Paul would like us to implement some sequential searches for his company. We are going to assume a very basic view of his roster.

First, let us define a class for a security Assignment called **`Assignment`**:

```
public class Assignment
{
     // instance variables
     private String time;
     private String location;
     private String person;

     /**
      * Constructor for objects of class Assignment
      */
     public Assignment(String t, String l, String p)
     {
          // initialise instance variables
          time = t;
          location = l;
          person = p;
     }
     public void setTime(String t)
     {
         time = t;
     }
     public String getTime()
     {
         return time;
     }
     public void setLocation(String l)
     {
         location = l;
     }
     public String getLocation()
     {
         return location;
     }
     public void setPerson(String p)
     {
         person = p;
     }
     public String getPerson()
     {
         return person;
     }
     public String toString()
```

```
    {
        return String.format("%-18s", time) + "\t" +
               String.format("%-10s", location) +
               "\t" + String.format("%-15s", person);
    }

}
```

Notice that we assume that a particular security Assignment only involves a **time**, **location**, and **person**. Also, note that to make the formatting in our **toString()** method easier, we use **String.format()** method. The first argument is a format string that specifies how many characters will be used to format the second argument. For example, looking at **time**, the format string it will fit into 18 spaces and be left-justified (as indicated by the – sign). Feel free to use this in your programming with strings.

Now, we need to set up the roster array so that we have items to search.

We are going to define it like this:

```
Assignment[] roster = new Assignment[15];

roster[0] = new Assignment("8:00 AM - 4:00 PM", "Safe Mart",
                           "John Tiller");
roster[1] = new Assignment("4:00 PM - 12:00 AM", "Safe Mart",
                           "Mary Ellen");
roster[2] = new Assignment("12:00 AM - 8:00 AM", "Safe Mart",
                           "DJ Turtle");
roster[3] = new Assignment("8:00 AM - 4:00 PM", "Wally World",
                           "Sue Manny");
roster[4] = new Assignment("4:00 PM - 12:00 AM", "Wally World",
                           "Julia Torte");
roster[5] = new Assignment("12:00 AM - 8:00 AM", "Wally World",
                           "Angela Ayres");
roster[6] = new Assignment("8:00 AM - 4:00 PM", "Stay Inn",
                           "Tom Beaumont");
roster[7] = new Assignment("4:00 PM - 12:00 AM", "Stay Inn",
                           "Larry Vibe");
roster[8] = new Assignment("12:00 AM - 8:00 AM", "Stay Inn",
                           "Emily Rose");
roster[9] = new Assignment("8:00 AM - 4:00 PM", "Castle Land",
                           "Aron Tropic");
roster[10] = new Assignment("4:00 PM - 12:00 AM",
                           "Castle Land", "Kyle Haney");
roster[11] = new Assignment("12:00 AM - 8:00 AM",
                           "Castle Land", "Mark Kennedy");
roster[12] = new Assignment("8:00 AM - 4:00 PM", "Kool Karts",
                           "Nina Rose");
```

```
 roster[13] = new Assignment("4:00 PM - 12:00 AM", "Kool Karts",
                             "Natalee Michaels");
 roster[14] = new Assignment("12:00 AM - 8:00 AM", "Kool Karts",
                             "Manny Rodriquez");
```

Notice that there are three time blocks for each location and only one person assigned per location.

For the first search, Paul would like to be able to search the roster and find out if a particular person is working. We have two choices of how this search should indicate whether or not it has found a person. One way is to have the method return the index where it has found the person in the array; otherwise it will return and a -1. The second way is to have the method either output the roster element it found, or a message saying the person was not in the roster. We will use the second method.

To do this, we are going to use a sequential search like the following:

```
public static void findPerson(Assignment[] r, String toFind)
{
      int found = -1;

      for(int i = 0; i < r.length; i++)
         if (r[i].getPerson().compareTo(toFind) == 0)
           {
               found = i;
               break;
           }
       if (found != -1)
        {  // we have found the person
           System.out.println("We found " + toFind +
                              " in the roster: ");
           System.out.println(r[found]);
        }
       else
           System.out.println(toFind + " is not in the roster");
}
```

Notice that we look through the array one at a time, and stop as soon as we find the person; however, if we do not find the person, we will go through the entire array. The more Assignments in the array, the longer the search may take.

A variation of this search is to find as many elements as there are that match a given search criteria. For example, Paul would like to be able to search his roster and return all the entries for a particular location. In order to do this, we cannot stop searching when we find just one roster entry. We will need to search and find them all. The nice thing is that Paul has only three shifts per location. So, if we find a location in our roster, there are three and only three listings for that

location. Therefore, we can make use of that information and stop searching after we have outputted three listings.

Here is the sequential method for that search:

```
public static void findLocation(Assignment[] r, String toFind)
{
    int found = 0;

    System.out.println("Find results: ");
    for(int i = 0; i < r.length; i++)
       if (r[i].getLocation().compareTo(toFind) == 0)
           {
               System.out.println(r[i]);
               found++;
               if (found == 3)
                  break;
           }
     if (found == 0)
     {  // we have not found the location
         System.out.println(toFind + " is not in the roster");
      }
}
```

Note that we stop the searching after we have found three listings.

That is all there is to sequential search. In fact, it reminds one of a plain traversal through the array. Now download and experiment with the demo programs.

- Download the Assignment.java and TestAssignment.java files to your module 18 demo programs directory and open them.
- Run the files and make sure you understand them before you continue.