Announcements & Reminders

- Project 6 milestone is due **today**!
- Project 6 final due April 1st. Ask questions and spend time on this project -- do not underestimate this project or any other project here on out.
- Project 7 will be put out the morning of April 2nd.
- Exam 2 Multiple Choice Stats: avg. - 25.7, std. dev. - 5.15, median - 26, max. - 38, min. - 14; programming questions yet to be graded.
Basics

- Arrays are an indexed collection of data values all of the same type

e.g. An array of length 20 -
Arrays of Primitive Data Types

- **Array Declaration**
  
  1. `<data type> [ ] <variable>` //variation
  2. `<data type> <variable>[ ]` //variation

- **Array Creation**

  `<variable> = new <data type>`

  **Variation 1**
  ```
  double[ ] rainfall;
  rainfall = new double[12];
  ```

  **Variation 2**
  ```
  double rainfall [ ];
  rainfall = new double[12];
  ```
Arrays of Objects

- In Java, in addition to arrays of primitive data types, we can declare arrays of object references.
- An array of primitive data is a powerful tool, but an array of object references is even more powerful.
  - Let’s look at an example
Array data type

- An array with elements of type \texttt{GraduateStudent} is a new data type represented as \texttt{GraduateStudent[ ]}
  - \texttt{int [ ] age;}
  - \texttt{double salary[ ]; //less than minimum wage}
  - \texttt{Person student[ ];}

  - age is of type \texttt{int[ ]}
  - salary is of type \texttt{double [ ]}
  - student is of type \texttt{Person [ ]}

- Each element of this array is of type \texttt{GraduateStudent}
  - age[0] is an \texttt{int}. salary[3] is a \texttt{double}.
  - student [1] is a \texttt{Person}. 
Array Initialization

- It is possible to declare and initialize an array at the same time.

```java
int[] number = { 2, 4, 6, 8 };

double[] samplingData = { 2.443, 8.99, 12.3, 45.009, 18.2, 9.00, 3.123, 22.084, 18.08 };

String[] monthName = {
    "August", "September", "October", "November", "December"
};
```

- It is also common to initialize an array using a for loop.
An index for an array can move between indices \( 0 \) and \( \text{length} - 1 \) at most.

An index outside of this range will throw an `ArrayIndexOutOfBoundsException`.

This does not need to be caught, but the program will terminate if it’s not.

Note:
- Arrays have a public constant length which will tell you the length of an array
  - `Person[] a = new Person[17];`
  - `a.length == 17;`
Array example 1

```java
int[] values = {0, 1, 1, 2, 3, 5, 8, 13, 21, 34};
String[] names = new String[5];
names[1] = "Jupiter";
names[3] = "Pluto";
for(i=0;i<values.length;++i)
    System.out.println(values[i] + " ");

System.out.println(names[values[values[values[3]]]]);
```
Array Example2

- Assume we have an array of double called numbers which has already been initialized.
- The average of the numbers is computed here-

```java
double sum = 0;
for(int i = 0; i < numbers.length; i++){
    sum += numbers[i];
}

double average = sum/numbers.length;
```
public class Brady{
    private String name;
    private int age;

    public Brady(String n, int a){
        name = n;
        age = a;
    }

    public String getName(){
        return name;
    }

    public int age(){
        return age;
    }
}

Brady[] bunch = new Brady[6];
bunch[0] = new Brady("Bobby", 8);
bunch[1] = new Brady("Cindy", 7);
bunch[2] = new Brady("Peter", 14);
bunch[3] = new Brady("Jan", 13);
bunch[4] = new Brady("Marcia", 15);
bunch[5] = new Brady("Greg", 16);

for(int i = 0; i < bunch.length; i++)
{
    System.out.println("Name: " +
                        bunch[i].getName() +
                        " Age: " +
                        bunch[i].getAge());
}

In Java, we are not required to declare the size at compile time.

The size of an array can be specified as a variable as shown below:

```java
int size;
int[] number;
size = Integer.parseInt(JOptionPane.showMessageDialog(null,
        "Size of an array:");
number = new int[size];
```
Passing Arrays to Methods

- When arrays are passed as arguments, the address of the array is copied to the parameter.
- Example:
  ```java
  float arrayOne[] = new float[10];
  ```

  ```java
  public int searchMinimum(float[] number))
  {
    ...
  }
  ```

  ```java
  minOne = searchMinimum(arrayOne);
  ```
Arguments and Return Values

- An array can be returned by a method.
- The return type must be an array in this case.

```java
public int[] evenIntValues(int[] inArray)

int[] array = new int[50];
int[] y = evenIntValues(array);
```
Two-Dimensional Arrays

- Two-dimensional arrays are useful in representing tabular information.
- 2-dimensional arrays are usually represented in a row-column approach.

