

Quiz 2 Review Topics:

1. Object Semantics (L2)
 - a. Variables, References, and Objects
 - b. Aliasing, Mutability, and Reference Equality
 - c. Null References
 - d. Object types vs. primitive types
 - e. Lexical scoping, invocation stack, call by sharing
 - f. Object diagrams
2. Subclassing and Dynamic Dispatch (L3)
 - a. Overriding and dynamic call dispatching
 - b. Overriding vs. overloading
 - c. Polymorphism
 - d. Template methods
 - e. Downcast vs. typecast
3. Object Models (L4)
 - a. Models vs. diagrams
 - b. Multiplicity (*, +, !, ?) at source and target
 - c. Mutability at source and target
 - d. Subclassing (filled vs. empty arrowhead)
4. Procedure Specification (L5)
 - a. Contracts, firewalls and decoupling
 - b. Behavioral equivalence
 - c. Specification structure: precondition (*requires*), post-condition (*effects*), frame condition (*modifies*)
 - d. Non-deterministic vs. under-determined
 - e. Operational vs. declarative specification
 - f. Abstract or specification fields
 - g. When to check pre-conditions
 - h. Specification ordering
5. Abstract Types (L6)
 - a. Classification of types: mutable or immutable
 - b. Classification of operations: creators, producers, mutators, observers
 - c. Representation independence vs. exposure
6. Representation Invariants (L7)
 - a. Abstract values vs. rep values
 - b. Rep to abstract mapping (surjective)
7. Abstraction Functions (L8)
 - a. Benevolent side-effects
 - b. AF idioms: comprehension, recursion, projection, and example
8. Dependences and Decoupling (L9)
 - a. Parnas's "uses" relation
 - b. Liskov's MDD notation
 - c. 3-Element model: grouping, assumption annotations, name dependences, families and patterns

9. Exceptions and Assertions (L10)
 - a. Defensive programming
 - b. Uses for assertions: what to assert and what not to assert
 - c. Assertions in Java: assert vs. assertAlways
 - d. Exceptions vs. special results
 - e. Checked vs. unchecked exceptions

10. Testing and Debugging (L11)
 - a. Validation: formal reasoning (verification), informal reasoning, testing
 - b. Regression testing
 - c. Testing strategy: unit testing, integration testing
 - d. Black box vs. glass box testing

11. Equality and Copying (L12)
 - a. Equals contract: reflexive, symmetric, transitive, nothing is equal to null, repeated calls should be the same, objects that are equal should have the same hash code
 - b. Behavioral vs. observational equivalence
 - c. Copying: clone vs. copy-constructor

12. Design Patterns (L13-L15)
 - a. Definitions: standard solution to a common problem, a shorthand for communicating design concepts, a particular shape of object diagram, OM, MDD
 - b. Benefits of using design patterns: more safety (e.g. typesafe enumeration pattern), better performance (e.g. flyweight), more flexibility (e.g. composite). But, overuse may decrease understandability of design and implementation.
 - c. Design patterns we have covered (know what problem they solve, when you should use them, what are their advantages/disadvantages):
 - i. Creational Patterns: Factory Method, Factory Object, Prototype
 - ii. Sharing Patterns: Singleton, Interning, Flyweight
 - iii. Structural Patterns: Composite
 - iv. Traversal Patterns: Iterator (linear traversal), Interpreter and Visitor (composite traversal)
 - v. Patterns for Events: Observer, Blackboard, Mediator
 - vi. Wrappers: Adapter, Decorator, Proxy

13. Subtyping (L16)
 - a. Subclassing: enables implementation and interface re-use. Subclassing is not the same as subtyping
 - b. Subtyping: if A is a subtype of B, then the specification of A implies the specification of B
 - c. Substitution Principle: we can substitute A for B if
 - i. $pre_B \Rightarrow pre_A$ (A requires less than B)
 - ii. $post_A \Rightarrow post_B$ (A does at least as much as B)
 - iii. $R_A \subseteq R_B$ (Return type of A is a subtype of return type of B)
 - iv. $P_B \subseteq P_A$ (Each parameter of B is a subtype of the parameters of A)
 - v. $E_A \subseteq E_B$ (A does not throw more exceptions than B)
 - d. Substitution principle alone is not enough to determine whether A is a subtype of B. It must also be true that A preserves all general properties of B (such as immutability).

14. Usability (L17 – L18)

- a. UIs are hard to design: you are not the user, user is (almost) always right, online help doesn't help, UI testing is expensive
- b. Usability dimensions: learnability, efficiency, memorability, errors, satisfaction
- c. The human machine: perception, motor, memory, color
- d. Usability engineering techniques
 - i. iterative design: design, implement, evaluate, repeat
 - 1. spiral model: use cheap throw-away prototypes and cheap evaluation for the first few iterations
 - ii. low-fidelity prototypes: paper, web-pages, etc.
 - iii. heuristics: 10 Nielsen's heuristics
 - iv. user testing: formative evaluation

15. GUI Design (L19)

- a. Swing widgets
- b. Layout managers
- c. Anonymous inner classes as Listeners
- d. Background tasks in Swing

16. Case Study: Java Collections API (L20)

- a. Type hierarchy (sets, lists, and maps)
- b. Hierarchy of interfaces and a separate hierarchy of implementation classes
- c. Optional methods for enabling unmodifiable views
- d. Polymorphism
- e. Skeletal implementations
- f. Capacity, allocation and GC (you are not completely free of memory management concerns)
- g. Copies, conversions, wrappers
- h. Views can provide: functionality extension, decoupling, coordinate transformation
- i. Views can be dangerous (the sublist problem)

17. Conceptual Modeling (L21)

- a. Graphical notation similar to code object models
- b. Atoms, sets, and relations
- c. Ternary relations