Lec 13 handout

```
public class ArrayFullException extends Exception {
    public String elem;
    public ArrayFullException(String elem) {
          this.elem = elem;
    }
}


public void addLast(String newItem)                          {
   if (this.isFull()) {


   }
   // . . . rest of addLast code . . .
}

public static void exceptionExample() {
   ArrList arr = new ArrList(2);

   try {
       arr.addLast("a");
       arr.addLast("b");
       arr.addLast("c"); // Will fail
   } catch (ArrayFullException e) {


   }
}
```

```
public class MyData {
    public int[] someData;

    public MyData() {
       this.someData = new int[1000];
    }

    public void addFromFile(String filename) throws IOException {
       BufferedReader reader = new BufferedReader(new FileReader(filename));
       // ... add some data from file...
       reader.close();
    }
}


public static void main(String[] args) {
   String myFile = args[0];
   MyData d = new MyData();
   try {
       d.addFromFile(myFile);
   } catch (FileNotFoundException e) {
       System.out.println("Could not find file " + myFile);
   } catch (IOException e) {
       System.out.println("Something went wrong" + e.getMessage());
   }
}
```

Review: Properties of ArrList / ArrayLists

Activity: Three Design Exercises

Design problem #1: A professor is trying to manage enrollments for several lab sections (numbered 01 through 14). For each lab, the professor needs to store the capacity of the room and the number of students in the lab. Propose specific data structures to organize this information.

Design problem #2: A department is trying to manage enrollments in several courses (numbered 1000 through 1999). For each course, the department needs to store the capacity of the room and the number of students in the course. Propose specific data structures to organize this information.

Design problem #3: A professor is trying to manage enrollments for multiple lab sections, each labeled with the day of week and start time (such as Mon 8-10, Tues 4-6, etc). For each lab, the professor needs to store the room where the lab is meeting.

Working with HashMaps

```
// Map lab times to room numbers
HashMap<String, String> labRooms = new HashMap<String, String>();

// Associate this key with this value
labRooms.put("Mon 4-6", "CIT219");
labRooms.put("Tue 6-8", "CIT501");

labRooms.get("Mon 4-6"); // Returns "CIT219"

// Changes the value mapped to this key
labRooms.put("Mon 4-6", "CIT444");
labRooms.get("Mon 4-6");

labRooms.get("Wed 8-10");

if(labRooms.containsKey("Mon 4-6")) {
   // . . .
}
```

Extra: Speeding Up Access to Labels (eg. student ID numbers)

Extra: addFirst on ArrList strikes back

```
public class ArrTest1 {
   ArrList flavors1 = new ArrList(4);
   flavors.addLast("mint");
   flavors.addLast("grape");

   // flavors.addFirst("orange");
}
```

ArrList end: 1 eltcount: 2
“mint”
“grape”