Overview & Java

CS 62 - Spring 2016
Michael Bannister

Homework

- Solutions to odd problems are in back of text.
- Ask questions at the beginning of class.
- Invitation to Piazza

Use Packages!!

- When writing programs, put all classes/interfaces in packages:

  package assignment1;
  ...

Card Deck Examples

- CardInterface -- interface
- AbsCard
  - abstract class, implements CardInterface
- Card extends AbsCard
- OtherCard extends AbsCard

$\left.\begin{matrix}\\ \\ \end{matrix}\right\}$ *alternate implementations*

- Deck
  - Class using cards

Java Keywords

- Abstract class -- can't be instantiated
  - usually some methods missing
- Information hiding qualifiers:
  - public
  - private
  - protected
- Static -- copy associated with class, not objects
- Final -- only assigned to once
  - in its declaration or constructor

Interfaces &Inheritance

- Class implements interface if supports all methods defined in interface
  - Try to use interfaces as types for flexibility
- Interface can extend another by adding methods
  - If A extends B and x has type A, then also has type B
- One class can extend another
  - inherits fields and methods
  - can override existing methods, add new ones
- instanceof & casts

Extending vs Implementing

- Extending a class allows sharing behavior:
  - Card, OtherCard extend AbsCard
- Implementing an interface provides an implementation
  - Card, OtherCard implement CardInterface
  - Either can be associated with variable of type CardInterface.
  - Makes it easier to replace implementations.

Generics

- Can write classes parameterized by types
- See Association class
- Can only instantiate type parameters by interfaces or classes, not primitive types
- "Wrapper" versions of primitive types can be used instead of primitive types:
  - int -> Integer, double -> Double, boolean -> Boolean

Association is part of Bailey structure5 library.
See documentation & code on web site!

JavaDoc

- Stylized form of comments, w/tools to extract
- Common tags:
  - for class
    - @author *author name*
    - @version *date*
  - for method
    - @param *param name and description*
    - @return *value returned, if any*
    - @throws *description of any exceptions thrown*

Comments

- Class header needs @author, @version
- Method header should include
  - Description of what (not how) it does
  - @param line for each parameter
  - @return if method returns a value
  - pre and post conditions as necessary
    - If no @return, then must have post
    - If checkable then add assert (see later) for postconditions

Pre and Post-conditions

- Pre-condition: Specification of what must be true for method to work properly
- Post-condition: Specification of what must be true at end of method if precondition held before execution.
- See Ratio class example

Assertions in Java

- Won't use Assert class from Bailey.
- Command to check assertions in standard Java
  - Two forms
    - assert boolExp
    - assert boolExp: message
- Article on when to use assert:
  - http://docs.oracle.com/javase/8/docs/technotes/guides/language/assert.html
  - Short summary -- never use for preconditions of public methods -- make explicit checks
  - Use for postconditions & class invariants

Assertions help …

- Defensive programming
  - Little cost to executing assertions … and can turn off checking
  - Extremely useful in debugging in tracking down what is going wrong - can be better than inserting println's.
  - Also useful in checking cases that should not occur
    - e.g., defaults in switch, other control paths not taken.

Turning on assert

- Turn on assertions when run program, by adding "-ea" (without quotes) as virtual machine argument in arguments tab in Eclipse when set up runtime configuration.
- If leave it off, then ignores assert statements.
- If on and the assertion is false, then will raise an AssertionError exception and will print associated message
  - They should not be caught as represents a program error