150	640-597-8...	3.3	56
1017	Ralph	Johnson	446 Main ...	IL	Middle Town	USA	93686	252-438-9...	3.8	64
1018	John	Valissides	775 North ...	NY	My City	USA	33595	864-969-7...	2.7	28
1019	James	Coplien	387 South ...	IL	Any Town	USA	49432	741-968-7...	3.9	100
1020	Mitchell	Kapor	4328 Cent...	MA	Sea Side	USA	71126	230-525-1...	3.1	44
1021	Donald	Knuth	723 Long ...	CA	See City	USA	74646	719-915-6...	4.0	172

```
C:₩Chapter10>java adapter.Main Delegation
```



ID	First Name	Last Name	Street Addr...	State	City	Country	Postal Code	Telephone	GPA	Total Credits
1001	Bill	Gates	1 Microsoft...	WA	Redmond	USA	65432	555-123-4...	3.9	32
1002	Steve	Jobs	100 Next D...	CA	Orchidville	USA	79910	321-654-4...	3.7	24
1003	Scott	McNealy	123 Main ...	CA	Sunnyville	USA	90715	590-298-4...	3.5	48
1004	Larry	Ellison	321 North ...	CA	Sea Side	USA	23456	808-750-8...	3.2	88
1005	Paul	Allen	51 Garden...	OR	Protland	USA	36845	455-757-7...	3.9	144
1006	Thomas	Jackson	543 Lake ...	IL	Plainville	USA	80108	103-367-4...	2.1	72
1007	Jim	Barksdale	789 Bay St...	CA	Any Town	USA	34191	156-303-8...	2.5	84
1008	Marc	Andreesen	333 Westg...	IL	Old Town	USA	33081	430-488-0...	3.7	24
1009	David	Boise	433 K Street	DC	Washington	USA	90324	981-981-8...	3.5	32
1010	James	Gosling	1 Oak Street	CA	Java Island	USA	98650	516-192-9...	4.0	64
1011	Chris	Galvin	768 My Str...	IL	Northfield	USA	37857	272-666-5...	2.9	32
1012	Linus	Torvalds	53884 Nor...	CA	Bay Side	USA	98260	815-150-3...	3.9	20
1013	Gordon	Moore	654 Moore...	FL	Any Town	USA	71333	880-310-0...	3.8	12
1014	Jerry	Young	748 Hillsid...	CA	Yahooville	USA	91578	397-716-6...	3.5	104
1015	Eric	Gamma	897 Centr...	NM	Any Town	USA	27351	431-878-7...	3.6	136
1016	Richard	Helm	567 Long ...	NY	My Town	USA	27150	640-597-8...	3.3	56
1017	Ralph	Johnson	446 Main ...	IL	Middle Town	USA	93686	252-438-9...	3.8	64
1018	John	Valissides	775 North ...	NY	My City	USA	33595	864-969-7...	2.7	28
1019	James	Coplien	387 South ...	IL	Any Town	USA	49432	741-968-7...	3.9	100
1020	Mitchell	Kapor	4328 Cent...	MA	Sea Side	USA	71126	230-525-1...	3.1	44
1021	Donald	Knuth	723 Long ...	CA	See City	USA	74646	719-915-6...	4.0	172

```
C:₩Chapter10>java adapter.Main
```

끝