Today's topics

Java
The Database Program

Upcoming
Recursion

Reading
Great Ideas, Chapter 4

Database

- Extend ideas used in phone program
- Can have many “parallel” arrays
- Design Used Car Database

<table>
<thead>
<tr>
<th>Make</th>
<th>Style</th>
<th>Color</th>
<th>Owner</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>“Mercedes”</td>
<td>“4 door”</td>
<td>“white”</td>
<td>“M.Ramm”</td>
<td>1987</td>
</tr>
<tr>
<td>“Mercedes”</td>
<td>“4 door”</td>
<td>“ivory”</td>
<td>“D.Ramm”</td>
<td>1987</td>
</tr>
<tr>
<td>“Ford”</td>
<td>“truck”</td>
<td>“blue”</td>
<td>“M.Ramm”</td>
<td>1972</td>
</tr>
<tr>
<td>“Mercedes”</td>
<td>“4 door”</td>
<td>“green”</td>
<td>“D.Ramm”</td>
<td>1971</td>
</tr>
<tr>
<td>“Datsun”</td>
<td>“2 door”</td>
<td>“red”</td>
<td>“K.Ramm”</td>
<td>1978</td>
</tr>
<tr>
<td>“Ford”</td>
<td>“4 door”</td>
<td>“blue”</td>
<td>“D.Ramm”</td>
<td>1978</td>
</tr>
</tbody>
</table>

Basic Functions

- What do we expect of a database program?
  - Load Data (file I/O would be nice)
  - Display Data (complete dump)
  - Edit/Update (make corrections, add more data)
  - Query/Search (selective info retrieval)
- What are we actually going to implement?
  - Know no file I/O: build data in
  - Will have a way to display all
  - This program will not attempt edit/update
  - Queries are the key to making this useful

Naïve Query

- Use same approach as telephone lookup
  - Specify value for only one of the lists
    - E.g., specify Ford
      - Get all entries that are Fords
      - OK for small lists like this toy example
  - What is the problem here?
  - ?
- Similar problem for using any other single specification
More Queries

- **Other Simple Query**
  - The example has 5 lists
  - Specify what we want to match in each list
  - E.g. Ford, truck, blue, M.Ramm, 1972
    - Very limiting
    - Can really only expect a True/False response of this
    - We either get a “hit” or we do not

- **Wild Card Specification**
  - Can specify that we do not care about one or more lists
  - Adds surprising versatility!
  - E.g. Mercedes, 4 door, don’t-care, don’t-care, don’t-care
  - Yields multiple responses for this example

Database Program

```java
public class UsedCars extends java.applet.Applet implements ActionListener {
    TextField gMake, gStyle, gColor, gOwner, mInstruct;
    IntField gYear;
    String Make[] =
    String Style[] =
        {"4 door", "4 door", "truck", "4 door", "2 door", "4 door"};
    String Color[] =
        {"white", "ivory", "blue", "green", "red", "blue"};
    String Owner[] =
        {"M.Ramm", "D.Ramm", "M.Ramm", "D.Ramm", "K.Ramm", "D.Ramm"};
    int Year[] =
    Label make, style, color, owner, year;
    mInstruct = new TextField(60);
    mInstruct.setText("Enter Selections, leave blank for Wild Card");
    gYear = new IntField(10);
    gMake = new TextField(10);
    gStyle = new TextField(10);
    gColor = new TextField(10);
    gOwner = new TextField(20);
    bSearch = new Button("Search");
    mResults = new TextArea(10, 60);
}
```

```java
public void init() {
    year = new Label("Year");
    make = new Label("Make");
    style = new Label("Style");
    color = new Label("Color");
    owner = new Label("Owner");
    mInstruct = new TextField(60);
    mInstruct.setText("
Enter Selections, leave blank for Wild Card"");
    gYear = new IntField(10);
    gMake = new TextField(10);
    gStyle = new TextField(10);
    gColor = new TextField(10);
    gOwner = new TextField(20);
    bSearch = new Button("Search");
    mResults = new TextArea(10, 60);
}
```

```java
public void actionPerformed(ActionEvent event) {
    int yea;
    String mak, sty, col, own;
    yea = gYear.getInt();
    mak = gMake.getText();
    sty = gStyle.getText();
    col = gColor.getText();
    own = gOwner.getText();
    mResults.setText("Cars Matching Parameters Specified
");
    report(yea, mak, sty, col, own, size);
}
```
boolean isMatch(int year, int yea, String make, String mak, 
String style, String sty, String color, 
String col, String owner, String own)
{
    if ((year == yea || yea == 0) && 
    (make.equals(mak) || mak.equals("")) && 
    (style.equals(sty)|| sty.equals(""))) && 
    (color.equals(col)|| col.equals(""))) && 
    (owner.equals(own)||own.equals("")))
    {
        return true;
    }
    return false;
}

void report(int yea, String mak, String sty, String col, 
String own, int num)
{
    int k = 0;
    while (k < num) {
        if (isMatch(Year[k], yea, Make[k], mak, Style[k], 
sty, Color[k], col, Owner[k], own)){
            mResults.append("year: " + Year[k] + " make: " + 
Make[k] + " style: " + Style[k] + " color: " + 
Color[k] + " owner: " + Owner[k] + 
"\n");
        } 
        k = k + 1;
    }
}

Other possible Features
- Allow output of counts rather than displaying field
- Permit logical expression, not just matching
  - E.g., Ford | Chevy
- For numeric data allow ranges
  - E.g., 1999 < year < 2003
- Allow merging or joining of databases
  - This capability is called a Relational Database
  - Like having multiple database all linked together

Relational Databases
- Possible Duke Databases:
  - Students
    Name, Year, Major, Schedule, ...
  - Classes
    Dept, Number, Section, Instructor, Times, Location, Roll, ...
  - Rooms
    Building, Number, Capacity, AV, ...
  - Courses
    Dept, Number, Title, Type, Description, Prerequisites, ...
- Possible Queries (Reports):
  - Class Roll: Which of the databases above are needed?