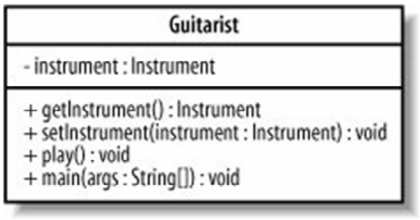
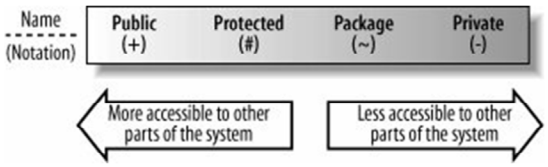
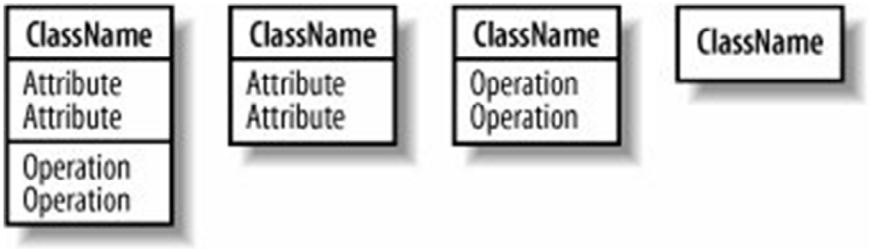
UML

