CS 501 Quiz 2 Review

1. Method overloading
2. Javadoc
3. Exceptions
4. Values vs references
5. Arrays
6. Using Generic classes
7. ArrayList
8. IO Streams
9. Checked vs Unchecked exceptions
10. Class Hierarchies
    a. Creating a subclass of a superclass
    b. The keyword "extends"
    c. The keyword "super"
        i. Use in subclass constructor
        ii. Use in subclass methods
    d. "protected" vs "private" data fields
    e. Is-a versus Has-a relationships for code re-use
    f. Assigning superclass variable to subclass instance is valid
    g. Assigning subclass variable to superclass instance requires explicit cast operator and may throw ClassCastException
11. Polymorphism
    a. Method overriding versus overloading
        i. To override a method, the subclass method signature (method name and parameter types) must match the superclass method signature exactly
        ii. Use "@Override" annotation to guarantee an override is valid
    b. Invoking a method on a superclass variable assigned to a subclass instance
        i. The method must be declared (not necessarily implemented) in the superclass, else compile-time error
        ii. If the method is overridden in the subclass, then the subclass implementation will run, else the superclass implementation will run
    c. Interfaces
        i. How to define
        ii. Can only have abstract methods (no body)
        iii. All methods are automatically public, abstract
        iv. All fields are automatically public, static, and final
        v. Cannot be instantiated
        vi. Define subclass by using the keyword "implements" unless the subclass is itself an interface, in which case you use the keyword "extends"
        vii. Allows multiple inheritance
        viii. Does not allow code to be re-used
    d. Abstract classes

        i. How to define
        ii. Cannot be instantiated
        iii. Can have abstract methods (no body)
        iv. Can implement some methods
        v. Define subclass by using the keyword “extends”
        vi. Does not allow multiple inheritance
        vii. Allows code to be re-used

e. Concrete class (also called Actual class)
    i. Implements all its methods
    ii. Can be instantiated
    iii. Define subclass by using the keyword “extends”
    iv. Does not allow multiple inheritance
    v. Allows code to be re-used
f. The Object class
    i. Every class has Object as a superclass
    ii. Most classes should override the “toString” and “equals” methods
    iii. The “getClass” method is useful for determining if two object instances are the same subclass
g. Code to an interface
    i. Motivation based on encapsulation and information hiding
    ii. Ideally, public methods (which specify your API) should be defined in terms of interfaces rather than concrete classes
    iii. This hides the details of your implementation from the user, which gives you the flexibility to change your implementation in the future
    iv. However, an interface should be viewed as a contract between you and your user that changes as little as possible over time