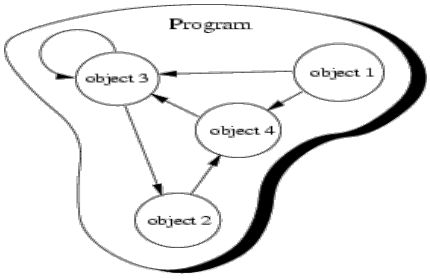
Week 5

Assignment 4 Review

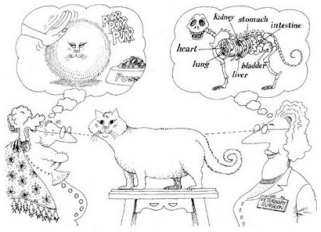
Object-Oriented Concepts



Object-Oriented Programming

- Abstraction
- Three Pillars of Object-Oriented Programming
  - Encapsulation
  - Inheritance
  - Polymorphism

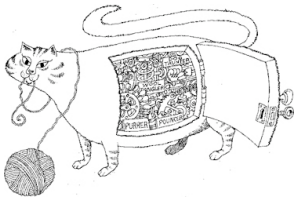
Abstraction



From http://samdelacruz.blogspot.com/2007_06_01_archive.html

- Wikipedia - http://en.wikipedia.org/wiki/Abstraction
- Examples

Encapsulation



From http://samdelacruz.blogspot.com/2007_06_01_archive.html

Wikipedi - http://en.wikipedia.org/wiki/Information_hiding

- Examples

Inheritance

- A relationship between two or more classes where some classes (the subclasses) are partially defined in terms of, i.e., inherit the features of, other classes (the superclasses).
  - Wikipedi - http://en.wikipedia.org/wiki/Inheritance_%28computer_science%29
  - Allows "differential programming", program only what differs from existing classes

Inheritance

Advantages & Disadvantages

+ Allows very abstract descriptions
+ Allows a significant amount of code reuse (not the only form of reuse)
– Places severe constraints on superclass flexibility
– Sensitive to order of class discovery
– Introduces high coupling

Types of Inheritance

- Single inheritance
  – Subclasses inherit from exactly one superclass
- Multiple inheritance
  – Subclasses may have more than one superclass
- Interface inheritance
  – Refers to inheriting a set of "signatures" but no associated behavior
  – Behavior must be implemented in the subclass.

Types of Inheritance
Advantages & Disadvantages

+ Multiple inheritance is a more general, flexible model which allows wider reuse
+ Interface inheritance eliminates the ambiguity associated with multiple inheritance
– Multiple inheritance suffers from ambiguity in interpretation of features inherited from multiple superclasses

Polymorphism

- Greek for "many shapes", it means allowing several classes to each offer an operation with the same signature, and further allowing those classes to respond differently to requests for that operation.
- Wikipedia - http://en.wikipedia.org/wiki/Polymorphism_%28computer_science%29

Types of Polymorphism

- Overriding
  - Allows a subclass to re-implement a method of its superclass, at run-time the subclass' method may be invoked when the object is being used as an instance of the superclass.
- Overloading
  - Allows a class to have any number of operations with the same name but varying signatures.
  - Not true polymorphism, methods are resolved at run time.

Class Design/Code Quality

Software Lifecycle

- Maintenance accounts for as much as 80% of total lifecycle costs
- Difficulty in maintenance or excessive maintenance costs will lead to replacement

Code Quality

- Two important code quality concepts:
  - Coupling
  - Cohesion

Coupling

- Coupling refers to links between separate units of a program
- Tightly coupled
  - Class depend closely on details of another
- Loose coupling allows
  - Understand one class independent of others
  - Modifications to one class do not impact others
  - change one class without affecting others.

Cohesion

- Cohesion refers to the the number and diversity of tasks that a single unit is responsible for
- Applies to classes and methods
- Highly cohesive
  - Each unit is responsible for one single logical task
- High cohesion allows
  - Understand what a class or method does
  - Reuse of classes or methods

Cohesion of methods

- A method should be responsible for one and only one well defined task
- Classes should represent one single, well defined entity

Code duplication

- Indicator of bad design
- Makes maintenance more difficult
- Can lead to maintenance induced errors
- May indicate lack of cohesion

Localizing change

- One aim of reducing coupling is to localize change.
- When a change is needed, as few classes as possible should be affected.