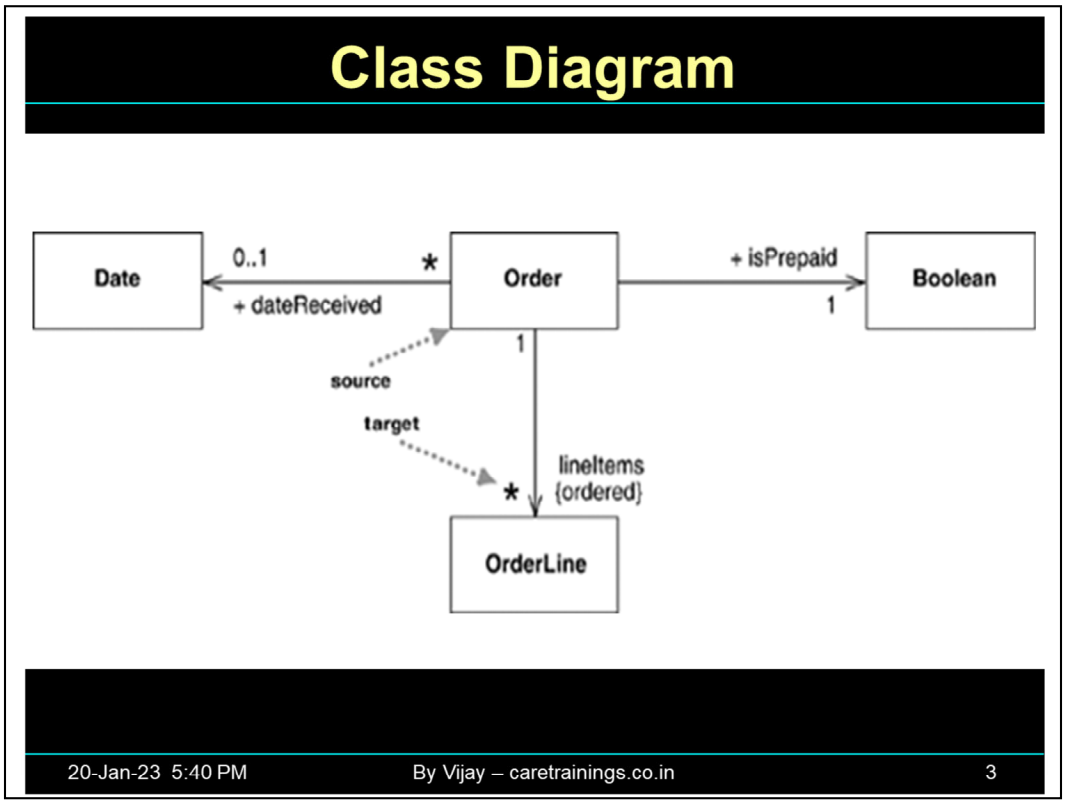
Class Diagram

Class in UML

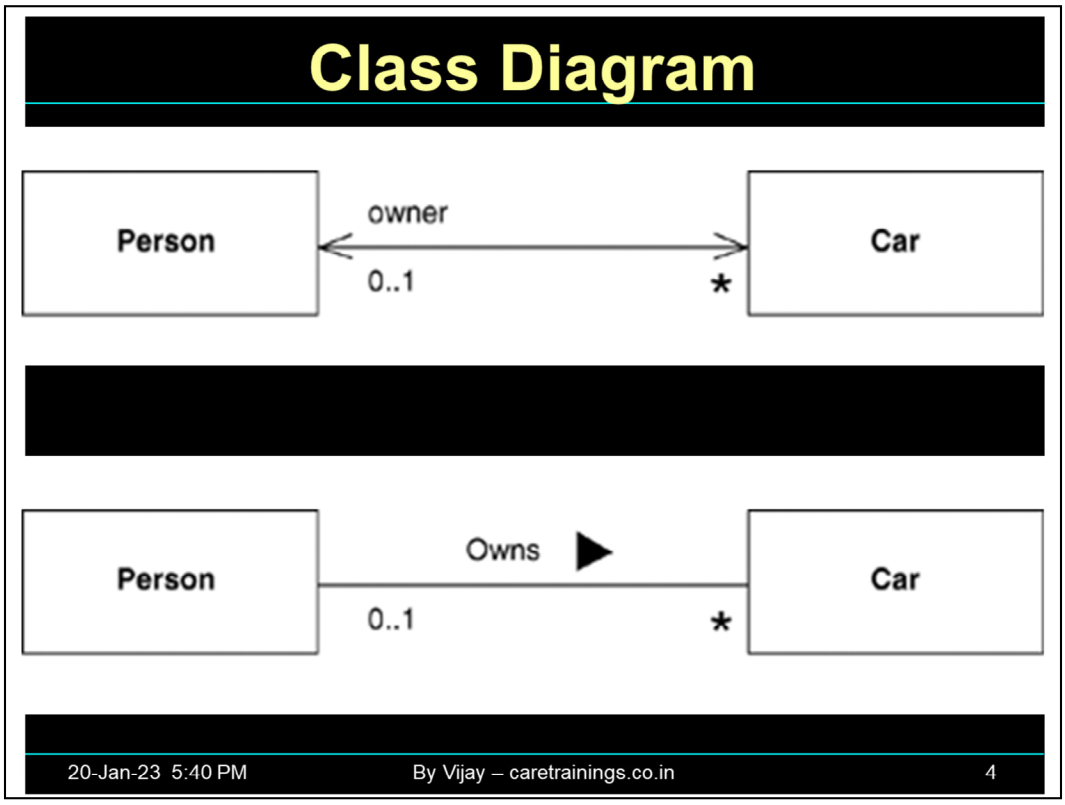

Guitarist
- instrument : Instrument
+ getInstrument() : Instrument
+ setInstrument(instrument : Instrument) : void
+ play() : void
+ main(args : String[]) : void


Name
(Notation)
Public (+)
Protected (#)
Package (~)
Private (-)
More accessible to other parts of the system
Less accessible to other parts of the system


ClassName
Attribute
Attribute
Operation
Operation
ClassName
