Group 1

1. Reuse

- Modularity
- Abstraction
- Parameterization
- Maintainability
- "Reduce, Re-use, Recycle"

2. Linux/UNIX Review—command line

ls - list directory
cat - output contents of file
cd - change current directory
mkdir - make a directory
javac, java - compile and run Java

3. Test First/Test-Driven

Testcases help to clarify requirements
Blackbox vs. Whitebox Testing
Write tests first
Code should be test-driven

4. Information hiding/access modifiers Class 3

public - accessible everywhere
private - accesible only within the same class
protected - accessible within the same package and also accessible within subclasses
default - accessible only within the same package

5. Immutable vs. Mutable

Immutable - the structure and information of the data should not change, rather new instances should be created

mutable - the structure and information of the data can be changed. One instance is passed around

6. Abstract data type (ADT)

- set of data
- set of operations
- description of what operations do

7. Abstract class vs. Concrete class. Abstract method vs Concrete method

Abstract classes cannot be instantiated; concrete classes can
↳ Classes can extend an abstract class to inherit its fields/classes.

Abstract methods have no body. All abstract methods must be implemented in all subclasses.

8. Java Syntax, including Liskov - Chapter 2: Review of Objects in Java
→ Similar to C and C++ ★ Type - Safety

Class: <visibility> <ClassName> extends <Class> implements <Interface> { }

Method: <visibility> <Type> <return type> (args) { body }

9. Write-Compile-Execute

Java compiles all of the code before executing.

1. Prep. of program text
2. Compilation of the programm
3. Execution of compiled program

10. Static Methods vs. Dynamic Method

Static Methods: Called by the class name
- Can't reference instance fields
- Use for factory methods for ADT's

Dynamic Methods: Called by the instance name
- Can reference instance fields

11. JUnit testing

Java's standard testing library
Can seperate tests by each method
-AssertTrue - checks that the given boolean is true
-AssertFalse - checks that the given boolean is false
-AssertEquals - Checks that two objects are equal
- Designate test methods by naming them testMethodName()
@Test designates the method as a test method

12. Designing test harness for given specifications

- Design AssertTrue, assertFalse, assertEquals
- Test only cases given from the specifications
- Print out results of total tests run, failed, passed, etc.

13. Abstraction barrier Slides - Class 04 - Pg. 29

Clients	Implementor
- Doesn't know how the datatype was implemented, but can use the datatype based on the specs.	- knows how the datatype was implemented

Both know the behavior of the datatype

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

- Define abstract class
- Define concrete subclasses for basic creators
- For each operation, define a static method within the abstract class
  - For each basic creator, return a new instance
  - For each non-basic creator, return an abstract method for the operation
- For each subclass, define all abstract methods in the abstract class

15. Software process

Phases

- Requirements
- Design
- Implementation
- Testing
- Maintenance

Brook's rule of thumb

- 1/3 design
- 1/6 coding
- 1/2 testing
  - 1/4 component testing
  - 1/4 system testing

Found in Slides - Class 04 pg. 4, pg. 9

Testing is a way of validating a program's correctness.

Debugging is the process of finding and removing bugs

16. Testing, including Liskov - Chapter 10: Testing and Debugging

Validation

- Ensuring the program works as intended
- Test paths through specification
- Test boundary conditions
- Aliasing Errors

Verification

- Program works on all possible inputs

Testing

- The process of running a program on a set of test cases and comparing the actual results with expected results

17. Black-box testing vs. White-box testing

Black-box

- Testing based on the specs alone
- Test Boundary conditions

White-box

- Implementation is known
- Test every path in the code

18. Dynamic Dispatch

- A mechanism by which a call to an overridden method is resolved at runtime.
- The type of object that's ~~ref~~ referred determines the method that is called

19. Interchangeable Parts

Guns / Modula programming

20. Liskov - Chapter 1: Introduction ABSTRACTION

- Abstraction by Parameterization
  - abstracts from the identity of the data by replacing them with parameters. It generalizes modules so that they can be used in more situations.
- Abstraction by Specification
  - abstracts from the implementation details to the behavior users can depend on
- Procedural Abstraction - allows us to introduce new operations
- Data Abstraction - allows us to introduce new types of data objects
- Iteration Abstraction - allows us to iterate over items in a collection without revealing details of how the items are obtained
- Type hierarchy - allows us to abstract from individual data types to families of related types

21. Debugging, including Liskov - Chapter 10: Testing and Debugging

- Debugging is the process of understanding and correcting errors

How to:

1. Begin by studying already available data.
2. Form a hypothesis that is consistent with those data.
3. Design and run a repeatable experiment that has the potential to refute the hypothesis.

22. The new rule

~~make~~

Calling new on a constructor returns a new instance of the object created by the constructor.

23. Factory method pattern

Isolates clients from the representations of a data type.