Thinking ahead

- When designing a class, we try to think what changes are likely to be made in the future.
- We aim to make those changes easy.

Refactoring

- Revising class structure, without modifying functionality
  - Divide class into multiple classes
  - Move methods to another class
  - Define interface from class operations
  - Rename classes or methods
  - Modify package structure
- Eclipse supports a number of refactoring operations

Revisted Junit and Logging

Junit Revisted

- Annotations
  - Adding metadata to Java source code that can also be available to the programmer at run-time
  - Alternative to XML technology
  - http://en.wikipedia.org/wiki/Java_annotation
- Debugging/Println/Logging
- Running Tests

Junit Revisted 2

- **Main method -> console**
  - `org.junit.runner.JUnitCore.main ("edu.washington.BathTest");`
- `Suites`
  - `org.junit.runner.JUnitCore.main ("edu.washington.AllTests");`

Logging

- **Logger logger = Logger.getLogger ("org.foo");**

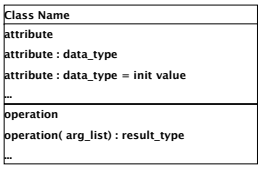
UML

Object Modeling

- Allows the communication of a design.
- The Unified Modeling Language (UML) is a well-accepted modeling language.
  - Class Diagram
    - Class
    - Association/Aggregation
    - Inheritance
  - Scenario/Sequence Diagram

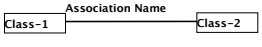
Class Notation

- Central to the class diagram is the class. A class may choose not to depict attributes and operations. The amount of detail specified about attributes and operations is also discretionary.



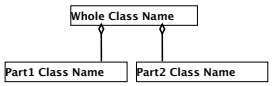
Association Notation

- Associations are depicted as lines between two classes. A solid arrowhead may be added indicate the direction of a relationship. An association simply indicates that one class has or can obtain a reference to the other. The association name is optional.



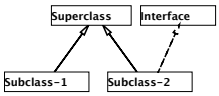
Aggregation Notation

•Aggregation is a more specific kind of association. Aggregation expresses a part-whole relationship. Aggregation is depicted like an association, with the addition of a diamond terminator on the aggregating class's end of the line. Aggregation is commonly referred to as a "has a" relationship.



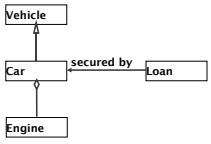
Inheritance Notation

•Inheritance is depicted simply as arrows with oversized, hollow, heads pointing from the subclass to the superclass or interface. Inheritance is commonly referred to as an "is a" relationship.



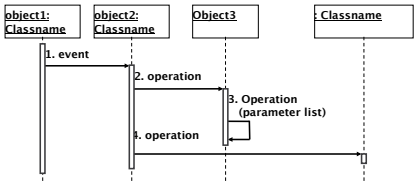
Vehicle Example

- The diagram says:
  - A car "is a" type of vehicle.
  - A car "has a" engine.
  - A loan is associated with the car.



Scenario/Sequence Diagram

•Shows how objects in the system interact.



Mid-Course Review

OO Concepts

- Abstraction
- Encapsulation
- Inheritance
- Polymorphism

Executing…

- Applications
  - The main method
- JUnit Tests
  - JUnit `TestCase`
    - `assertXxx…` methods
  - Test methods

UML Fundamentals

- UML constructs
  - Classes
  - Hierarchies
  - Association
  - Sequences

Java Basics

- Variable declarations
  - Types
- Expressions
  - Variables
  - Operators

Java Constructs

- Classes
  - Methods
  - Fields/Attributes/Variables
- Branching
  - `if/else`
  - `switch`
- Looping
  - `while`
  - `for`
  - `do/while`
- Arrays

Type Manipulation

- Casting
- Generics
- Boxing

Tools

- Compiler
  - `javac`
- Virtual Machine
  - `java`
- Document extraction
  - `javadoc`
- Classpath

Library Classes

- **Collections**
  - `ArrayList`
  - `HashMap`
  - `Iterator`
- **String helpers**
  - `StringBuffer`
  - `StringBuilder`

Modifiers

- **Visibility**
  - `public`
  - `private`
  - `protected`
  - `package` **(default)**
- **Other**
  - `static`
  - `final`

Mid-Course Assignment