Attribute
Attribute
ClassName
Operation
Operation
ClassName

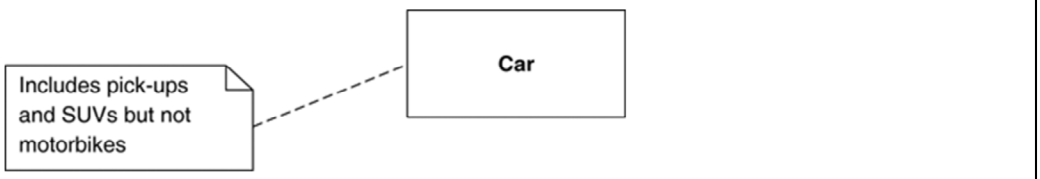


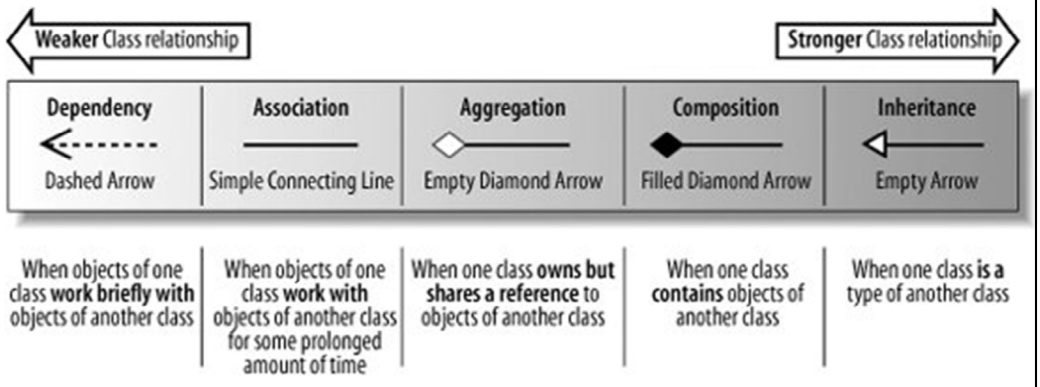

```
using namespace std;

class Boolean {};
class Date {};
class OrderLine {};
class Order {
    vector<OrderLine*> lineItems;
public:
    Date* dateReceived;
    Boolean isPrepaid;
};
```



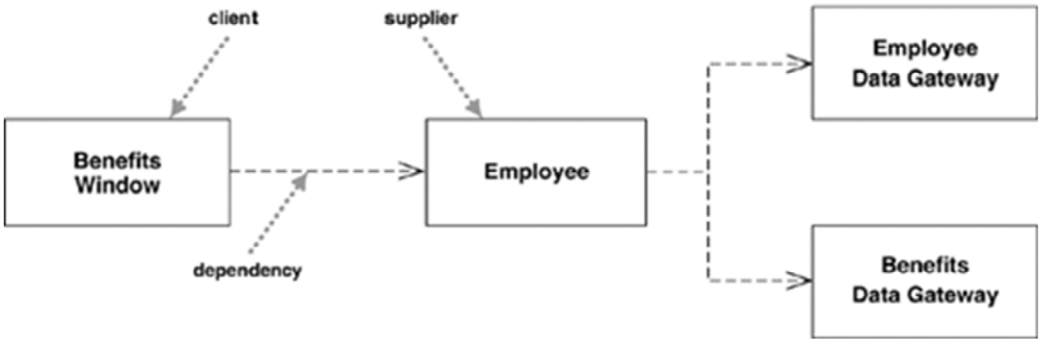

```
using namespace std;
class Car;
class Person {
    vector<Car*> cars;
};
class Car {
    Person* owner;
};
```

