CSCI 136
Data Structures & Advanced Programming

Bill Jannen

Lecture 5

Feb 13, 2017

Announcements

- Lab 1 is **due** today
  - Wednesday Labs: 7pm
  - Thursday Lab: 10pm
- Lab 2 will be **posted** today
  - Read it and do the prep *before* your lab section

Last Time

- Continued reviewing Java
  - Arrays
  - Strings
  - Histograms
  - java.util.Random
  - public static void main(String args[])

Today's Outline

- Refresher of the "static" keyword
- Quick review of Strings
- Learn about pre/post conditions and assertions
- Discuss Associations and Vectors
- We need to go quickly… we will revisit topics on Wednesday

Static vs. Non-static

Static variables

- One copy shared by all instances
- Accessed using the class name

Instance variables

- Unique copies for each class instance
  - Describe the internal state of an object
- Can always use this.___ to refer to an instance variable

Static vs. Non-static

Static methods

- Do not depend on any internal state
  - Cannot use this.___ variables/methods
  - Necessary inputs are passed as parameters (or are static)
- Called using the class name

Non-static methods

- Depend on the internal state of an object
  - Use this.___ instance variables/methods
- Called on an object *instance*

Static vs. Non-static Example: Person.java

Shaquille O'Neal: 7' 1"
(aka Shaq Diesel, Superman, The Big Aristotle, …)

Simone Biles: 4' 9"

Quick Review: Strings in Java

- Useful methods (also check javadocs)
  - `indexOf(string);`
    `indexOf(string, startIndex);`
  - `substring(start, end); //[start,end)`
  - `charAt(int index);`
  - `equals(other);`
  - `toLowerCase();`
  - `toUpperCase();`
  - `compareTo(string);`
  - `length();`
  - `startsWith(string);`

Using Strings

- Suppose we want to parse an XML listing of our music library
  - XML = eXtended Markup Language
  - XML is used for many things
  - CD info:

```
<CD>
    <TITLE>Shaq Diesel</TITLE>
    <ARTIST>Shaquille O'Neal</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Jive Records</COMPANY>
    <YEAR>1993</YEAR>
</CD>
```

- How can we find and print just the titles?
  - See CDTitles.java
  - Redirecting System.in in Unix: java CDTitles < cds.xml

Moving on…

Pre and Post Conditions

- Recall `charAt(int index)` in Java String class
- What are the pre-conditions for charAt?
  - 0 <= index < length()
- What are the post-conditions?
  - Method returns char at position index in string
- We put pre and post conditions in comments above most methods

```
/* pre: 0 ≤ index < length
 * post: returns char at position index
 */
public char charAt(int index) { … }
```

Pre and Post Conditions

- Pre and post conditions "form a contract"
- Your method should guarantee that the post-condition is true if called when the pre-condition is true
- Examples:
  - `s.charAt(s.length() - 1)`: index < length, so valid
  - `s.charAt(s.length() + 1)`: index > length, not valid
- These conditions document requirements that the program **should** satisfy

Java Assertions

- Pre and post condition comments are useful as a programmer, but it would be really helpful to know as soon as a pre-condition is violated (and return an error)

- Java's `assert` keyword let's us enforce conditions in our running code.

```
assert <condition> : <error-msg>;
```

- The Assert class (in structure5 package) was necessary when the book was written, but assert is now a part of the language

Assert Example: Fill.java

General Rules about Assertions

1. State pre/post conditions in comments
2. Check conditions in code using "assert"
3. Fail in unexpected cases (such as the default block of a switch statement)
4. Run your code with the `-ea` flag (`-enableassertions`)

```
$ java –ea Program
```

- Any questions?
- You should use Assertions in Lab 2

Moving on…Dictionary Class

- Now we're going to discuss our first general data structure!

- What is a Dictionary?
  - Really just a *map* from word to definition…
  - We will call these mappings Associations
  - Task: given word, lookup and return definition

```
$ java Dictionary <word>
```

    - Prints definition

Other Associations

- Websters:
  - Word → Definition
- MtnOne:
  - Account number → Balance
- Peoplesoft:
  - Student name → Grades
- NSA:
  - SSN → ???
- In general:
  - Key → Value

Association Class

- We want to capture the "key → value" relationship in a general class that we can reuse everywhere
- What type do we use for key and value instance variables?
  - Object!
  - We can treat any thing as an Object since all classes inherently extend Object class in Java…

Association Class (see javadoc)

```
import structure5.*;
class Association {
   protected Object key;
   protected Object value;

   //pre: key != null
   public Association (Object K, Object V) {
        Assert.pre (K!=null, “Null key”);
        key = K;
        value = V;
   }

   public Object getKey() {return key;}
   public Object getValue() {return value;}
   public Object setValue(Object V) {
        Object old = value;
        value = V;
        return old;
   }
}
```

Dictionary Class

- Now that we have an Association class, let's implement Dictionary.java
- A Dictionary object is really just a collection of Associations
- What should we use to store our Associations?
  - An array!

Dictionary.java (version 1)

```
protected Association words[] = new Association[5];
public Dictionary() {
   words[0] = new Association("perception", "Awareness of an
              object of thought");
   words[1] = new Association("person", "An individual capable of
              moral agency");
   words[2] = new Association("pessimism", "Belief that things
              generally happen for the worst");
   words[3] = new Association("philosophy", "Literally,
              love of wisdom.");
   words[4] = new Association("premise", "A statement whose
              truth is used to infer that of others");
}

// post: returns the definition of word, or "" if not found.
public String lookup(String word) {
     for (int i = 0; i < words.length; i++) {
         Association a = words[i];
         if (a.getKey().equals(word)) {
             // note cast to recover type from Object
             return (String)a.getValue();
          }
     }
     return "";
}
```

Problems with Arrays

- Dictionary is a fixed size
  - How do we support addWord?
- Possible solutions:
  - Big array and keep a counter of current number of words
    - Error prone. What if we run out of space in array?
  - Big array-like data structure that can dynamically grow and manage itself

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()`

Dictionary.java (version 2)

```
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 (version 2)

```
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");
}
```

Recap

- Preconditions and postconditions define a contract for our methods
- Assertions can verify our assumptions + give useful feedback
  - Must be enabled (disabled by default for performance reasons)
- Dictionaries map keys to values
- The Association class contains a key-value pair
- Vectors are like arrays, but *they can grow!*

Next Class

- All about Vectors
- What are “Generics”?
- The principle of abstraction