Full Class

CSE 114 INTRODUCTION TO OBJECT-ORIENTED PROGRAMMING

Announcements

Topics:

- Static and non-static members in a class (“full class”)
- Visibility control of objects
- Memory representation of an object with static and non-static members

Reading: follow the lecture notes closely and use textbook as a reference

[Notes + Chapters 9, 10 ,11, 12]

Expanding the program structure again

Now mix static and dynamic members (fields and methods) in a class

See the needs for both in a class by understanding how they are used

See program_structure_6.txt

See Account.java that now contains a mix of static and dynamic members in a class

Also see UseAccount.java

Visibility control on state info in objects

public vs. private

With private, you would have to provide getters (readers) and setters (writers) unless you want to hide the private member data from outside

With public, you can access the fields (static and non-static) directly without using getters and setters

Why use private? Why hide state info?

- It makes software more maintainable!

See AccountPublic.java and UseAccountPublic.java

Memory representation of a class with static and dynamic members

See Account.java and UseAccount.java

Using static fields vs. using an additional class

See static_or_another_class.txt

Variables in Java

1. Local variables
   - within a method (function)
2. Instance variables (aka dynamic fields; non-static or dynamic variables)
   - within a class without the static keyword
   - a copy in each instance of the class (if you create 234 instances, there will be 234 copies)
3. Static variables (aka class variables)
   - within a class with the static keyword
   - only one copy in the entire class
   - shared by all the instances of the class

Lifetime of these variables

1. Local variables
   - alive only while the method is running/executing
2. Instance variables (aka dynamic fields; non-static or dynamic variables)
   - alive as long as an instance (object) is alive
   - when does an instance die, i.e., goes away from memory?
3. Static variables (aka class variables)
   - alive as long as the program is alive, i.e., until the main exits

Static objects vs. dynamic objects

Static object, e.g., the Account object

What do you mean?

- Well, it is a ‘meta-object’.

Dynamic objects, e.g., objects created as instances of a class (e.g., Account) using new