Compsci 201
Objects, Tradeoffs, NBody

Susan Rodger
January 24, 2020
E is for …

- **Encryption**
  - Why SSH and SSL work

- **Exception**
  - A Throwable you may catch, sometimes you may rethrow
Announcements

• Assignment P0 - grace period to today 11:59pm
  • With late penalty last change one week later
• APT-1 due – now in grace period today 11:59pm
  • Do not accept after grace period
• Discussion 3 on January 27
  • Prediscussion, do before
• APT-2 due January 28
• Assignment P1 due Thursday, Jan 30
When you submit an APT

• Submit a REFLECT form for each APT

<table>
<thead>
<tr>
<th>APT</th>
<th>test code</th>
<th>Reflect</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>APT 1</td>
<td>Submitting</td>
<td>reflect</td>
<td>Tues. January 21</td>
</tr>
<tr>
<td>APT 2</td>
<td>Submitting</td>
<td>reflect</td>
<td>Tues. January 28</td>
</tr>
</tbody>
</table>

• Submit REFLECT form for each Assignment

<table>
<thead>
<tr>
<th>Name</th>
<th>Analysis</th>
<th>Reflect</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assignment P0: Simple Java</td>
<td>None</td>
<td>None</td>
<td>January 16</td>
</tr>
<tr>
<td>Assignment P1: N-Body</td>
<td>Analysis</td>
<td>Reflect</td>
<td>January 30</td>
</tr>
</tbody>
</table>
From Last Time …
Go over
WOTO: Correctness Counts

Objects from the ground up
  • What is java.lang.Object? Its methods?
    • .equals(), .toString(), later .hashCode()

Concepts in P1: Arrays, Scanners, Testing
  • Completing P1 with minimal angst

ArrayList from high to low level (mostly Friday)
  • Fits into Collections hierarchy
  • How to build it or do it yourself: diyad
For me, the differences are simple to state: Computationalists grok that models, languages and machines are equivalent.
Algorithmic Tradeoffs

• We will use a problem to understand algorithmic trade-offs and how ArrayList works
  • java.util.ArrayList is "growable array", but more!
  • What is the class, what is the package
    • Package is a collection of related classes

• Given a list of words, find the unique words
  • Algorithms with ArrayLists
  • Alternative with Set data structure
Array and ArrayList

• Array can hold primitive or Object types
  • `int[]` and `String[]` work
  • Fixed size, cannot grow

• Use `java.util` class `ArrayList` for growth, more
  https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/ArrayList.html
  • Contain object types, not primitives
  • Use `.get()`, `.set()` and not `[]` for indexing
Look at Code: ArrayListUnique

• Problem
  • Read words from a file
  • Want the unique words in sorted order
Tradeoffs: Algorithmic Approaches

- [https://coursework.cs.duke.edu/201spring20/classcode/src](https://coursework.cs.duke.edu/201spring20/classcode/src)
- Read words from a file, store in ArrayList, class is `ArrayListUnique` – why array doesn't work?
  - Tradeoffs in creating sorted list of unique words

- Algorithmic concepts with `ArrayList` methods
  - Compare three different algorithmic approaches
  - Reasoning with and learning about Java code
Method A: Add each word to a sorted list

- Code in methodA: process each word in list, add X to list of sorted, unique words
  - If X already in sorted-list? Nothing to do
  - If X greater than all words in list? Add at end
  - Some word greater than X? shift to make room

All elements in sorted order, add X to list

< X | First element > X ...

Shift to make room for X
Example: insert “egg”

“egg” comes after “cat”
Method A: How to shift to add "in middle"

- Find first element bigger than String X at index k
- Shift right end to index > k, then add X there

https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/ArrayList.html#add(int,E)

```java
public void add(int index, E element)
```

