08.01 Virtual Lecture Notes

Imagine you've been asked to implement some sequential searches for a company that schedules personnel to be present at various venues at different times.

First, let us define a class for an assignment for such an event called `Assignment`:

Assignment
<<Private Instance Variables>> - String time - String location - String person
<< Constructor >> + Assignment(String t, String l, String p) **<< Methods >>** + String toString() + String getTime() + void setTime(String t) + String getLocation() + void setLocation(String l) + String getPerson() + void setPerson(String p)

Open the **`Assignment.java`** file to compare the class implementation to the class diagram above. Notice that a particular event assignment only involves a `time`, `location`, and `person`. Also, to make the formatting in the `toString` method easier, we use `String.format` method that works similar to the `printf` method.

Now, in the client class, a roster array of the event assignments is established to provide items to search.

```
Assignment[] roster = new Assignment[10];
```

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

For the first search, we want to search the roster and find if a particular person is working. We have two choices for implementing this search and indicating whether or not a particular person is found. One way would have the method return the index where the person was found in the array; otherwise, it will return and a `-1`. The second way is to have the method either output the roster element found or a message saying the person was not in the roster. We will code the second approach.

To do this, the sequential search to find the first occurrence of a particular person would look 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; //terminates the for loop
      }
   }

   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 traverse through the array one element at a time and stop as soon as we find the desired person. If we do not find the person, we will go through the entire array. The more elements in the array, the longer the search may take.

A variation of this search is to find all elements that match given search criteria. For example, we need to search the 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.

Here is the sequential method that will find all elements for a designated location:

```
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 == 0)
```

```
    { // we have not found the location
      System.out.println(toFind + " is not in the roster");
    }
}
```

That is all there is to sequential search. In fact, it reminds one of a plain traversal through the array. If you haven't done so already, open and run the **`TestAssignment1`** program. Look closely at the output. What adjustments would you like to make to the searches to provide you with information you want to know?

Print