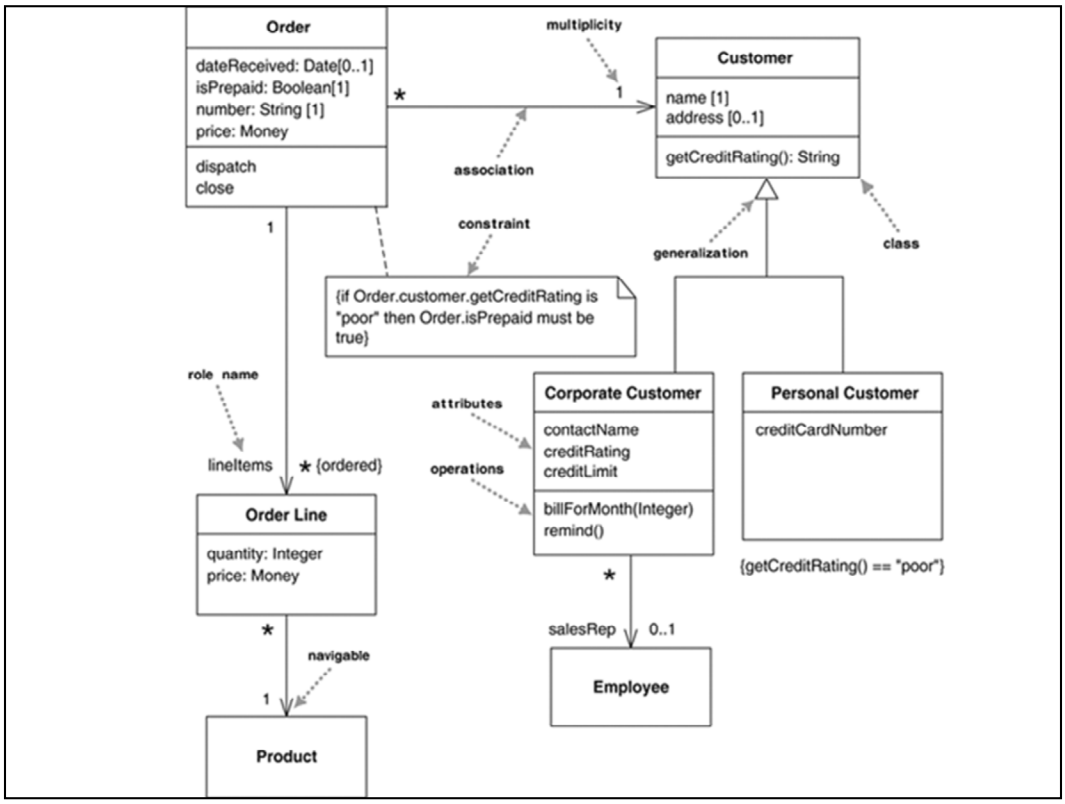
Comments and Relationships





Dependency


client
supplier
Benefits Window
Employee
dependency
Employee Data Gateway
Benefits Data Gateway

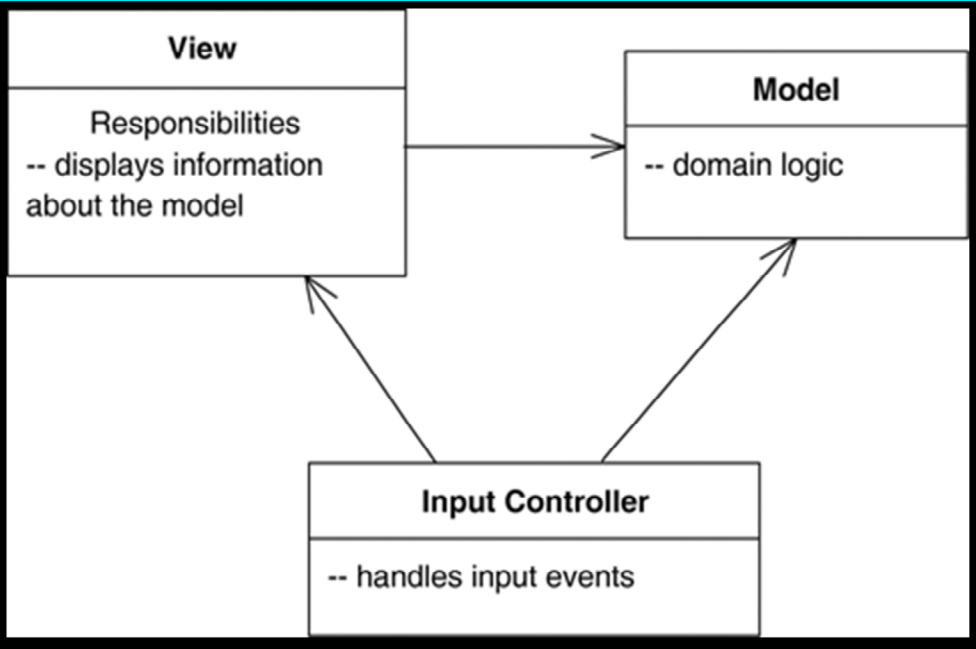



```
using namespace std;
class Customer {
    string name;
    string address;
public:
    virtual string getCreditRating() {
return "..."; }
    ~Customer() {}
};
class Employee {};
class CorporateCustomer : public Customer
{
```

