CMSC 131
Object-Oriented Programming I
Mutability, StringBuffer
Dept of Computer Science
University of Maryland College Park

This material is based on material provided by Ben Bederson, Bonnie Dorr, Fawzi Emad, David Mount, Jan Plane
Overview

- Mutability
- StringBuffer
- Password Example
Immutable

- Immutable object
  - State is initialized when it is constructed but it will never change
  - Instance variables (state) cannot be changed
- Mutable object
  - Instance variables (state) can be changed
- There is no language construct that guarantees immutability
  - You have to design the class to enforce it
- Keep classes immutable (if possible)
  - If aliasing occurs it will not matter (no problem with sharing object if it cannot change)
- With mutable classes we need to make “defensive copies”
  - Changes made in one are not seen in another
- It is good practice to document whether a class is immutable
- How to define a class as immutable
  - Make all the instance variables private
  - Do not provide set methods
  - Only the constructor will initialize the instance variables
Using the Cat and CatOwner example, which map represents the assignments we have in main?

Is pet = otherGuy.pet something you want?
Anytime you are passing an object to a method or returning an object from a method you should ask whether a copy should be made.

Strings are immutable:
- Great! → Not need to make copies when we pass them to methods.
- Bad! → Inefficient to modify one.
  - Example: loop that reads characters one at a time and makes a string out of it (creates a lot of objects!)
A mutable String class
- Good → very efficient to modify them
- Bad → we have to worry about aliasing

Main methods
- `append` → add characters to end
- `insert` → add characters in middle
- `delete` → remove characters

Note
- `append`, `insert` → return object of type `StringBuffer`
- This is alias to object that the methods belong to

API:
- [http://download.oracle.com/javase/6/docs/api/java/lang/StringBuffer.html](http://download.oracle.com/javase/6/docs/api/java/lang/StringBuffer.html)

Example: `StringBufferExample.java`
Password Example
- Let’s write a Password class, along with some JUnit tests. This will allow us to review several of the concepts discussed so far.
- Specifications
  - Instance variable → value (String)
  - Methods
    - Constructor (value as parameter)
    - Copy Constructor
    - Equals method
    - toString method
    - stringChecker → verifies no characters are duplicated
- We will also try refactoring our code
  - Changing variable names
- While defining a JUnit test case we will need to catch an exception