<table>
<thead>
<tr>
<th>Multiplication Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
</tbody>
</table>
Declaring and Creating a 2-D Array

- Declaration
  - `<data type> [][] <variable>`  //variation 1
  - `<data type>   <variable>[][]`  //variation 2

- Creation
  - `<variable> = new <data type> [ <size1> ][ <size2> ]`

- Example

```java
int[][] array;
array = new int[4][5];
```
More on 2-d arrays

- A 2-d array is essentially an array of arrays

- The subarrays may have different lengths
2-d arrays example

- Subarrays of different lengths
- Executing

```java
triangularArray = new double[4][];
for (int i = 0; i < 4; i++)
    triangularArray[i] = new double[i + 1];
```

results in an array that looks like:
/*assume numRows and numCols are greater than 3*/

int[][] a = new int[numRows][numCols];

for(int i =0; i<numRows; i+=2){
    for(int j=0; j<numCols; j++){
        a[i][j] = 1;
    }
}

for(int[] row : a){
    for(int col0fRow : row){
        System.out.print(col0fRow);
    }
    System.out.println();
}

for(int i=1; i < numRows-1; i++){  
    for(int j=1; j < numCols-1; j++){  
        System.out.println("at cell ("+i+","+j+")");  
        if(a[i-1][j] == 1){  
            System.out.println("see 1 at top");  
        }  
        if(a[i+1][j] == 1){  
            System.out.println("see 1 at bottom");  
        }  
    }
    System.out.println();
}
Limitation of Arrays

- Once an array object is created, its size is fixed -- it cannot be changed.
- To overcome this, we can use Lists or Maps.
Lists and Maps

- The `java.util` standard package contains different types of classes for maintaining a collection of objects.
- These classes are collectively referred to as the *Java Collection Framework (JCF)*.
- JCF includes classes that maintain collections of objects as sets, lists, or maps.
The List interface

- The **List** interface supports methods to maintain a collection of objects as a linear list.
- We can add to, remove from, and retrieve objects in a given list.
- A list does not have a set limit to the number of objects we can add to it.
- The **ArrayList** class uses an array to manage data.
- The **LinkedList** class uses a technique called *linked-node representation*. 
Here are a few list methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>boolean add (Object o)</code></td>
<td>Adds an object o to the list</td>
</tr>
<tr>
<td><code>void clear</code></td>
<td>Clears this list, i.e., make the list empty</td>
</tr>
<tr>
<td><code>Object get (int idx)</code></td>
<td>Returns the element at position idx</td>
</tr>
<tr>
<td><code>boolean remove (int idx)</code></td>
<td>Removes the element at position idx</td>
</tr>
<tr>
<td><code>int size</code></td>
<td>Returns the number of elements in the list</td>
</tr>
</tbody>
</table>
List example 1

```java
list = new ArrayList();

list.add("a"); // Append an element to the list

// list - (a)

list.add(0, "b"); // Insert an element at the head of the list

// list - (b,a)

int size = list.size(); // Get the number of elements in the list

// size - 2
```
List example 1 (contd.)

Object element = list.get(list.size()-1);    // Retrieving the element at the end of the list
   // element returned- a

list element = list.get(0);    // Retrieving the element at the head of the list
   // element returned- b

boolean b = list.remove("b");    // Remove the first occurrence of an element
   // b - true, list – (a)

element = list.remove(0);       // Remove the element at a particular index
   // list – (), element - a
JCF Maps

- JCF includes the **Map** interface that supports methods to maintain a collection of objects (key, value) pairs called map entries.

<table>
<thead>
<tr>
<th>key</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_0$</td>
<td>$v_0$</td>
</tr>
<tr>
<td>$k_1$</td>
<td>$v_1$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>$k_n$</td>
<td>$v_n$</td>
</tr>
</tbody>
</table>

one entry
Here are a few Map methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>void clear ()</td>
<td>Clears this list, i.e., make the map empty</td>
</tr>
<tr>
<td>boolean containsKey ( Object key )</td>
<td>Returns true if the map contains an entry with a given key</td>
</tr>
<tr>
<td>Object put (Object key, Object value)</td>
<td>Adds the given (key, value) entry to the map</td>
</tr>
<tr>
<td>boolean remove ( Object key )</td>
<td>Removes the entry with the given key from the map</td>
</tr>
<tr>
<td>int size ()</td>
<td>Returns the number of elements in the map</td>
</tr>
</tbody>
</table>
Map example

Map map = new HashMap();   // hash table

// Add key/value pairs to the map
map.put("a", new Integer(1));
map.put("b", new Integer(2));
map.put("c", new Integer(3));

// Get number of entries in map
int size = map.size();   // size - 3

// Adding an entry whose key exists in the map causes
the new value to replace the old value
Object oldValue = map.put("a", new Integer(9));   // oldValue - 1
Map example (contd)

```java
int value = map.get("a");
// value - 9

// Remove an entry from the map and return the value of the removed entry
oldValue = map.remove("c");
// 3
```
Before the Quiz!

- Your next lab will consist of programming a LEGO robot!

If you’ve never logged on to a CS Windows machine, you’ll need to:

- Retrieve your initial password from [http://portals.cs.purdue.edu](http://portals.cs.purdue.edu) (after logging in with your career account, see “My Accounts” on the left)
- Go to LWSN B160 & try to log on with your initial password
- You’ll be asked to change your Windows account password (it’s different from your password on your lore account)
- Be sure to change your password & remember it!
Quiz

- How would I instantiate an array to store 100 real value points from a 3-dimensional space?