CSCI 136
Data Structures & Advanced Programming

Lecture 5

Fall 2019

Bill & Sam

Administrative Details

- Read and prepare for Lab 2
  - Bring a design document!
  - We'll collect them
  - We'll also hand out one of our own for comparison

Last Time

- String Manipulation Example: XML parsing
- More on Java Program Organization

Today

- Vectors
- Code Samples
  - WordFreq (Vectors, Associations, histograms)
  - Dictionary (Associations, Vectors)
- If time : Lab 2 Intro

Vector: A Flexible Array

A Limitation of Arrays

- Must decide size when array is created
- What if we fill it and need more space?
  - Must create new, larger array
  - Must copy elements from old to new array

Enter the Vector class

- Provides functionality of array
  - Sadly, can't use [] syntax…
- Automatically grows as needed
- Can hold values of any class-based type
  - Not primitive types---but there's a work-around

Vectors

- Vectors are collections of Objects
- Methods include:
  - `add(Object o), remove(Object o)`
  - `contains(Object o)`
  - `indexOf(Object o)`
  - `get(int index), set(int index, Object o)`
  - `remove(int index)`
  - `add(int index, Object o)`
  - `size(), isEmpty()`
- Remove methods preserve order, close "gap"

Example: Word Counts

- Goal: Determine word frequencies in files
- Idea: Keep a Vector of (word, freq) pairs
  - When a word is read…
  - If it's not in the Vector, add it with freq = 1
  - If it is in the Vector, increment its frequency
- How do we store a (word, freq) pair?
  - An *Association*
- Let's take a look at WordFreq.java

WordFreq.java

- Uses a Vector
  - Each entry is an Association
  - Each Association is a (String, Integer) pair
- Notes:
  - Include structure.*;
  - Can create a Vector with an initial capacity
  - Must *cast* the Objects removed from Association and Vector to correct type before using

Implementing Vectors

- A Vector holds an array of Objects
- Key difference is that the number of elements can grow and shrink dynamically
- How are they implemented in Java?
  - What instance variables do we need?
  - What methods? (start simple)
- Let's explore the implementation….

Class Vector : Instance Variables

```
public class Vector {
private Object[] elementData;     // Underlying array
protected int elementCount;       // Number of elts in Vector
protected final static int defaultCapacity;
protected int capacityIncrement;  // How much to grow by
protected Object initialValue;    // A default elt value
}
```

- Why Object[]?
  - Don't know the actual type of data
- Why elementCount?
  - size won't usually equal capacity
- Why capacityIncrement?
  - We'll "grow" the array as needed

Core Vector Methods

```
public class Vector {
    public Vector()  // Make a small Vector

    // Make Vector of given capacity
    public Vector(int initCap)

    // Add elt to (high) end of Vector
    public void add(Object elt)

    // Add elt at position I
    public void add(int i, Object elt)

    // Remove (and return) elt
    public Object remove(Object elt)

    // Remove (and return) elt at pos I
     public Object remove(int i)  //
```

Core Vector Methods

```
   public int capacity()   // Return capacity
   public int size()       // Return current size
   public boolean isEmpty()// Is size == 0?

   // Is elt in Vector?
   public boolean contains(Object elt)

   // Return elt at position I
   public Object get(int i)

   // Change value at position I
   public Object set(int i, Object elt)

   // Return earliest position of elt
   public int indexOf(Object elt)
}
```

Class Vector : Basic Methods

- Much work done by few methods:
  - indexOf(Object elt, int i)
    - Find first occurrance of elt at/after pos. I
    - Used by indexOf(Object elt)
    - remove methods use indexOf(Object elt)
  - firstElement(), lastElement() use get(int i)
- Method names/functions in spirit of Java classes
  - indexOf has same behavior as for Strings
- Let's explore Vector.java….
- Methods are straightforward except when array is full
- How do we add to a full Vector?
  - We make a new, larger array and copy values to it

