1. Terminology

- mutable vs. immutable

Mutable: change states , immutable: can't change states

- static vs. dynamic

Static: One insance, Same for all objects dynamic: called usig object as reciever

- concrete vs. abstract

concrete: can instantiate abstract: can't instantiate

- overriding vs. overloading

overriding: Changing a previous method in a super class overloading: same method having multiple instances w/ different parameters

2. Abstract data type (ADT)

can't be initalized, provides framework.

extending classes inherit with non-abstract modifiers methods, must re-write with abstract modifiers methods in subclass

Name

Operators.

Behavior of Operators :

3. Recipe for implementing an immutable ADT that is specified by an algebraic specification

Establish difference between creators and operations.

Each creator should have a subclass that extends ADT.

For each ~~[illegible]~~ static method define ~~as~~ a dynamic method in ADT.

Write operations based on algebraic specs.

4 Abstraction

- Abstraction barrier
    - Increase privacy between client implementation
    - Simplify complex concepts
- Client perspective versus implementer perspective
    - The client is the programmer writing code that calls the function (or the code itself).
    - The implementer is the programmer who write the implementation.
- Information hiding
    - Adding modifiers to limit client access
    - Allows you to change part of the code without worrying about who has used it before
- Encapsulation
    - A language mechanism for restricting access to some of the object's components.
    - A language construct that facilitates the bundling of data with the methods (or other functions) operating on that data.
    - Is the technique of making the fields in the class private and providing access to the fields via public methods.

5. Design patterns

- Factory Method
    - When you want to create custom objects that take unique parameters
    - Example: you create objects via static methods vs the new constructor
- Singleton
    - Restricts class to only allows one instance

Good when there are no fields, example: empty

6. Testing

~~Statean~~

- Purpose

Prevent + find bugs as well as making sure we meet our specs.

- White-box vs. Black-box

Blackbox: only have specs/requirements

White-box: have spess/requirements and design implementation

- Types of testing

Equivaleace Partitoning: identify classes of errors, not specifics, for efficiency
Boundary value analysis: checking each side of boundery cases close to boundaries
Specs/requirements: making sure the program meets the specs/requirements
Statement Coverage
Brench coverage
Path Coverage

7. Debugging

To find and correct errors. Write test cases first.

Do not make random changes.

Use breakpoints or print statements to help.

8. People

- Fred Brooks

Brooks Law: Adding manpower to a late project makes it later
Rules of thumb for scheduling: 1/3 Design, 1/6 Coding, 1/2 Testing
- Incremental Testing: Test each method upon completion
Turing Award winner - given for major contributions of lasting importance to computing

9. Evaluating expressions

FSet Example: FSet.contains(f1, bob)

```
→ FSet.contains(FSet.insert(f0, alice), bob)
→ FSet.contains(f0, bob)
→ FSet.contains(FSet.emptySet(), bob)
→ false
```

Expanding expressions and reducing them to basic creators.

10. Dynamic method dispatch

At runtime, Java knows which implementation of a method to use by the instance type of what it is being called on

11. Polymorphism

- ad hoc – allows methods to perform differently when given different types of input (ie overloading)
- inclusion
- parametric – using generic (parametric?) data types that can later be instantiated to a specific type.

12. Access modifiers

- private within class
- public anywhere
- protected class, package and subclass
- (default) within package & class

13. Refactoring

– make changes to the code but not the functionality

Examples of refactoring: renaming, redundancy, etc.

Types:

- privatization of members
- nested classes
  good for: logical grouping, readable & easy to maintain code, encapsulation
- merge subclass with base class
- Singleton pattern
- using null ~~[illegible]~~ - inlining

14. Efficiency

- Big-O if $t(n)$ is bounded above by some constant multiple of $g(n)$ then $t(n) \in O(g(n))$
- Big Omega if $t(n)$ is bounded below by some positive constant multiple of $g(n)$ the $t(n) \in \Omega(g(n))$
- Big Theta if $t(n)$ is bounded above and below by some positive constant multiples of $g(n)$ then $t(n) \in \Theta(g(n))$

15. Optimization

- subexpression elemination
- precomputation
- caching

Rules:

1. Don't
2. Don't Yet
3. Don't optimize more than necessary

- Techniques for improving the performance of programs

- Don't compute things that don't need to be computed
- Don't recompute things if you can help it.
- Use more efficient representations and algorithms

16. Divide and Conquer Algorithms

•Binary Search
•-O(logn)
•-sorted data structor using total ordering
•Merge Sort
•- O(nlogn) ®C average
•- O(nlogn) ®C worst case
- split array in half
use merge sort on both halves
recombine arrays in sorted order
•Quick Sort
•-
•- choose pivot point
•arrange other elements around pivot, greater on the right, less on the left
•choose new pivot point
17. Interfaces
Declares functions that must be implemented

•Iterator
•- next()
•- hasNext()
•- remove()
•Comparator
•method for comparing two types

18. Total Order

- antisymmetry
- transitivity
- law of tracheotomy

19. Phases of software engineering process

- Requirements
- Design
- Implementation
- Maintenace
- Testing

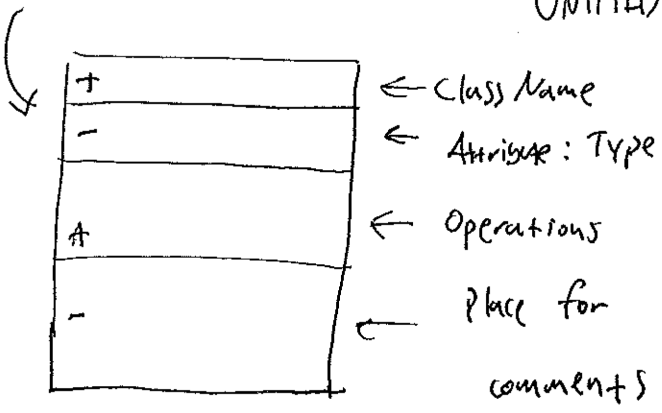
20. Trees and Binary Search Trees

AN ABSTRACT DATA TYPE (ADT) CONTAINING NODES & EMPTY'S WHERE EACH NODE CONTAINS SONS. (CHILDREN) A BINARY SEARCH TREE IS A SORTED TREE WHERE ALL ELEMENTS TO THE LEFT ARE LESS THAN ELEMENTS TO THE RIGTS.

21.<Generics>

- GENERICS ENABLE TYPES (classes & interfaces) to BE PARAMETERS WHEN DEFINING CLASSES INTERFACES AND METHODS.
- Typically used for Collections or Storage mechanisms

22. UML UNIFIED MODELING LANGUAGE



+ : indicates visibility

- : indicates invisibility