Inserts the specified element at the specified position in this list. Shifts the element currently at that position (if any) and any subsequent elements to the right (adds one to their indices).
Method A: Processing every string
• Add unique elements from list to ret, keep sorted
• Code reason: flag, break, best, worst cases ...

```java
ArrayList<String> ret = new ArrayList<>();
for (String s : list) {
    boolean done = false;
    for (int k = 0; k < ret.size(); k += 1) {
        String current = ret.get(k);
        if (current.equals(s)) {
            done = true;
            break; // break out of loop looking at ret
        }
        if (current.compareTo(s) > 0) {
            ret.add(k, s);
            done = true;
            break;
        }
    }
    if (!done) {
        ret.add(s); // add at end, greater than all so far
    }
}```
Method A: Details of shifting to add X

- Don't know about `list.add(k, X)` then …
- Shift from end to index > X, then add X

```java
if (current.compareTo(s) > 0) {
    ret.add(null); // grow with nothing
    for (int shift = ret.size() - 1; shift > k; shift -= 1) {
        ret.set(shift, ret.get(shift - 1)); // ret[k] = ret[k-1]
    }
    ret.set(k, s);
    done = true;
    break;
}
```

All elements in sorted order, add X to list:

- `< X`
- First element > X ...
- Shift to make room for X
Method B: Tradeoff: Sort first, keep unique

- There are duplicates in list, but it's sorted
  - Process sorted elements, add to end if unique
  - Use of `copy`, why for-loop starts at 1 (priming)

```java
public static ArrayList<String> methodB(ArrayList<String> list) {
    ArrayList<String> copy = new ArrayList<>(list);
    Collections.sort(copy);
    ArrayList<String> ret = new ArrayList<>();
    ret.add(copy.get(0));
    for (int k=1; k < copy.size(); k += 1) {
        String current = copy.get(k);
        if (!current.equals(ret.get(ret.size()-1))) {
            ret.add(current);
        }
    }
    return ret;
}
```
Example: Sorted with duplicates

k

copy

"cat"  "cat"  "dog"  "dog"  "dog"  "foo"
Comparing Tradeoffs: Performance

- Both `methodA` and `methodB` process every word in the list of words
  - In loop body in `methodA`, shift happens
  - Could every element be shifted every time?
    - Shift 1, then 2, then 3, then … then shift N
  - Total work done? $1 + 2 + \ldots + N$
Sort first, why is this faster?

• Both methodA and methodB process every word in the list of words
  • In loop body in methodB, NO shift happens
  • But, all strings sorted before loop

• Sorting takes $N \times \log N$ for $N$ strings
• Shifting takes $1 + 2 + \ldots + N = \frac{N(N+1)}{2}$
• If $N =$ one million? One billion operations/second
  • Sorting is 20 million, shifting is 0.5 trillion
Method C: What if we use API, other classes

- A set contains no duplicates, a TreeSet maintains unique elements in sorted order
  - Create set, contains no duplicates
  - Create ArrayList from set
  - Where are the loops?

```java
public static ArrayList<String> methodC(ArrayList<String> list) {
    TreeSet<String> set = new TreeSet<>(list);
    ArrayList<String> ret = new ArrayList<>(set);
    return ret;
}
```
What you will know …

- Which of method A, method B, method C is better?
  - It depends, but on what does it depend?

- How does method A scale as # words increases?
  - \( 1 + 2 + \ldots + N = \frac{N(N+1)}{2} \), just say no!

- What is \( \log_2(1,024) \) or \( \log_2(1,048,576) \)?
  - Well, \( 2^{10} = 1024 \) so …
Why is methodA slow?

- Add unique elements from list to ret, keep sorted
- Code reason: flag, break, best, worst cases …

```java
ArrayList<String> ret = new ArrayList<>();
for(String s : list) {
    boolean done = false;
    for(int k=0; k < ret.size(); k += 1) {
        String current = ret.get(k);
        if (current.equals(s)) {
            done = true;
            break; // break out of loop looking at ret
        }
        if (current.compareTo(s) > 0) {
            ret.add(k, s);
            done = true;
            break;
        }
    }
    if (! done) {
        ret.add(s); // add at end, greater than all so far
    }
}
```
Java Concepts

• Loops execute until loop-guard is false
  • `break` exits loop early
  • `continue` re-checks guard, skipping body

• Some loops need initialization before loop guard
  • aka "priming the loop", e.g., `done = false`

• `if (!(done))` same as `if (done == false)`
Tradeoff: Sort first, keep unique

- There are duplicates in list, but it's sorted
  - Process sorted elements, add to end if unique
  - Use of `copy`, why for-loop starts at 1 (priming)

