Object-oriented Design

Chapter 2

Objectives

You will learn:

- the concepts of cohesion and coupling.
- difference between information and functional cohesion in system design.
- the use of encapsulation in OOD.
- how to recognize the practical usage of an abstract data type.

Objectives

You will learn:

- what a class is and its components including constructor, destructor, and members.
- how to create and utilize the advantages associated with inheritance.
- polymorphism and where it is used in OOD and OOP.

Modules

- A lexically contiguous sequence of program statements, bound by boundary elements, with an aggregate identifier.
- Cohesion
  - Degree of interaction within a module.
- Coupling
  - Degree of interaction between modules.
- Goals are:
  - (1) Maximal interaction within module.
  - (2) Minimal interaction between modules.

Cohesion

- There are seven categories or levels of cohesion (non-linear scale):

7.	Informational cohesion	(Good)
	Functional cohesion	
5.	Communicational cohesion	
4.	Procedural cohesion	
3.	Temporal cohesion	
2.	Logical cohesion	
1.	Coincidental cohesion	(Bad)

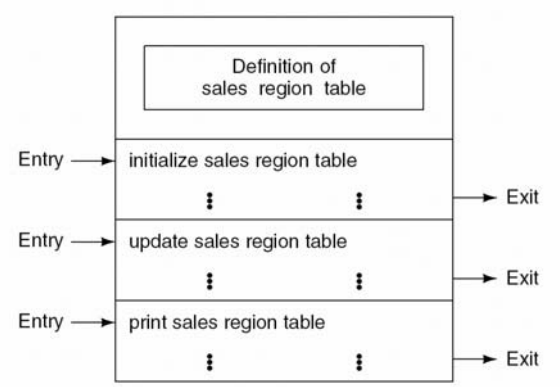
2: 5

Informational Cohesion

- A module has informational cohesion if:
  - it performs a number of actions.
  - each has its own entry point, with independent code for each action.
  - all are performed on the same data structure.

SYS-ED/Computer Education Techniques, Inc. 2: 6

Why Informational Cohesion is Important



- This is essentially an abstract data type.

SYS-ED/Computer Education Techniques, Inc. 2: 7

Functional Cohesion

- Module with functional cohesion performs exactly one action.
  - Example 1: Get temperature of furnace.
  - Example 2: Compute orbital of electron.
  - Example 3: Calculate sales commission.

SYS-ED/Computer Education Techniques, Inc. 2: 8

Why Functional Cohesion is Important

- More reusable.
- Corrective maintenance is easier:
  - Fault isolation.
  - Fewer regression faults.
- Easier to extend the product.

Coupling

- There are seven categories or levels of cohesion (non-linear scale):

5.	Data coupling	(Good)
4.	Stamp coupling	
3.	Control coupling	
2.	Common coupling	
1.	Content coupling	(Bad)

Data Coupling

- Two modules are data coupled when all parameters are homogeneous data items; simple parameters, or data structures all of whose elements are used by the called module.
- Examples:
  - Display time of arrival (flight number).
  - Compute product (first number, second number).
  - Get job with highest priority (job queue).

Why Data Coupling is Important

- The difficulties of content, common, control, and stamp coupling are not present.
- Maintenance is easier.

Information Hiding

- Data abstraction
  - Designer thinks at the level of an ADT.
- Procedural abstraction
  - Define a procedure-extend the language.
- Instances of a more general design concept, information hiding.
  - Design the modules in way that items likely to change are hidden.
  - Future change is localized.
  - Changes cannot affect other modules.

Class

- A representation of real world.
- Consist of data and its operation.

Class Rule

- API: Application Programming Interface
- Representation of concept
- Completion
- Robust

Class Member

- Data Member
- Member function (Method)
- Static or nonstatic members

Class Operation

- Constructor and destructor
- Inheritance
- Overloading and overriding method
- Access control
- Abstraction

Constructor and Destructor

- Constructor is called when the object is created.
- Destructor is called when the object is destroyed.

Inheritance

- Class can inherit from another class.
  - Child class.
  - Parent class.
- Child has the same ability as parent; plus its own specialization.

