Announcements

• Introductory/Eclipse help session
  – Option 1: Wednesday, 1/23, 1:20-2:20pm
  – Option 2: Friday, 1/25, 4-5pm

• Tomasz’s office hour
  – Tu 5:15-6:15pm
  – W 12:20-1:20pm
  – F 2-3pm

• Hw1
  – To be assigned today, due next Tuesday
Roadmap

• Java Review/Basics
  – Types, variables, assignments, expressions
  – Control flow statements
  – Methods

• Arrays
  – Array basics
  – Binary Search Algorithm
  – Binary Search program
  – Hw1: number guessing game
  – ArrayList

• OO and Inheritance
Arrays

- Array is a data structure that stores a sequence of values that are of the same type

```java
double[] myList = new double[10];
```

<table>
<thead>
<tr>
<th>Array element at index 5</th>
<th>Element value</th>
</tr>
</thead>
<tbody>
<tr>
<td>myList[0]</td>
<td>5.6</td>
</tr>
<tr>
<td>myList[1]</td>
<td>4.5</td>
</tr>
<tr>
<td>myList[2]</td>
<td>3.3</td>
</tr>
<tr>
<td>myList[3]</td>
<td>13.2</td>
</tr>
<tr>
<td>myList[4]</td>
<td>4</td>
</tr>
<tr>
<td>myList[5]</td>
<td>34.33</td>
</tr>
<tr>
<td>myList[6]</td>
<td>34</td>
</tr>
<tr>
<td>myList[7]</td>
<td>45.45</td>
</tr>
<tr>
<td>myList[8]</td>
<td>99.993</td>
</tr>
<tr>
<td>myList[9]</td>
<td>11123</td>
</tr>
</tbody>
</table>

Array reference variable

myList reference
Java Array

• Initialization
  1. For primitive data types, all elements are initialized to be zero by default.
     – But you can explicitly initialize the array by providing the list of elements:

     ```java
     int[] myarray = {2,3,5,7,9,11,13,17};
     ```
     – This will both create the array and initialize the elements.
Java Array

• Initialization

  2. For class types, all elements are initialized to be **null**

Apple[] appArray = new Apple[100];
– At this point, all elements in **appArray** are null.
– To create object for each element, you can do:
  for (int i=0; i<appArray.length; i++) {
    appArray[i] = new Apple();
  }
The Length of an Array

Once an array is created, its size is fixed. It cannot be changed. You can find its size using

arrayRefVar.length

For example:

```
int[] numbers = new int[10];

int len = numbers.length; // 10
```
Accessing Array Elements

• The array elements are accessed through the index.

• Each element in the array is represented using:

  \[ \text{arrayRefVar}[\text{index}] \]

• The array indices starts from 0 to \text{array.length-1}

• Example:

```java
double[] myList = new double[10];
myList[0] = 5.6;
myList[1] = 4.5;
myList[10] = 3.33; // ArrayIndexOutOfBoundsException
```
Using for loops to process arrays

- Using for loops to traverse or process array elements
  - Elements are of the same type and processed repeatedly
  - The size of the array is known

```java
for (int i=0; i<myList.length; i++) {
    // process ith element myList[i]
    System.out.println(myList[i]);
}
```
Typical Array Processing Code

- Finding maximum/minimum of array values
- Computing the average/sum of array values
- Copy to another array
- Reverse the elements within an array
Typical Array Processing Code

• Finding maximum of array values

double max = a[0];
for (int i=0; i< a.length; i++)
    if (a[i] > max)
        max = a[i];
Typical Array Processing Code

• Compute the sum of array values

double sum = 0;
for (int i=1; i< a.length; i++)
    sum += a[i];
Typical Array Processing Code

• Copy values to another array

double[] b = a; //?
Typical Array Processing Code

• Copy values to another array

```java
int N = a.length;
double[] b = new double[N];
for (int i=0; i<N; i++)
    b[i] = a[i];
```
Problem

• Given an array of elements, find the count of elements that are smaller than the element immediately following it. For example, if the array consists of 3, 7, 8, 5, 4, 9, the method should return 3, because 3<7, 7<8, and 4<9.
Algorithm

1. Initialize counter to 0
2. for each element, compare it with the one following it, if smaller, increase counter by 1
public int inorder(int[] a) {
    int counts = 0;
    for (int i=0; i<a.length; i++)
        if (a[i] < a[i+1])
            counts ++;
    return counts;
}
Correct solution

public int inorder(int[] a)
{
    int counts = 0;
    for (int i=0; i<a.length-1; i++)
        if (a[i] < a[i+1])
            counts ++;
    return counts;
}

Operations on Array

• Provided operations by Java Array
  – Accessing or updating a particular item by index

• Desired operations
  – insert an item
  – Delete an item
  – search an item with a particular value (key)
Search in an Array

• How many steps does it take to search for a key in an array of size N?
Search in an Array

• How many steps does it take to search for a key in an array of size N?
  – Best case: 1
  – Worst case: N
  – Average case: N/2

• What if there are duplicates and we need to search for all matching keys?
Search in an Array

• How many steps (comparisons) does it take to search for a key in an array of size $N$?
  – Best case: 1
  – Worst case: $N$
  – Average case: $N/2$

• What if there are duplicates and we need to search for all matching keys?
  – Always $N$
Ordered Array

• An array in which data elements are sorted in ascending values.
  – The smallest value at index 0
  – Each element is larger than its previous element

• Search
  – The main advantage of an ordered array is that it allows faster search using binary search.
Guess-a-Number Game

- I am asking you to guess a number that I am thinking of between 1 and 100
- When you make a guess, I will tell you one of three things: too large, too small, your guess is correct.