```java
public static ArrayList<String> methodB(ArrayList<String> list) {
    ArrayList<String> copy = new ArrayList<>(list);
    Collections.sort(copy);
    ArrayList<String> ret = new ArrayList<>();
    ret.add(copy.get(0));
    for(int k=1; k < copy.size(); k += 1) {
        String current = copy.get(k);
        if (!current.equals(ret.get(ret.size()-1))) {
            ret.add(current);
        }
    }
    return ret;
}
```
WOTO

Measurement and Analysis

• We measured runtimes empirically
  • Same on laptop tomorrow? Next year?
  • What about your computer, super computer?

• Mathematical analysis of runtimes
  • Machine independent
  • Compare algorithms without timing them!
Analysis via Pictures

- Reverse alphabetical order, shift all strings
  - Shift 1, then 2, then …, finally N strings
- $1+2+ \ldots + N = \frac{N(N+1)}{2}$
  - Roughly $N^2$
  - Square with side N?

N elements shifted the final time means …
Joy Buolamwini

• Founded Algorithmic Justice League
  • Rhodes Scholar, Anita Borg Scholar
  • TedX: Fighting Algorithmic Bias
  • Facial Recognition Bias
  • MIT MS with Ethan Zuckerman

And so in exploring this [facial recognition], I could have viewed my face not being consistently detected as, “Oh, this is a technical challenge” — but being in the space of the Center for Civic Media definitely orients me to [say], “This is not just a technical challenge ... this is as much a reflection of society as other spaces where you see inequities that need to be addressed.”

https://mitadmissions.org/blogs/entry/interview-joy-buolamwini/
From Point to Nbody ...

- Making a Point class to learn about objects
  - What's familiar can be helpful

- Concepts for Nbody
  - Constructing objects, reading from files
Java is Object-Oriented

• Every class is-a Object, Java parlance: extends
  • Inherits certain properties of Object.java
  • API: https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/lang/Object.html
  • 201: toString(), equals(), hashCode()

• New classes can override these methods
  • How do you print yourself? Compare yourself?
  • How can we remember this?
What’s the Problem?

https://coursework.cs.duke.edu/201spring20/classcode/blob/master/src/PointDriver.java

- Generate points and add to list if not there
  - Which method called is static method below?

```java
import java.util.*;

public class PointDriver {

    public static void main(String[] args) {
        int size = 10;
        int max = 2;
        Point[] array = PointGenerator.getRandomPointsInt(size, max);
        ArrayList<Point> list = new ArrayList<>();
        for (Point p : array) {
            if (!list.contains(p)) {
                list.add(p);
            }
        }
        System.out.printf("array = %d, list = %d\n", array.length, list.size());
        for (Point p : list) {
            System.out.println(p);
        }
    }
}
```
A Few ArrayList details

- Access to class via import statement
- Definition of ArrayList variable <...
- What happens when list.add(..) called?

```java
import java.util.*;

public class PointDriver {

    public static void main(String[] args) {
        int size = 10;
        int max = 2;
        Point[] array = PointGenerator.getRandomPointsInt(size, max);
        ArrayList<Point> list = new ArrayList<>();

        for(Point p : array) {
            if (!list.contains(p)) {
                list.add(p);
            }
        }
    }
}
```
Why does .contains fail?

• Points (x,y) with 0 <= x < 2 and 0 <= y < 2
  • How many are there? How many generated?
What’s the solution?

• How does `a.contains(x)` work where `a` is an `ArrayList<String>`, `ArrayList<Point>`
  
  • Code below is not `ArrayList` method, ...
    
    `http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/tip/src/share/classes/java/util/ArrayList.java`
  
  • Works for String, does NOT work for Point!

```java
public boolean contains(ArrayList<String> a, String elt) {
    for(String s : a) {
        if (s.equals(elt)) return true;
    }
    return false;
}
```
WOTO (2+ minutes, correctness)