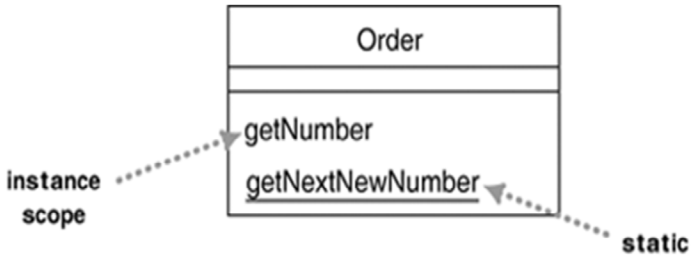
```
    string contactName, creditRating;
    double creditLimit;
    Employee* salesRep;
public:
    string billForMonth(int month);
    void remind();
};
class PersonalCustomer : public Customer {
    string creditCardNumber;
public:
    string getCreditRating() override {
return "poor"; }
};

class Boolean {};
class Date {};
class Money {};
class Product {};
class OrderLine {
    int quantity;
    Money price;
    Product* p;
};
```

```
class Order {
    string number;
    Money price;
    vector<OrderLine*> lineItems;
    Customer *c;
public:
    Date* dateReceived;
    Boolean isPrepaid;
    void dispatch();
    void close();
};
```

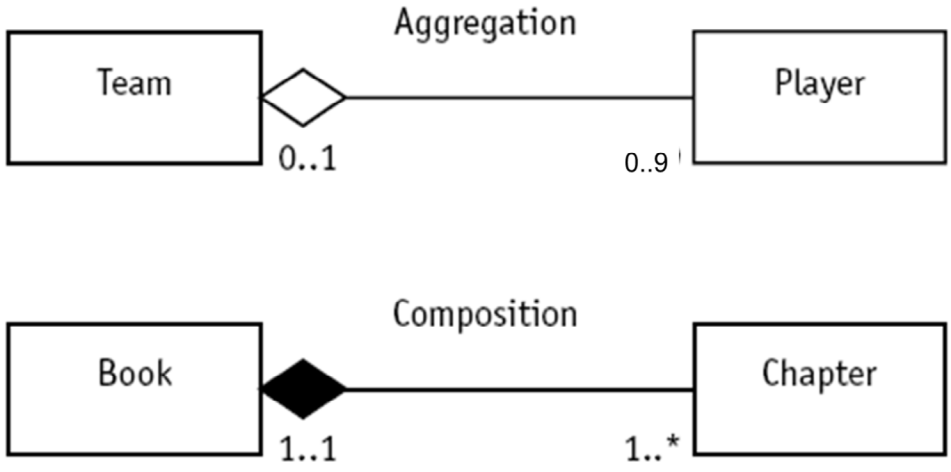
Showing Responsibilities



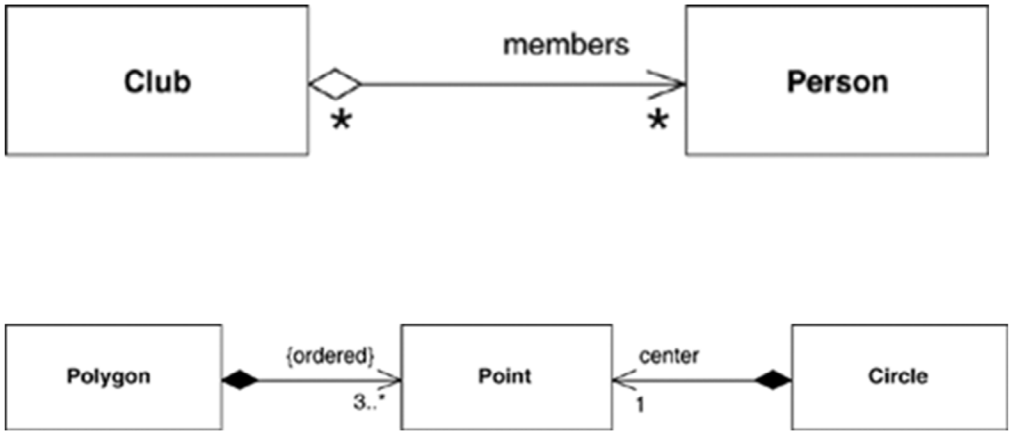
Notation



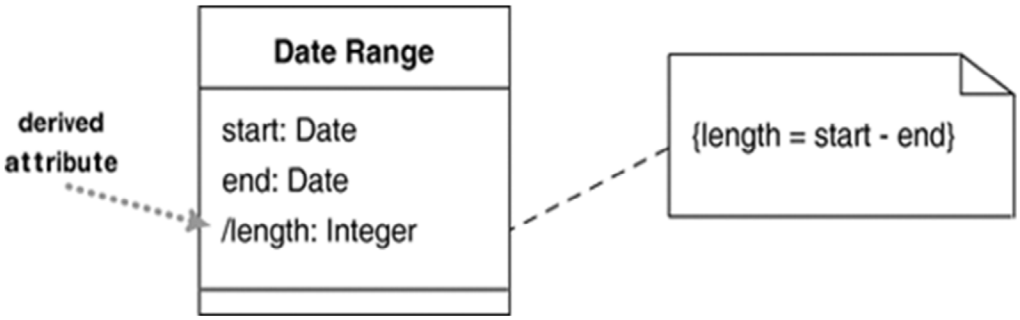
Aggregation & Composition



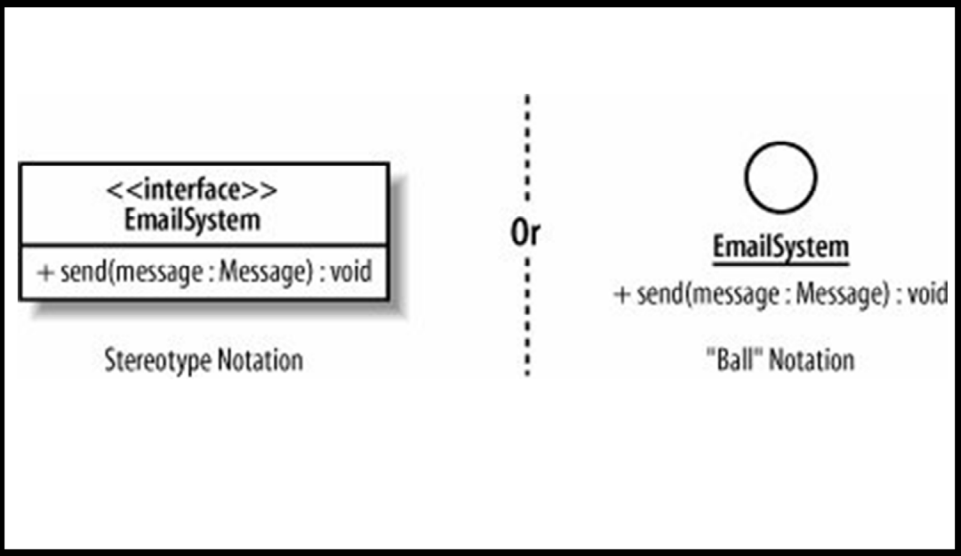
Aggregation & Composition



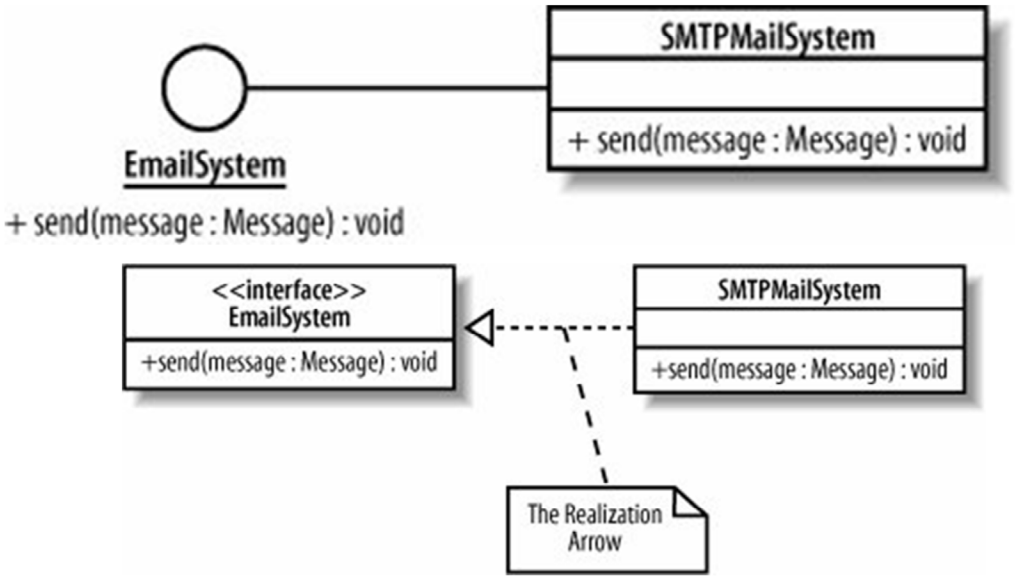
Derived


Date Range
start: Date
end: Date
/length: Integer
derived
attribute
{length = start - end}

