Lecture 25

Strings, Strings, and more Strings

Access Levels
- private: visible only in class
- protected: visible in class and subclasses
- public: visible everywhere

Basic Rules
- Instance variables:
  - private if no plan for future extensions
  - protected if class might be extended
- Methods:
  - public if to be used by other classes
  - private if only used inside class (helper methods)
  - protected if used in subclass

extends vs implements
- Interface: specify set of methods
- Inheritance:
  - Reuse code
  - Can treat subclass as superclass
    - FallingObject o = new FallingSleet;
    - images label
  - Subtype polymorphism: methods written to operate on the supertype also
    - One “extends” allowed.
    - Multiple “implements” allowed.

Strings
- Many programs manipulate text
  - Email
  - Web browsers
  - Peoplesoft
- [Demo: CountWords]

Strings are not new...
- msg.setText("Try again");
- String chosenColor =
  colorMenu().getSelectedltem().toString();
- String s = ""; // initialization for array
  // toString()
Strings are special

Object types
- Ex) FilledRect, JButton
- Create with new
- Invoke methods on them

Primitive types
- Ex) int, double, boolean
- Use directly
- Apply operators to them (+, -, <, & &)

Strings are special

- Strings are objects
  - Many, many String methods

- But also can be used as primitive types
  - Can specify directly with String literals
    - msg.setText("You got it in!");
    - private static final String TITLE = "Concentration";
  - Can use concatenation operator
    - s = s + shapes[i] + "in";

Some useful String methods

- public int length()
  - Not the same as .length for an array
- public boolean startsWith(String s)
- public int indexOf(String s)
- public int indexOf(String s, int startIndex)

JavaDoc: note importance of method comments

private String courseName = “Intro To CS”;

- courseName.length() => 11
- courseName.startsWith("In")
- courseName.startsWith("in")
- courseName.indexOf("o")
- courseName.indexOf("zebra")
- courseName.indexOf("o", 5)
private String courseName = “Intro To CS”;

- courseName.length() => 11
- courseName.startsWith(“In”) => true
- courseName.startsWith(“in”) => false
- courseName.indexOf(“o”) => 4
- courseName.indexOf(“zebra”) => -1
- courseName.indexOf(“o”, 5) => 7

Count Words

- Let’s find all occurrences of a given word in a text
- And count them

private int countWords(String text, String word) {
    int count = 0;
    int pos = text.indexOf(word, 0);
    while (pos > -1) {
        count++;
        pos = text.indexOf(word, pos + word.length());
    }
    return count;
}
private int countWords(String text, String word) {
    int count = 0;
    int pos = text.indexOf(word, 0);
    while (pos > -1) {
        count++;
        pos = text.indexOf(word, pos + word.length());
    }
    return count;
}

countWords("abracadabra", "ra")
    pos = 2

countWords("abracadabra", "ra")
    pos = 9

private int countWords(String text, String word) {
    int count = 0;
    int pos = text.indexOf(word, 0);
    while (pos > -1) {
        count++;
        pos = text.indexOf(word, pos + word.length());
    }
    return count;
}

countWords("abracadabra", "ra")
    pos = -1

Case-Sensitivity

• public String toUpperCase()
• public String toLowerCase()

• Note that these return Strings
• No changes to the String to which they are applied
• Strings are immutable in Java
   – myString = myString.toUpperCase();
     • Makes myString upper case
   – myString.toUpperCase();
     • Does not

Substring

public String substring(int start, int end) inclusive not

courseName = “Intro To CS”;

public String substring(int start)
the substring beginning at start and going all the way to the end of the string
private String courseName = "Intro To CS";

• courseName.substring(0, 5) => "Intro"
• courseName.substring(6, 9) => "To "
• courseName.substring(1, 1) => "" (empty string)
• courseName.substring(0, 1) => "I"
• courseName.substring(4, 0) => string index out of bounds
• courseName.substring(4, 20) => same
Link Extraction

- Let’s look at the html file for the CS 134 page
- Note the text mark-up commands:
  - `<a href>`
  - `<table>`
- Now let’s write a method that will extract all the links from a file like this. Why?
  - Check that no links are dead
  - If we were a spammer, might want to look for email addresses
  - If we were Google crawling the web,
  - If we were an organization looking for addresses for outreach
- The text in a file is just one giant String

```java
private String findLinks(String fullPage) {
    // A lower case only version of the page for searching
    fullPage = fullPage.toLowerCase();
    // The A tags found so far
    String result = "";
    // Start of <a tag specification
    int tagStart = fullPage.indexOf("<a ", 0);
    while (tagStart > -1) {
        int tagEnd = fullPage.indexOf(">", tagStart);
        String tag = fullPage.substring(tagStart, tagEnd + 1);
        result = result + tag + "n";
        tagStart = fullPage.indexOf("<a ", tagEnd);
    }
    return result;
}
```