Binary Search et al

Dec 6

Make a Random String

just because

- Suppose I wanted to make a string composed of random characters of some length.
  - HOW???
  - Start with an array of chars and pick randomly from that
    - HOW?
  - Start with a string and pick randomly from that
    - HOW?
  - Use ASCII!
    - HOW?

Writing Comments

- Code should be commented in 3 ways
  - Every file/class should have a top level comment explaining what it does and why it exists

- Every instance variable should have a comment about what it does
  - maybe just one line with //

- Every method should have a comment with:
  - summary description of what it does
  - description of each param
  - description of return value
    - very simple function may not need this
    - Javadoc style
      - See Utils.java

Finding things

can we do this efficiently?

- Have looked for stuff a LOT, usually the max
- Slightly different problem
  - determine if an object with a value is in a list
- Basic Algorithm

```
let arr = array of integers
let target = an integer (the thing to find)
for ii in 0..arr.length
    if arr[ii]==target
        return TRUE
return FALSE
```

Finding Things

Basic Algorithm

- Need to go though whole list
  - Average Case vs Worst Case
- How can I do better!!!

```
let arr = array of integers
let target = an integer
for ii in 0..arr.length
    if arr[i]==target
        return TRUE
return FALSE
```

Binary Search

On a sorted list

```
let arr = array of integers
let target = an integer
let lo = 0
let hi = arr.length-1
While (lo < hi)
    let mid = (lo+hi) / 2
    if arr[mid]==target
        return TRUE
    if arr[mid] < target
        lo = mid + 1
    else
        hi = mid - 1
// end while
return FALSE
```

Practice

How many things to you need to check to find 25?
What items did you look at?

0 0	0 0	0 0	0 0
1 0	1 2	1 2	1 5
2 1	2 2	2 2	2 5
3 1	3 2	3 3	3 6
4 5	4 25	4 9	4 11
5 11	5 26	5 11	5 15
6 11	6 26	6 15	6 16
7 12	7 26	7 15	7 17
8 14	8 27	8 19	8 20
9 16	9 29	9 21	9 22
10 19	10 31	10 21	10 24
11 21	11 32	11 22	11 25
12 23	12 34	12 23	12 28
13 23	13 36	13 24	13 31
14 24	14 37	14 24	14 33
15 25	15 38	15 24	15 34
16 32	16 39	16 24	16 35
17 36	17 41	17 24	17 38
18 37	18 43	18 24	18 41
19 41	19 45	19 25	19 42
20 41	20 47	20 48	20 47
21 46	21 47	21 49	21 48

```
let arr = array of integers
let target = an integer
let lo = 0
let hi = arr.length-1
While (lo < hi)
    let mid = (lo+hi) / 2
    if arr[mid]==target
        return TRUE
    if arr[mid] < target
        lo = mid + 1
    else
        hi = mid – 1
// end while
return FALSE
```

Binary Search

Using Recursion!!!

```
public Flight findFlightWithOrigin(String orig) {
    return ffwoHelper(orig, 0, flights.length - 1);
}

private Flight ffwoHelper(String orig, int lo, int hi) {
    if (lo > hi) {
        return null;
    }
    int mid = (lo + hi) / 2;
    int ii = flights[mid].getOriginCity().compareTo(orig);
    if (ii == 0) {
        return flights[mid];
    }
    if (ii < 0) { // mid < dest  so can eliminate the bottom
        return ffwoHelper(orig, mid + 1, hi);
    } else {
        return ffwoHelper(orig, lo, mid - 1);
    }
}
```

Classes can have instances of other classes

```
public class Flight {
    private int year;
    private int month;
    private int date;
    private String flightNumber;
    private int departDelay;
    private int arriveDelay;
    private String originCity;
    private String destCity;
    public String getDate() {
        return year + "/" + month +
"/" + date;
    }

}
```

For instance, replace this with in instance of

```
public class GTDate {
   int month;
   int day;
   int year;

   public GTDate(int mt, int dt, int yr) {
        month = mt;
        day = dt;
        year = yr;
    }

    // Override
    public String toString() {
        return month + "/" + day + "/" + year;
    }
    //Override
    public boolean equals(Object ob) { ... }
}
```

Overriding .equals

!!!!

- Every class / object has a .equals method
  - default is same as ==
- Like toString, you can write your own

```
public boolean equals(Object ob) {
        if (ob instanceof String) {
            String oString = (String) ob;
            //return oString.equals(toString()); // works but fragile
            // break the string up into components
            String[] ss = oString.split("/");
            if (ss.length == 3) {
                try {
                    int d=Integer.parseInt(ss[0]);
                    int m = Integer.parseInt(ss[1]);
                    int y = Integer.parseInt(ss[2]);
                    return y==year && m==month && d==day;
                } catch (Exception ee) {
                    return false;
                }
            } else {
                return false;
            }
        }
        if (ob instanceof GTDate) {
            GTDate gtd = (GTDate) ob;
            return month == gtd.month && day == gtd.day && year == gtd.year;
        }
        return super.equals(ob);
    }
```

Classes can have instances of other classes

```
public class Flight {
    private GTDate theDate;
    private String flightNumber;
    private int departDelay;
    private int arriveDelay;
    private String originCity;
    private String destCity;
    public String getDate() {
        return theDate.toString();
    }

}
```

What about the constructor?

Other Usage?