Overloading and Overriding

- Overloading:
  - Same method name and signature, but different class.
- Overriding:
  - Same method name, but different signature and class.
  - A unique method signature usually includes the method name, the number and type of its parameters, and its return type.

Access Control

- Member should be protected from the environment.
- Programming languages support a variety of access control:
  - private
  - protected
  - public

Abstraction

- Class and method can be an abstract.
- Abstract method must be in abstract class.
- Cannot be instanced.
- Has to inherit and override the abstract method.

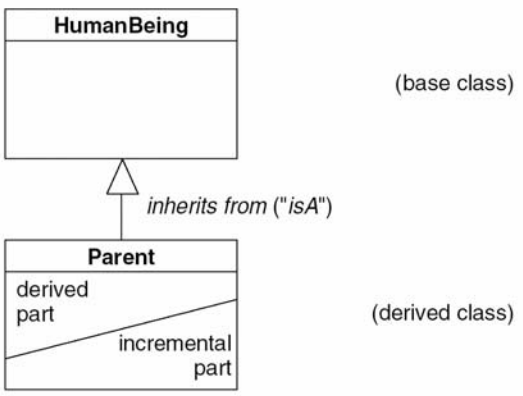
Objects

- First refinement:
  - Product is designed in terms of abstract data types.
  - Variables ("objects") are instantiations of abstract data types.
- Second refinement:
  - Class: abstract data type that supports inheritance.
  - Objects are instantiations of classes.

Inheritance

- Define **humanBeing** to be a *class.*
  - A **humanBeing** has attributes, such as age, height, gender.
  - Assign values to attributes when describing object.
- Define Parent to be a subclass of HumanBeing.
  - A Parent has all attributes of a HumanBeing, plus attributes of his/her own (name of oldest child, number of children).
  - A Parent inherits all attributes of **humanBeing.**
- The property of inheritance is an essential feature of object-oriented languages.

Inheritance

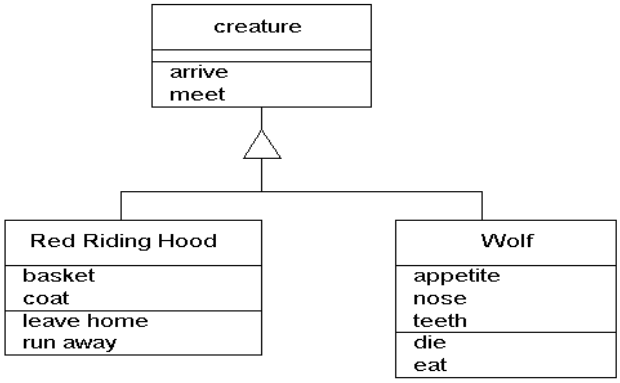


- UML notation
  - Inheritance is represented by a large open triangle.

Java Implementation

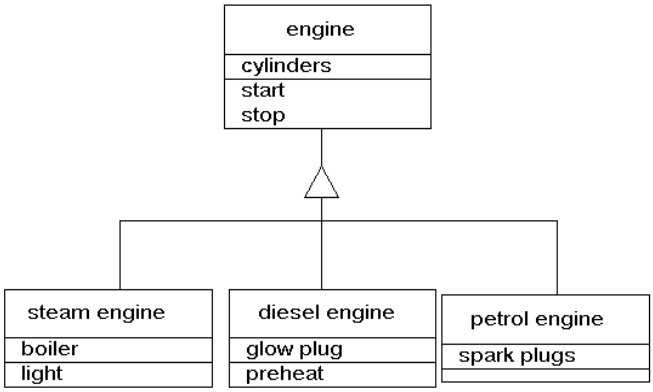
```
class HumanBeing
{
  private int      age;
  private float    height;

  // public declarations of operations on HumanBeing

}// class HumanBeing


class Parent extends HumanBeing
{
  private String   nameOfOldestChild;
  private int      numberOfChildren;

  // public declarations of operations on Parent

}// class Parent
```