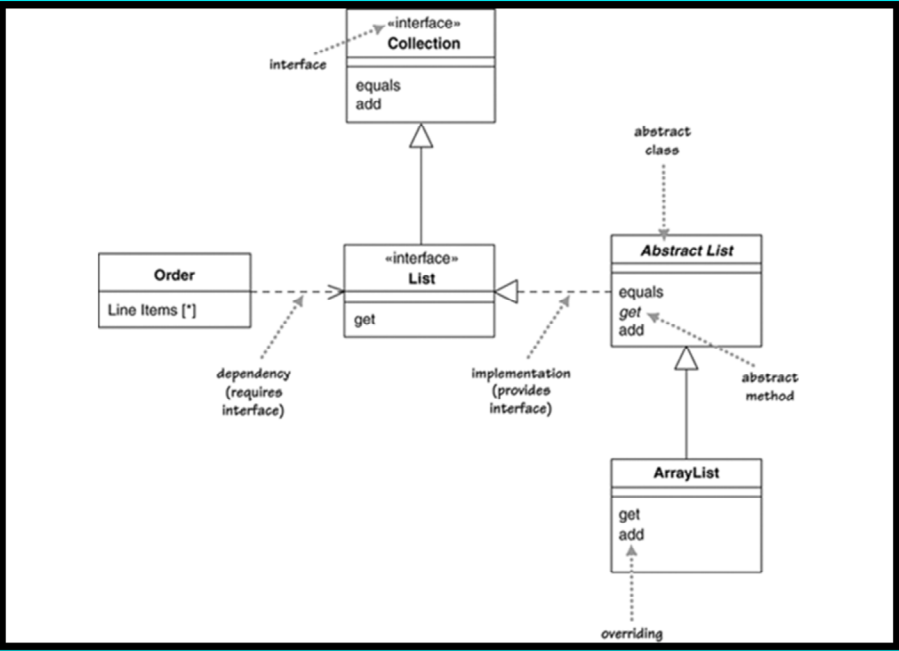
Interface notation



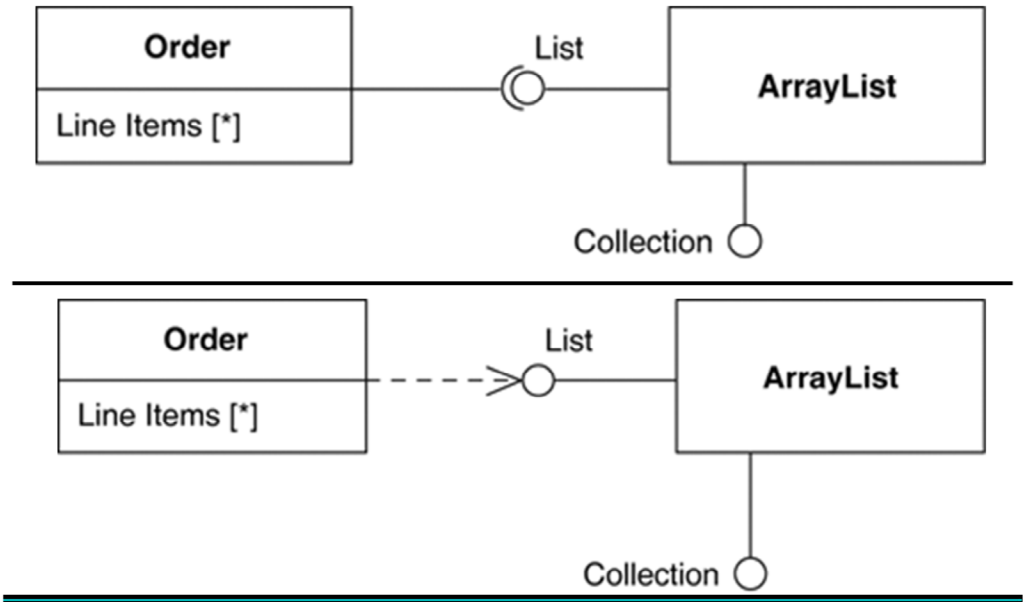
Class implementing Interface



Interface & Abstract classes

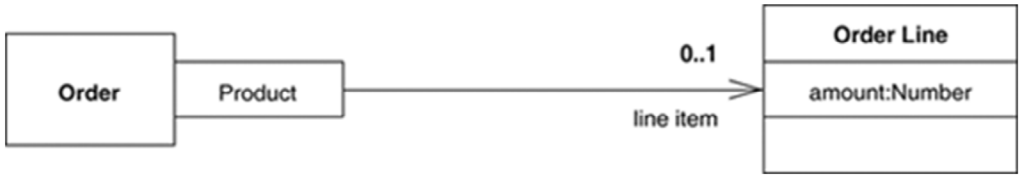


Interface

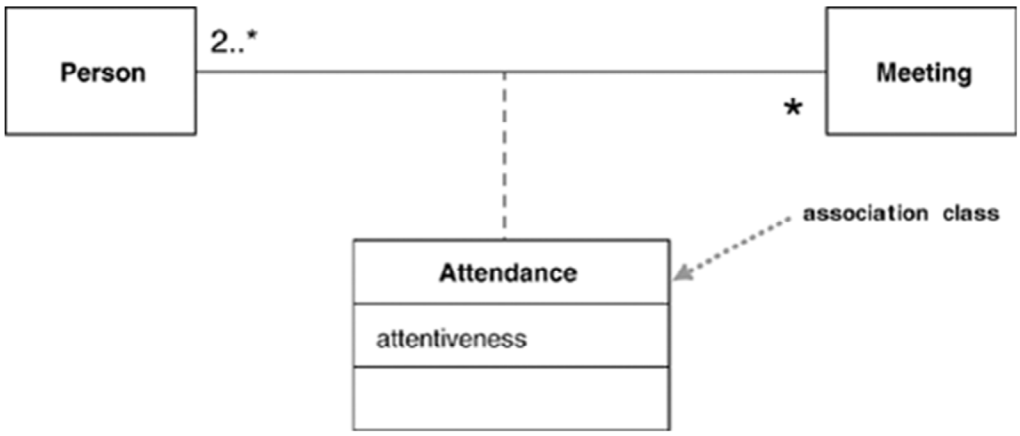

Order
Line Items [*]
List
ArrayList
Collection

Order
Line Items [*]
List
ArrayList
Collection

Qualified Association

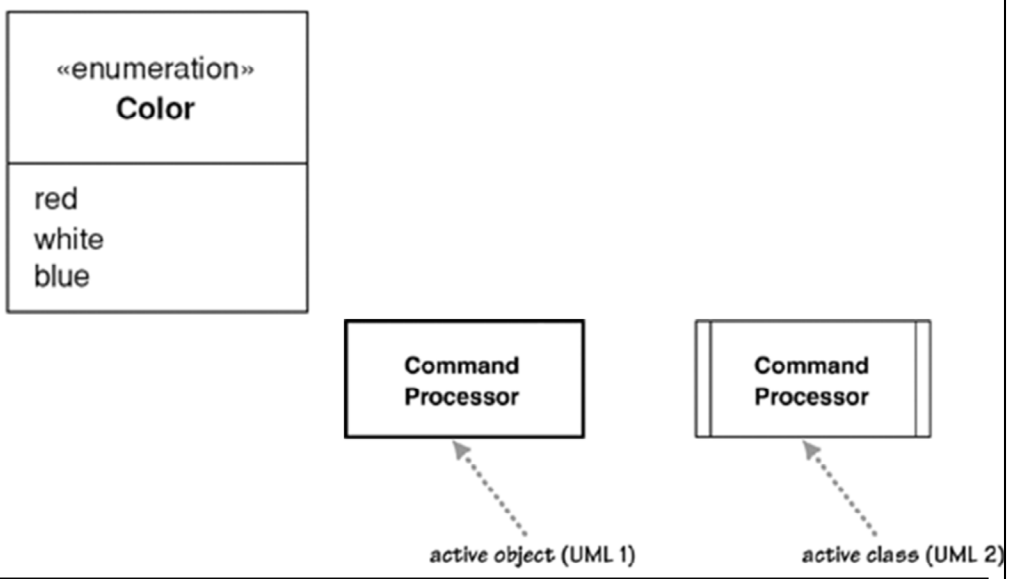


Association class

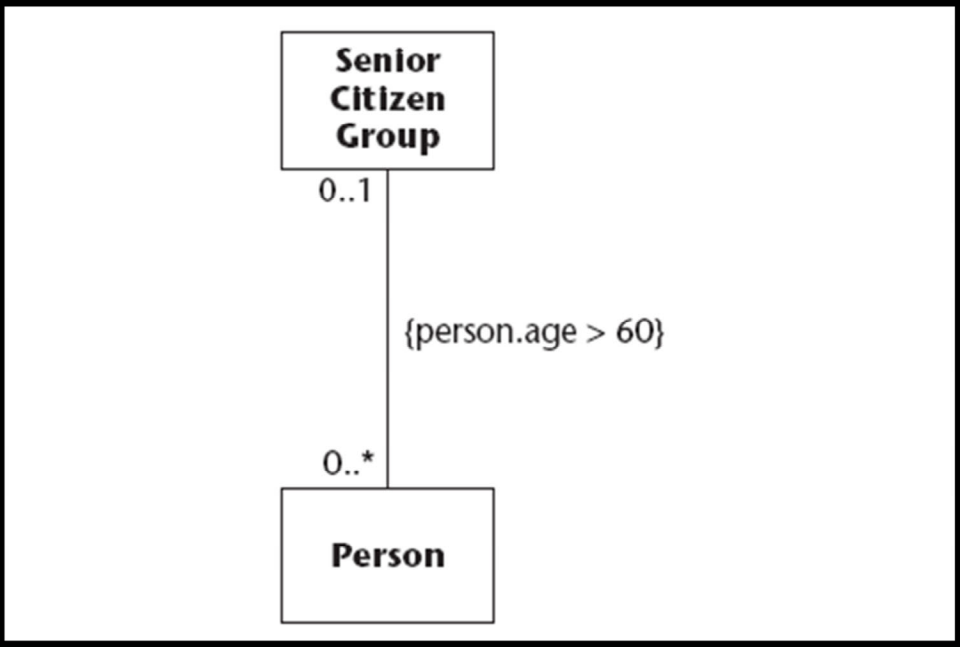


