Logistics

- JAVA- Resources- Prof. Tessier ECE122

http://www ecs umass edu/ece/ece122/

- Lectures
- HW
- Projects
- Other Resources
# Logistics

<table>
<thead>
<tr>
<th>Month</th>
<th>Mon</th>
<th>Wed</th>
<th>Thu</th>
<th>Fri</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sep</td>
<td></td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>16</td>
<td>17</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>23</td>
<td>24</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>30</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Oct</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>28</td>
<td>29</td>
<td>30</td>
</tr>
<tr>
<td>Nov</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td></td>
<td>No class</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>25</td>
<td></td>
<td>No class</td>
</tr>
<tr>
<td>Dec</td>
<td>30</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td></td>
<td>9</td>
<td>10</td>
</tr>
</tbody>
</table>

- **New HW**
- **New project**
- **project due at 11:50pm**
- **Mid-term**
- **Review session for Mid-Term or Final**
Logistics

- TA Office hours (Tuesday - M5) (you can go to any TA for help)
  - 10-11:00 Fubao fubaowu@umass.edu
  - 11:30-12:30 Guoyi guoyi@umass.edu
  - 1-2:00 Braegan bspring@student.umass.edu

- E-mail: TA will respond to all technical e-mails (project/HW, etc.)

- My Office hours: Friday 11:15-12:05 – Marcus 201C

- Learning review sessions – Francesca - TBD

- Projects:
  - Need a partner? Decisions after this class
  - You are allowed to change partner for the next projects (put your name on the ballot again if needed)
Arrays: Introduction

- The **most commonly used data storage structure**: Built into most programming language
- Arrays have **fixed size** (Java indexes from 0...length-1)
- Arrays are collections of values of identical type
- Arrays can be multidimensional 1d arrays [], 2d arrays [[]], etc.
- Arrays can be Unordered or Ordered that will affect the efficiency of searching algorithms for example.
Creating Arrays

- There exists two kinds of data: **primitives** (int, double, etc.) or **objects**

- In java Arrays are objects (they are primitive types in many other languages: C++, etc.). The **new** operator must be used.

  ```java
  int[] intArray; // defines a reference to an array
  intArray = new int[10]; // creates the array, and
  // sets intArray to refer to it
  ```

  which is equivalent to:

  ```java
  int[] intArray = new int[10];
  ```

- Since a Array is a object;
  - intArray holds only the addresses of the Array elements (not the elements themselves – stored somewhere else in memory)
  - Arrays have a length field which can be used to determine their (fixed) size

  ```java
  int n = intArray.length; // find array size
  ```
Initialization of Arrays

- **Global**
- **Elements by elements**
  
  ```java
  intArray[0]=0;
  intArray[1]=3;
  etc.
  ```

- **Input from command line**

  ```java
  EasyIn easy = new EasyIn();
  int[] intArray = new int[10];
  for(int i=0; i<10; i++)
  {
      intArray[i] = easy.readInt();
  }
  ```

- **Input from file**

  ```java
  File file = new File(filename);
  Scanner scan = new Scanner(file);
  int[] intArray = new int[10];
  int i=0;
  while(scan.hasNext() && i<10)
  {
      intArray[i]=scan.nextInt();
      if (scan.hasNext()) scan.nextLine();
      i++;
  }
  ```
Basic Operations on Arrays

- **Insertion** - insertion of new item at the end (should not exceed the fixed size !)
- **Permutation of 2 items** (use temp variable)
- **Shifting items** (e.g. shift up- 5 first elements)

```java
int temp = intArray[3];
IntArray[3] = intArray[4];
IntArray[4] = temp;
```

```java
for(int i=5; i>=1; i-- )
{
    intArray[i+1] = intArray[i];
}
```

- **Searching** - step through the array until item is found
- **Deletion** -
  - Begins with a search for the item to be removed
  - If item is found, shift all the items with higher index values down (shift down)

**To do**: Test Java applet Array.html
Unordered Arrays

Search: $N/2$ comparisons
Insertion: No comparisons, one move
Deletion: $N/2$ comparisons, $N/2$ moves
Object Oriented Arrays

Main Goals:

- Focus on ”What to do” rather than ”How to do it” (abstraction)
- Class takes responsibility for handling index numbers privately
- Provide higher level interfaces to handle any data structures – organize more complex data (not limited to arrays)

HighArray array = new HighArray(10); vs int[] intArray = new int[10];
Applications often need to store Array of Objects (rather than Primitives variables)
Ordered Arrays

- The N Data items are stored in ascending or descending order
- Why use ordered array?
  - Searching an unordered array is rather slow (N/2 comparisons)- it uses a "linear search" algorithm
  - Searching an ordered array is very fast using the "binary search" algorithm (an example of finely tuned data structure to improve the efficiency of an algorithm)
- Drawback
  - Insertion takes longer- all data with higher key values must be shifted up
  - Ordered arrays are then useful in situation where search are frequent but insertions and deletions are not.
    - Good situation: database of company employees.
    - Bad situation: retail store inventory
Binary Search

- **Analogy:** The *guess-a-number* game!

  - Choose a number between 1-100

  - Ok Let me guess....
    - (1-100) 50? .................................. nope too high
    - (1-49) 25? .................................. nope too low
    - (26-49) 37? .................................. nope too high
    - (26-36) 31? .................................. nope too low
    - (32-36) 34? .................................nope too high
    - (32-33) 32? .................................nope too low
    - (33-33) 33? .................................Correct!

but you are not a good mind reader