Inheritance Example



SYS-ED/Computer Education Techniques, Inc. 2: 27

Inheritance Example

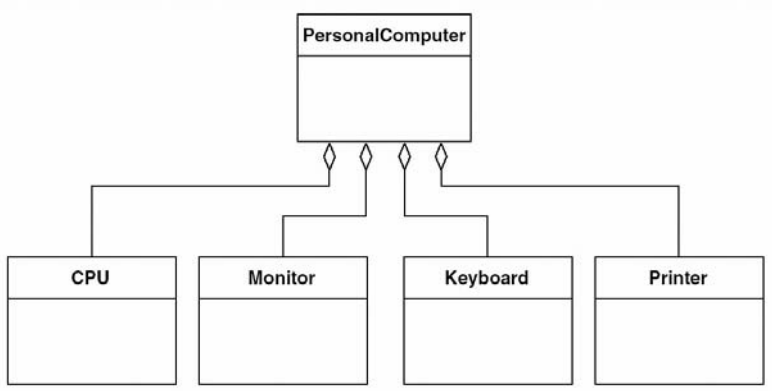


SYS-ED/Computer Education Techniques, Inc. 2: 28

Aggregation

- Aggregation differs from ordinary composition in that it does not imply ownership.
  - In composition, when the owning object is destroyed, so are the contained objects.
  - In aggregation, this is not necessarily true.
  - For example, a university owns various departments (e.g., chemistry), and each department has a number of professors.
    - If the university closes, the departments will no longer exist, but the professors in those departments will continue to exist.
    - Therefore, a university can be seen as a composition of departments
    - Departments have an aggregation of professors.
  - In addition, a professor could work in more than one department; but a department could not be part of more than one university.

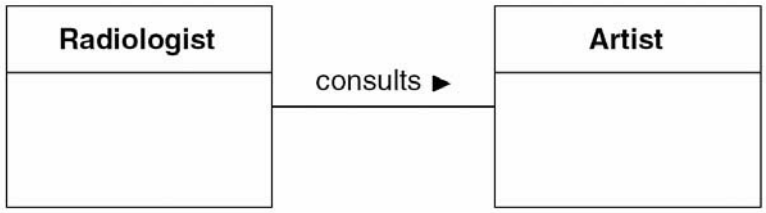
Aggregation



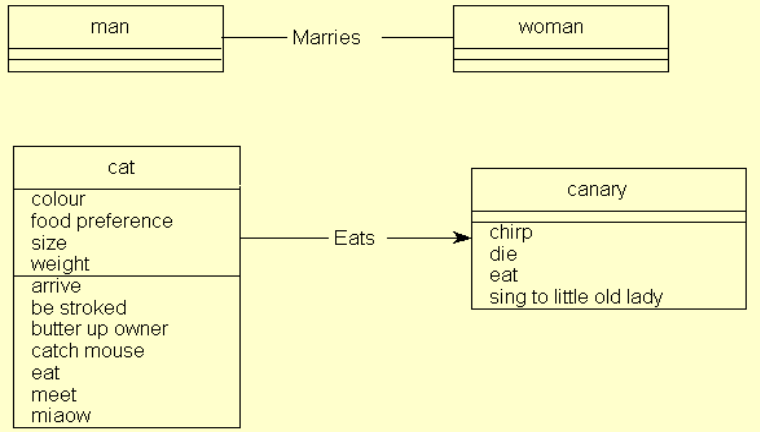
Association

- Association defines a relationship between classes of objects which allows one object instance to cause another to perform an action on its behalf.
- This relationship is structural, because it specifies that objects of one kind are connected to objects of another.
- Unless otherwise specified, navigation across an association is bidirectional, although it may be limited to just one direction by adorning some end with an arrowhead pointing to the direction of traversal.
- In generic terms the causation is usually called "sending a message", "invoking a method" or "calling a member function" to the controlled object. Concrete implementation usually requires the requesting object to invoke a method or member function using a reference or pointer to the memory location of the controlled object.