Temporal Relationship

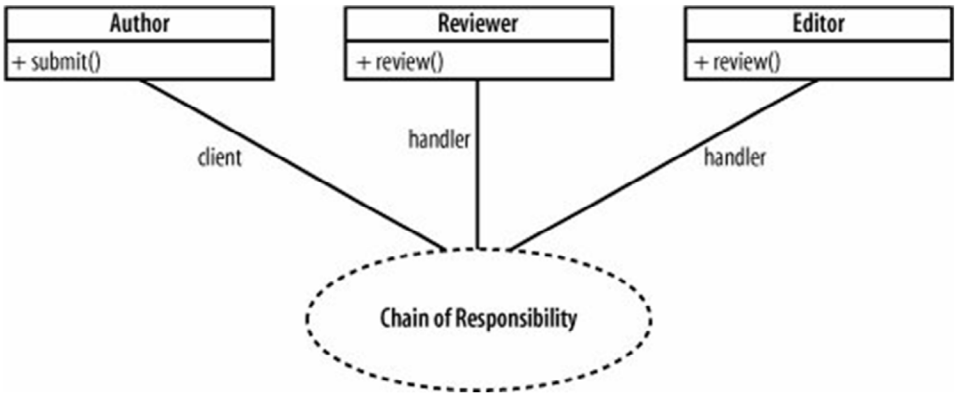
Enumerations & Active Objects



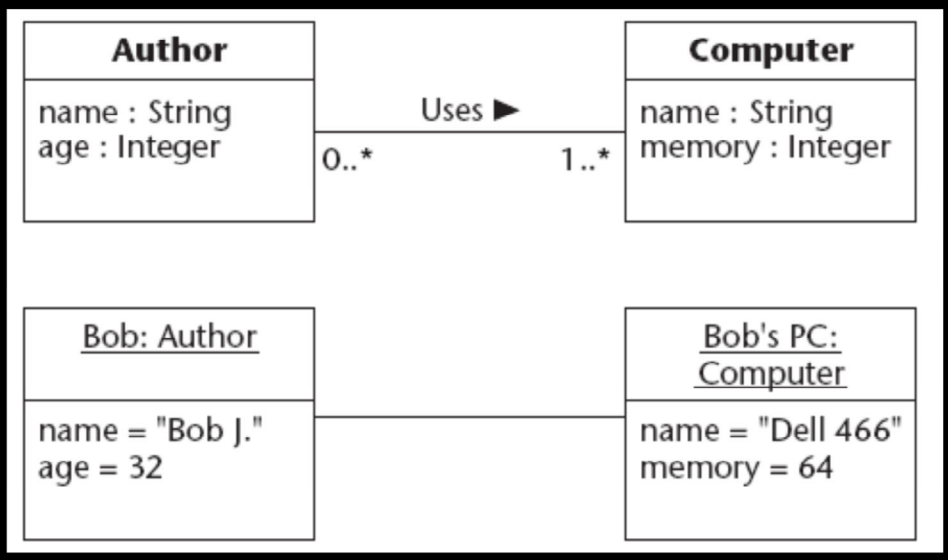
Constraints

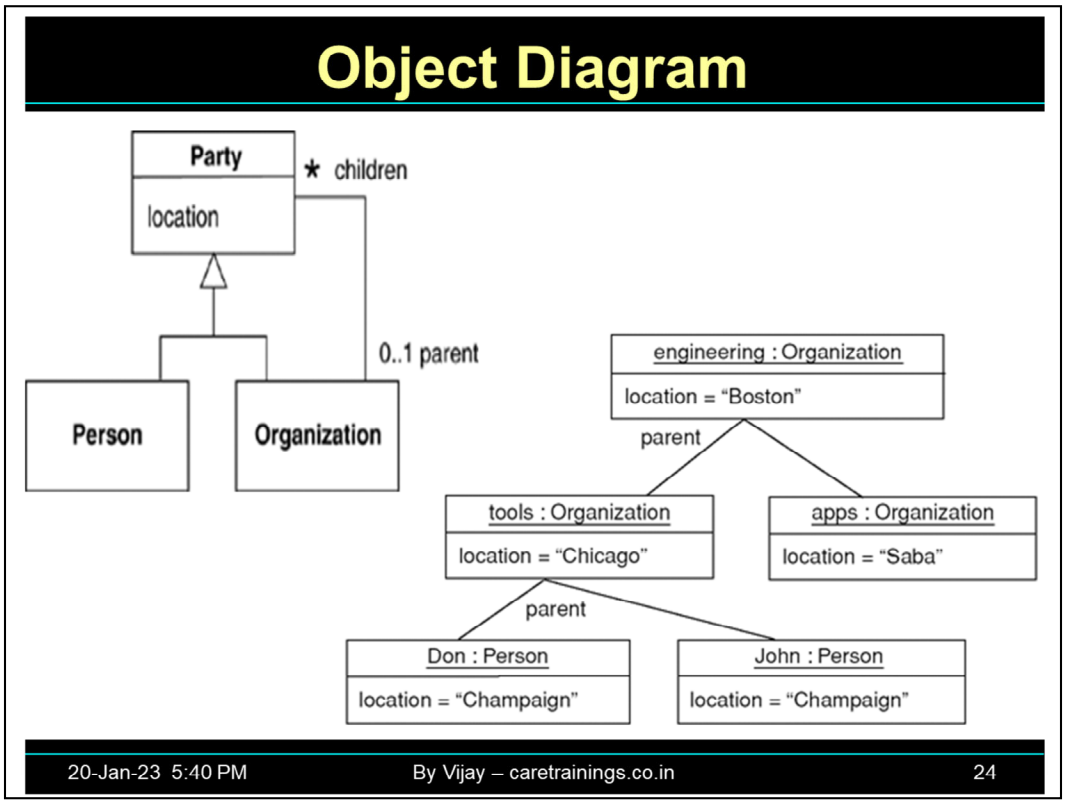


Design Patterns



Object Diagram






```
using namespace std;
class Party {
    string location;
public:
    Party(const char* loc) { location =
loc; }
};
class Person : public Party {
public:
    using Party::Party;
};
class Organization : public Party {
```

```
    vector <Party*> children;
public:
    using Party::Party;
    void addChild(Party* p) {
children.push_back(p); }
};
int main() {
    Person don("champaign"),
john("Champaign");
    Organization tools("Chicago"),
apps("Saba"), engineering("Boston");
    tools.addChild(&don);
    tools.addChild(&john);
    engineering.addChild(&tools);
    engineering.addChild(&apps);
}
```

Exercises

- Conceptual Class Diagram
  - Banking
  - Railway
  - Airline
- Software Class Diagram
  - Card and Chips
  - HR System
  - QCoffee