




Lecture 2: Comments,
Conditions and Assertions



+ Today

- Reading
 - JS Chapter 2
- Objectives:
 - Finish review of Java
 - Comments
 - Pre- and post- conditions
 - Assertions
- Announcements
 - Book problems (solutions to odd-numbered in back of book)
 - comprehension_ques folder on Piazza
 - No office hours on Monday

+ Interfaces & Inheritance

- Class implements interface if supports all methods defined in interface
- 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
 - Ex: in Ratio class later

+ Extending vs Implementing

- Extending a class allows **sharing** behavior:
 - Card, OtherCard extend AbsCard
- Implementing an interface allows **replacing** implementation
 - Card, OtherCard implement CardInterface
 - Either can be associated with variable of type CardInterface.

+ Card Deck Examples

- CardInterface -- interface
 - AbsCard
 - abstract class, implements CardInterface
 - Card extends AbsCard
 - OtherCard extends AbsCard
 - Deck
 - Class using cards
- } Alternate implementations

End of Java Review!

+ Use Packages!

- What's a package?
- When writing programs, put all classes and interfaces in packages
- Familiar packages:
 - java.util
 - java.io
 - java.lang
 - javax.swing
 - java.awt

```
package assignment1;  
...
```

+ Generics

- Can create classes parameterized by types
 - Vector<E>, List<E>, Association<K,V>
- See Association class
 - part of Bailey structure5 library
 - See documentation and code on Bailey website
- Can only instantiate type parameters by interfaces or classes, not primitive types
- Wrapper versions of primitive types can be used instead of primitive types:
 - Integer (int), Double (double), Boolean (boolean)

+ JavaDoc

- Stylized form of comments, w/tools to extract

```
/**  
 * comments here  
 */
```

- Common tags:
 - @author *author name*
 - @version *date*
 - @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

- (Optional) practices that result in better code!
- 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

- We won't use the Assert class from Bailey.
- Command to check assertions in standard Java
 - `assert boolExp`
 - `assert boolExp: message`
- Article on when to use assert:
 - <http://download.oracle.com/javase/7/docs/technotes/guides/language/assert.html>
 - Short summary -- never use for preconditions of public methods -- make explicit checks
 - Use for postconditions & class invariants