Extending the Array

- How should we extend the array?
- Possible extension methods:
  - Grow by fixed amount when capacity is reached
  - Double array when capacity is reached
- How could we compare the two techniques?
  - Run speed tests?
    - Hardware/system dependent
  - Count operations!
  - We'll do this soon

ensureCapacity

- How to implement `ensureCapacity(int minCapacity)`?

```
// post: the capacity of this vector is at least minCapacity
public void ensureCapacity(int minCapacity) {
   if (elementData.length < minCapacity) {
      int newLength = elementData.length; // initial guess
      if (capacityIncrement == 0) {
      // increment of 0 suggests doubling (default)
         if (newLength == 0) newLength = 1;
            while (newLength < minCapacity) {
              newLength *= 2;
            }
       } else {
      // increment != 0 suggests incremental increase
         while (newLength < minCapacity) {
            newLength += capacityIncrement;
         }
      }
```

```
  // assertion: newLength > elementData.length.
     Object newElementData[] = new Object[newLength];
     int i;

  // copy old data to array
     for (i = 0; i < elementCount; i++) {
        newElementData[i] = elementData[i];
     }

     elementData = newElementData;
        // garbage collector will pick up old elementData
  }
  // assertion: capacity is at least minCapacity
}
```

Notes About Vectors

- Primitive Types and Vectors

```
Vector v = new Vector();
v.add(5);
```

  - This (technically) shouldn't work! Can't use primitive data types with vectors…they aren't Objects!
  - Java is now smart about some data types, and converts them automatically for us -- called *autoboxing*
- We used to have to "box" and "unbox" primitive data types:

```
Integer num = new Integer(5);
v.add(num);
…
Integer result = (Integer)v.get(0);
int res = result.intValue();
```

- Similar wrapper classes (Double, Boolean, Character) exist for all primitives
  - Each has a valueOf() method to return primitive

Vector Summary & Notes

Vectors: "extensible arrays" that automatically manage adding elements, removing elements, etc.

1. Must cast Objects to correct type when removing from Vector
2. Use wrapper classes (with capital letters) for primitive data types (use "Integers" not "ints")
3. Define equals() method for Objects being stored for contains(), indexOf(), etc. to work correctly

A Vector-Based Dictionary (read on your own)

```
protected Vector defs;
public Dictionary() {
   defs = new Vector();
}

public void addWord(String word, String def) {
   defs.add(new Association(word, def));
}

// post: returns the definition of word, or "" if not found.
public String lookup(String word) {
   for (int i = 0; i < defs.size(); i++) {
       Association a = (Association)defs.get(i);
       if (a.getKey().equals(word)) {
           return (String)a.getValue();
       }
   }
   return "";
}
```

Dictionary.java

```
public static void main(String args[]) {
   Dictionary dict = new Dictionary();
   dict.addWord("perception", "Awareness of an object of
        thought");
   dict.addWord("person", "An individual capable of moral
        agency");
   dict.addWord("pessimism", "Belief that things generally
        happen for the worst");
   dict.addWord("philosophy", "Literally, love of
        wisdom.");
   dict.addWord("premise", "A statement whose truth is used to
        infer that of others");
}
```

Randomizing a Vector (discuss with a friend)

- How would we shuffle the elements of a Vector?
- `shuffle(Vector v)`
  - Many ways to implement.
  - An efficient way
    - Randomly move elements to "tail" of vector
    - Do this by swapping random element with last element
- swap is a little tricky
  - Three step process, not two!

Lab 2 Preview

- Three classes:
  - FrequencyList.java
  - Table.java
  - WordGen.java
- Two Vectors of Associations
- toString() in Table and FrequencyList for debugging
- What are the key stages of execution?
  - Test code thoroughly before moving on to next stage
- Use WordFreq as example

Lab 2: Core Tasks

- FreqencyList
  - A Vector of Associations of String and Integer
  - Add a letter
    - Is it a new letter or not?
    - Use indexOf from Vector class
- Pick a random letter based on frequencies
  - Let total = sum of frequencies in FL
  - generate random int r in range [0…total]
  - Find smallest k s.t. r <= sum of first k frequencies

Lab 2: Core Tasks

- Table
  - A Vector of Associations of String and FrequencyList
  - Add a letter to a k-gram
    - Is it a new k-gram or not?
  - Pick a random letter given a k-gram
    - Find the k-gram then ask its FrequencyList to pick
- WordGen
- Convert input into (very long) String
  - Use a StringBuffer---see handout