Association

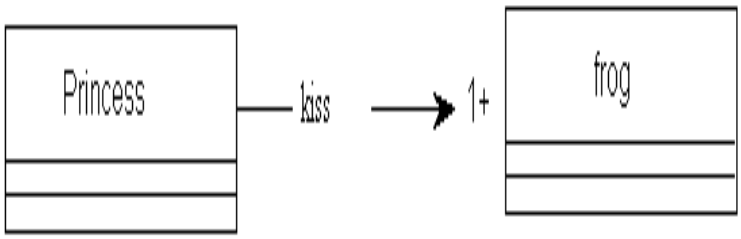


One to One Relationships



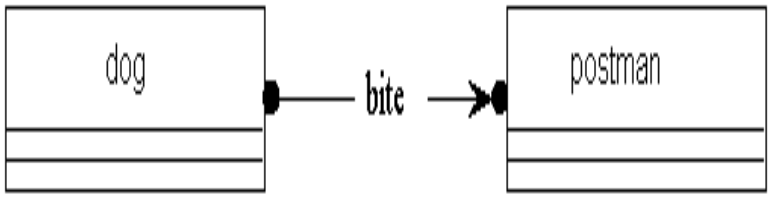
SYS-ED/Computer Education Techniques, Inc. 2: 33

One to Many Relationships



SYS-ED/Computer Education Techniques, Inc. 2: 34

Many to Many Relationships



Advantages of Objects

- Same advantages as abstract data types.
  - Information hiding.
  - Data abstraction.
  - Procedural abstraction.
- Inheritance provides additional data abstraction.
  - Easier and less error-prone product development.
  - Easier maintenance.
- Objects are more reusable than modules with functional cohesion.

Association and Aggregation Lab

Which of the following is association and the other aggregation?

1. Houses on a street.
2. Pages in a book.
3. Notes in a symphony.
4. Components in a home entertainment system (TV, VCR, tape deck, etc.).

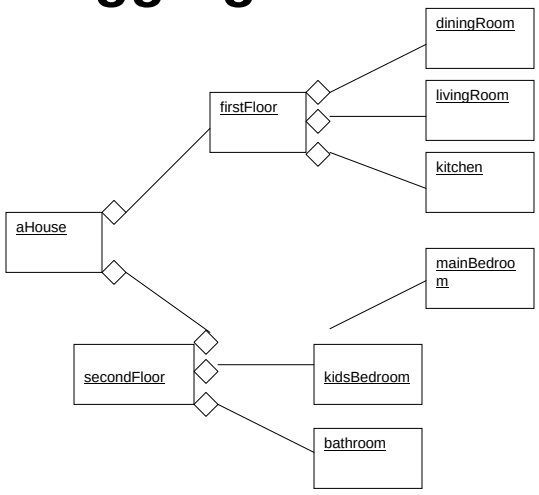
Association and Aggregation Solution

1. Association
   - A street exists independent of its houses.
2. Aggregation
   - If a pager is ripped out and the book is no longer useful, aggregation would be the right choice.
   - However, if the book was made to remove pages, it would be association.
3. Aggregation
   - Notes are similar to pages in a book.
4. Association
   - The pieces are all independent.

Aggregation Lab

- Aggregation means putting objects together to make a larger object.
- For example, a house can have two floors and three rooms on each floor.
  - The first floor has a dining room, living room, and a kitchen.
  - The second floor has a main bedroom, a kids bedroom and a bathroom.
- Draw a diagram to depict the aggregation of a house.

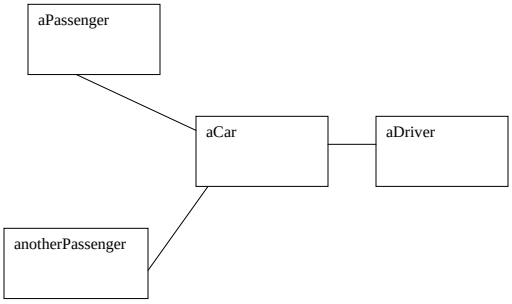
Aggregation Solution



Association Lab

- Association is a weak form of a connection.
- For example, consider a car, a driver, a passenger and another passenger. They occupy the same volume in space but the association is loose. The driver can drop off a passenger to go their separate way.
- Draw a diagram using association to depict the above objects.

Association Solution



Coffee Machine Object

- Consider a coffee machine object.
- First it will be necessary to define what operations a coffee machine needs:
  - Display drinks
  - Select drink
  - Accept money
  - Dispense drink
- Next it will be necessary to state what the coffee machine needs to know in order to perform these operations:
  - Available drinks
  - Drink prices
  - Drink recipes
- Draw a class diagram for a coffee machine object.

Coffee Machine Object