Use static methods rather than constructors to make new instances

new StackInt() ⟶ StackInt.empty()

They ~~are~~ don't have to make new instances every time if a pre-constructed instance exists

24. *Effective Java* items

~~from~~ slides for class 5 slide 37, listed on

important

- Obey general contract when overriding equals()
  - leave default if
    - don't care
    - instances inherently unique
    - overridden in superclass
    - never called
  - override should be
    - reflexive
    - symetric
    - transitive
    - consistant
    - false if passed null
- Always override hashcode() if override equals()
  - hashcodes must be same if objects equal
  - must return same value for same parameters

LOOK AT SLIDES FROM CLASS 06

25. Liskov - Chapter 3: Procedural Abstraction

CLASS DATE: 1/24/14

"An abstraction that hides details associated with executing an operation or task."

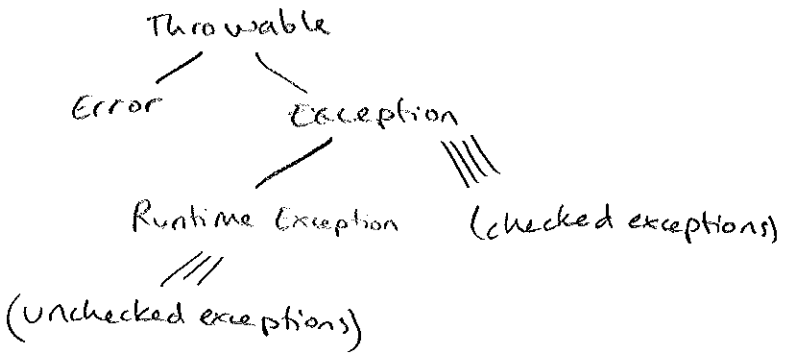
TEMPLATE FOR PROCEDURAL ABSTRACTION:

```
return_type pname(...)
  //REQUIRES: This clause states any constraints on use
  //MODIFIES: This clause identifies all modified inputs
  //EFFECTS: This clause defines the behavior.
```

26. Liskov - Chapter 4: Exceptions

TYPES OF EXCEPTIONS:

See: class 06 ppt, slide 19.



Items: 57
58
59
60
61
62
63
64
65

27. Data Structures—List, Set, Map, Queue, Stack

See: class 06 ppt, slide 21

DATA STRUCTURE	DESCRIPTION/ STRENGTHS	WEAKNESSES	EXAMPLE USAGES
List	a sequence of elements arranged in order of insertion	slow to search, slow to add/ remove arbitrary elements	List of accounts; prime numbers; the lines of a file
Set	a set of unique elements that can be searched quickly	does not have indexes; user cannot retrieve arbitrary elements	unique words in a book; lottery ticket numbers
Map	a group of associations between pairs of "key" and "value" objects	not a general-purpose collection; cannot easily map backward from a value to its key.	Word counting; phone book creation

Stack: First in last out

Queue: First in first out

Group 10

28. Generics

Used to force a class to only take the object type it is declared with

ArrayList<Integer> means that every object in the arraylist must be an Integer

29. Liskov - Chapter 5: Data Abstraction

Creation of data types that implement abstraction by specification - the methods are part of the type - and by parameterization. Allows the object to be used by user who only knows the behavior of an object rather than the specifics of its implementation (abstraction barrier).

30. Iterators, including Liskov - Chapter 6: Iteration Abstraction

Iterator is a generalization of iteration mechanisms. In Java, iterators are procedures that return a generator which produces elements used in iteration. Generators extend Iterator. They must implement:

boolean hasNext()

object next()

void remove()

31. Abstraction Function (Class 08 slides → 27) P99 ← Liskov

Connection between choice of implementation and abstract data type
→ instance variables used and relation to abstract object represented

AF: C → A
AF: abstraction function
C: Your representation
A: Abstract type

32. Rep Invariant (Class 08 slides → 39) (Liskov p. 107) P102.

Representation Invariant
Something that is true for all instances of an object
It's a predicate (returns a Boolean) that returns true for all legit objects
Method repOK checks all rep invariants for a class (something you can write)
(Class invariant mostly used interchangeably with Rep Invariant)

33. Binary Search (Class 09 slides → 8)

Requirements to do binary search:
- linear sequence
- sorted with total order

Can find elements in log time

Start in the middle and only look at the half that could contain the element

Example of binary search code on slide 20

34. Total Order

Requirements
① transitive
② anti symmetric
③ law of Tricholomy

35. Binary Search Tree (BST) "a non-linear structure in which elements are organized into a hierarchy"
- No duplicates
- Every left tree < parent, right tree > parent
- Left & Right must be BSTs

generic
↓
36. Comparator <T>

Java interface that defines two methods
- Compare (T o1, T o2)
  - returns an integer < 0 if o1 is < o2, 0 if they are equal, and > 0 otherwise
- Equals (Object obj)
  - returns a boolean stating whether obj is equal to this

37. Asymptotic notation Class 10 Slides (Slide 16)

- Order of growth of a function.
  - Big-O (upper bound) (Slide 19)
  - Big-Omega (lower bound) (Slide 50)
  - Big-Theta (bounded both above & below) (Slide 59)

38. Efficiency Class 10 Slides (Slide 71)

- Best case
- Worst case
- Average case

39. Optimization Class 10 Slides (Slide 74)

WII?
- Don't
- Don't yet
- Don't optimize more than necessary