- Think about a strategy that leads to the fewest number of guesses on average.
Guess-a-Number Game

- Assuming the correct answer is 33:

<table>
<thead>
<tr>
<th>Step Number</th>
<th>Number Guessed</th>
<th>Result</th>
<th>Range of Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>1–100</td>
</tr>
<tr>
<td>1</td>
<td>50</td>
<td>Too high</td>
<td>1–49</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>Too low</td>
<td>26–49</td>
</tr>
<tr>
<td>3</td>
<td>37</td>
<td>Too high</td>
<td>26–36</td>
</tr>
<tr>
<td>4</td>
<td>31</td>
<td>Too low</td>
<td>32–36</td>
</tr>
<tr>
<td>5</td>
<td>34</td>
<td>Too high</td>
<td>32–33</td>
</tr>
<tr>
<td>6</td>
<td>32</td>
<td>Too low</td>
<td>33–33</td>
</tr>
<tr>
<td>7</td>
<td>33</td>
<td>Correct</td>
<td></td>
</tr>
</tbody>
</table>
Binary Search

- **Goal**: given a specific key, find it’s location in the order array if it exists.
// precondition: array a[] is sorted
public static int rank(int key, int[] a) {
    int lo = 0;
    int hi = a.length - 1;
    while (lo <= hi) {
        // Key is in a[lo..hi] or not present.
        int mid = lo + (hi - lo) / 2;
        if (key < a[mid]) hi = mid - 1;
        else if (key > a[mid]) lo = mid + 1;
        else return mid;
    }
    return -1;
}
Binary Search Cost

• What’s the number of steps (comparisons) for binary search in a sorted array of size $N$?
Binary Search Cost

• What’s the number of steps (comparisons) for binary search in a sorted array of size N?
  – Best case: 1
  – Worst case: log N + 1
Programs from the book

• All book code (java files) are available at ~cs171000/share/book
• Syntax highlighted version (html files) are available at http://algs4.cs.princeton.edu/code/
• Relevant programs for each lecture are linked on the class schedule page http://www.mathcs.emory.edu/~cs171000/schedule.html
public class BinarySearch {
    public static int rank(int key, int[] a) {
        int lo = 0;
        int hi = a.length - 1;
        while (lo <= hi) {
            int mid = (lo + (hi - lo)) / 2;
            if (key < a[mid]) hi = mid - 1;
            else if (key > a[mid]) lo = mid + 1;
            else return mid;
        }
        return -1;
    }
    public static void main(String[] args) {
        int[] whitelist = In.readInts(args[0]);
        Arrays.sort(whitelist);
        while (!StdIn.isEmpty()) {
            int key = StdIn.readInt();
            if (rank(key, whitelist) == -1) StdOut.println(key);
        }
    }
}
Hw1: Number guessing game

- Guess a number I am thinking of between [1000, 9999]
- When you make a guess, I will tell you the number of matching digits between your guess and my number
- Think about a strategy that leads to the fewest number of guesses on average.
Hw1 Discussion

• Read the instructions carefully
• Think before you code
• Some hints and useful classes/methods
  – Algorithm
  – ArrayList
  – Random Number generation
Hw1 Algorithm

• Initialize a list of possible numbers
• Repeat
  – Give a random guess
    • think about how to generate a random index
  – Update the list based on input matching digits
    • think about how to eliminate impossible numbers (read the hint)
• Until the list is empty or the guess is correct
ArrayList

- ArrayList – resizable-array implementation
- Use generics - parameterized types
  - Type parameters have to be instantiated as reference types
  - Autoboxing: Automatically casting a primitive type to a wrapper type
  - Auto-unboxing: automatically casting a wrapper type to a primitive type

```java
ArrayList<Integer> numbers = new ArrayList<Integer>();
numbers.add(1001);
int mynumber = numbers.remove(0);
```
ArrayList

• Useful methods
  – add(E e): Appends the specified element to the end of this list
  – size(): returns the number of elements in this list
  – remove(int index): Removes the element at the specified position in this list. Shifts any subsequent elements to the left (subtracts one from their indices)
  – get(int index): Returns the element at the specified position in this list.

```java
ArrayList<Integer> numbers = new ArrayList<Integer>();
numbers.add(1001);
numbers.add(1002);
int n = numbers.size();
int mynumber = numbers.get(0);
mynumber = numbers.remove(0);
mynumber = numbers.get(0);
```
Random Number Generation

• Method

```java
static double Math.random() - returns a random double value between [0.0, 1.0)
```

• Examples

- Generate a random number between [0, 100)
  ```java
  int index = (int) Math.random() * 100;
  ```

- Generate a random integer between [a, b)
  ```java
  int x = a+ (int) (Math.random() * (b-a));
  ```
Roadmap

• Java Review/Basics
  – Types, variables, assignments, expressions
  – Control flow statements
  – Methods
• Arrays
  – Array basics
  – Binary Search Algorithm
  – ArrayList
  – Hw1: number guessing game
• OO and Inheritance