UML

Diagram Types

- structural-gives us the structural relationships between system
- behavioral-concerned with the execution of the system

Class Model Diagram

- most popular
- also called static model
- it is an example of a structured diagram (shows system structures
- shows classes and relationships

UML classes

- 3 compartments separated by horizontal lines

Counter
- count : int
+increment():void +display():void +set(aCount : int=0):void

Name of the class
attributes of the class (instance variables)
methods and operations the class provides

another formal example:

An ADT representing a rectangle:

```
class Rectangle
{
 private:
    double width;
    double length;
    string name;
    void initName(string n);
  public:
  //constructors
  Rectangle();
  Rectangle(double, double,
                string);
  //destructor
    ~Rectangle();
    void setWidth(double);
    void setLength(double);
    void setName(string);
    double getWidth() const;
    double getLength() const;
};
```

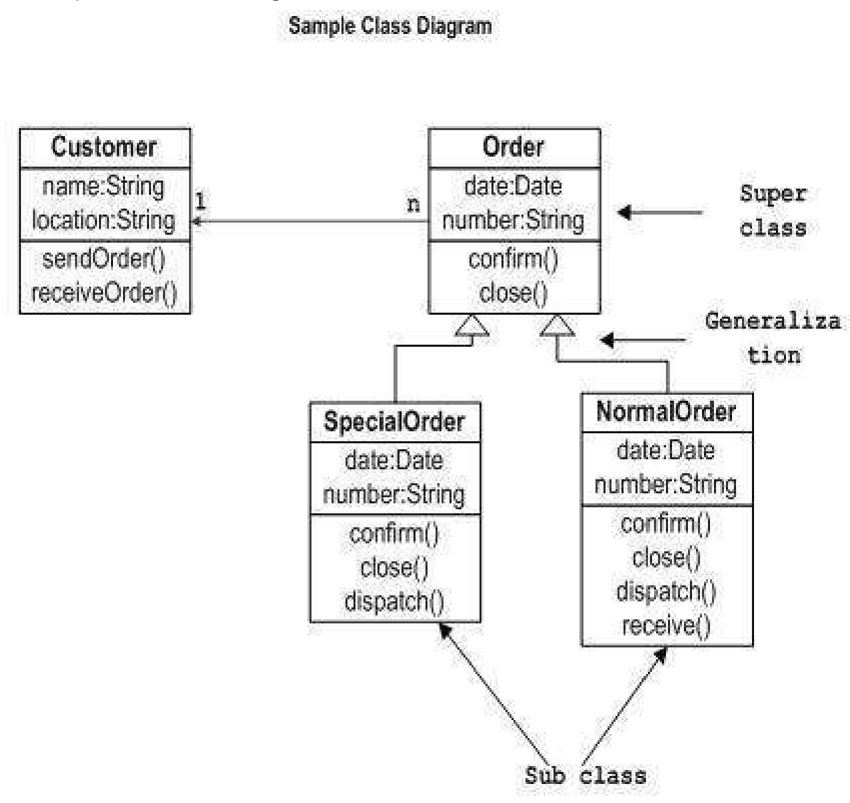
Rectangle
- width: double - length: double - name: string

+ Rectangle() :
+Rectangle(w : double, l : double, n : string) :
- initName(n : string) : void
+setWidth(w : double) : void
+setLength(l : double) : void
+setName(n : string) : void
+getWidth() : double {query}
+getLength() : double {query}

UML Relationships

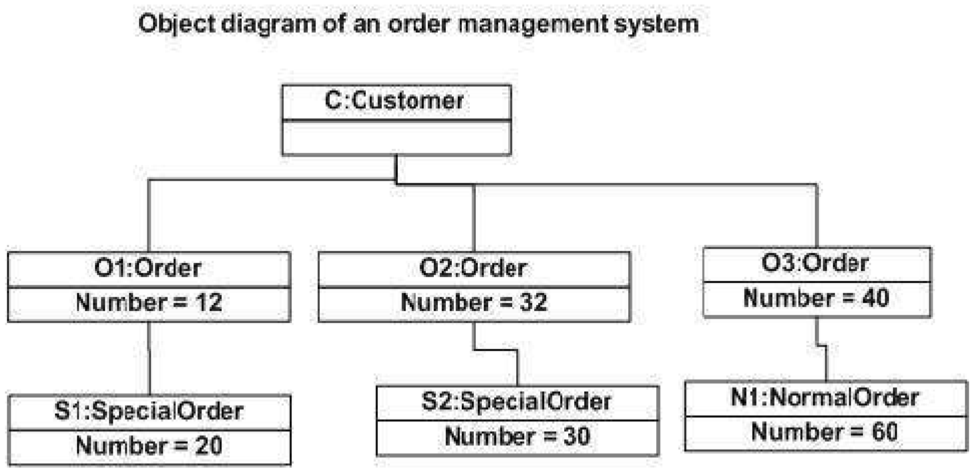
- dependency: x uses y - - - - - - > (dashed direct line)
- associations: x affects Y ______ (solid undirected line)
- generalization: x is a kind of y ____>(solid with open arrowhead)

Example of class diagram



Object Diagram

- conveys objects and links instead of classes and relationships
- name is composed by the class name and the specific instance
- for example in the first box the class is customer and the instance is c, they are separated by a colon
- some instances also have their attribute fields with values filled in (example in the class order,the instance o1 has the attribute number = 12



Review on terminology

- A solution is said to be efficient if it solves a problem within required resource constraints (example total space or time to perform a task
- the steps to select a data structure are:
  - Analyze your problem to determine the basic operations that must be supported.Examples of basic operations include inserting a data item into the data structure, deleting a data item from the data structure, and finding a specified data item.
  - Quantify the resource constraints for each operation.
  - Select the data structure that best meets these requirements.

- the questions regarding the operation constraints should be:
  - Are all data items inserted into the data structure at the beginning, or are insertions interspersed with other operations? Static applications (where the data are loaded at the beginning and never change) typically require only simpler data structures to get an efficient implementation than do dynamic applications.
  - Can data items be deleted? If so, this will probably make the implementation more complicated
  - Are all data items processed in some well-defined order, or is search for specific data items allowed? “Random access” search generally requires more complex data structures

- A type is a collection of values (for example an int is a simple type because its value contains no subparts)
  - an aggregate/composite type contains several pieces of data (example a bank account-address, number, name, etc)
- A data item is a piece of info whose value is drawn from a type (it is a member of a type)

- A data type is a type together with a collection of operations to manipulate the type. For example, an integer variable is a member of the integer data type. Addition is an example of an operation on the integer data type.
- An ADT is the realization of a data type as a software component. The interface of the ADT is defined in terms of a type and a set of operations on that type. The behavior of each operation is determined by its inputs and outputs. An ADT does not specify how the data type is implemented. These implementation details are hidden from the user of the ADT and protected from outside access, a concept referred to as encapsulation.
- A data structure is the implementation for an ADT
    - ADT+implementation=class
    - operations associated to ADT are implemented by member functions
    - variables that define space required by data item are data members
    - an object is an instance of a class