UML

διαγράμματα κλάσεων

Τμήμα Πληροφορικής και Τηλεπικοινωνιών (Άρτα)

Πανεπιστήμιο Ιωαννίνων

Γκόγκος Χρήστος

V 1.0

Τι είναι η UML;

- Η UML (Unified Modeling Language) είναι μια γλώσσα μοντελοποίησης, βασισμένη σε διαγράμματα, που υποστηρίζει την ανάπτυξη λογισμικού σε όλα τα στάδιά του. Ορίζει ένα κοινό λεξιλόγιο και κοινά αποδεκτούς τύπους διαγραμμάτων για τη περιγραφή του λογισμικού καθώς αναπτύσσεται εξετάζοντάς το υπό διάφορες οπτικές γωνίες
- Η πρώτη έκδοση της UML ανακοινώθηκε το 1997 και η τρέχουσα έκδοσή της είναι η UML 2.5.1 (Δεκέμβριος 2017)
  - https://www.omg.org/spec/UML/About-UML/

Σημαντικοί τύποι διαγραμμάτων στην UML

Διαγράμματα δομής

- Διάγραμμα κλάσεων (Class Diagram)
- Διάγραμμα συστατικών (Component Diagram)
- Διάγραμμα σύνθετης δομής (Composite Structure Diagram)
- Διάγραμμα αντικειμένων (Object Diagram)
- Διάγραμμα πακέτων (Package Diagram)
- Διάγραμμα ανάπτυξης/εγκατάστασης (Deployment Diagram)

Διαγράμματα συμπεριφοράς

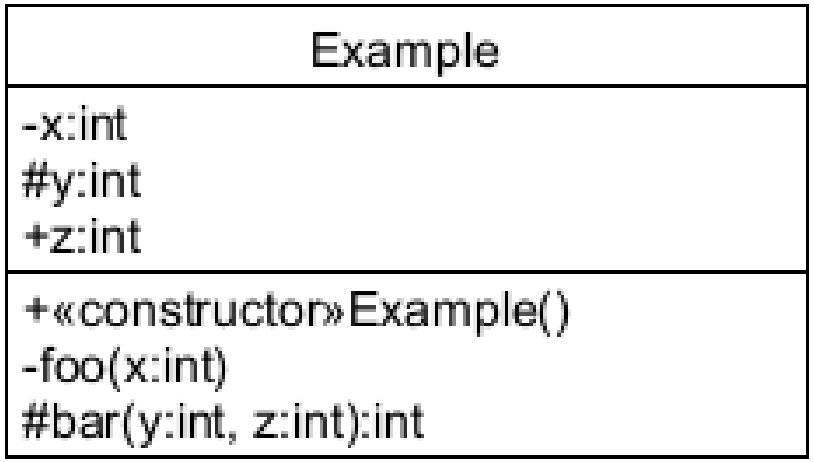
- Διάγραμμα περίπτωσης χρήσης (Use Case Diagram)
- Διάγραμμα δραστηριότητας (Activity Diagram)
- Διάγραμμα μηχανών καταστάσεων (State Machine Diagram)
- Διάγραμμα ακολουθίας (Sequence Diagram)
- Διάγραμμα επικοινωνίας (Communication Diagram)
- Διάγραμμα χρονισμού (Timing Diagram)

Διαγράμματα κλάσεων

- Τα διαγράμματα κλάσεων είναι τα πλέον κοινά διαγράμματα UML
- Τα διαγράμματα κλάσεων περιγράφουν:
  - τους τύπους των αντικειμένων
  - τις στατικές σχέσεις μεταξύ των αντικειμένων
  - τις ιδιότητες των κλάσεων
  - τις λειτουργίες των κλάσεων
  - τους περιορισμούς στον τρόπο με τον οποίο τα αντικείμενα συνδέονται μεταξύ τους
- Στην UML οι ιδιότητες (μέλη δεδομένων στη C++) και οι λειτουργίες (συναρτήσεις μέλη στη C++) μιας κλάσης αναφέρονται από κοινού ως χαρακτηριστικά (features)

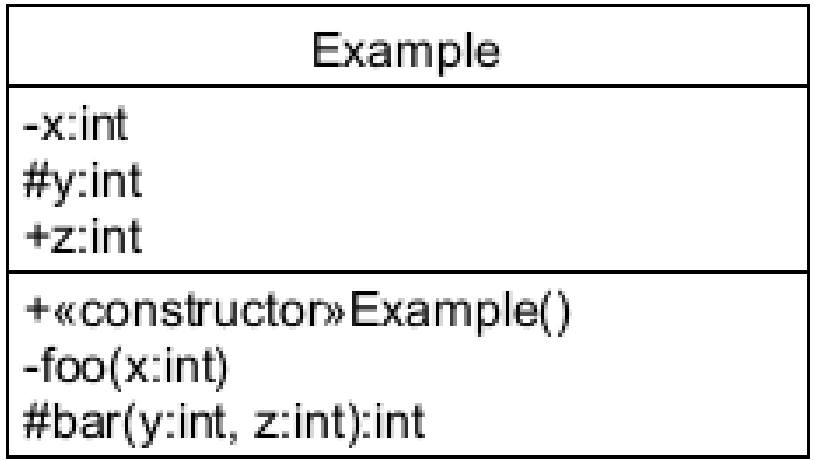
Απεικόνιση μιας κλάσης σε διάγραμμα κλάσεων

- Μια κλάση σε ένα διάγραμμα κλάσεων αναπαρίσταται με ένα τετράγωνο που χωρίζεται σε 3 τμήματα (το πολύ)
  - Στο 1ο τμήμα βρίσκεται το όνομα της κλάσης
  - Στο 2ο τμήμα βρίσκονται οι ιδιότητες
  - Στο 3ο τμήμα βρίσκονται οι λειτουργίες
- Κάθε χαρακτηριστικό της κλάσης επισημαίνεται με ειδικά σύμβολα(+,-,#) προκειμένου να προσδιοριστεί το επίπεδο προστασίας του
  - + για public
  - - για private
  - # για protected
- Για τα χαρακτηριστικά της κλάσης χρησιμοποιείται στυλ δήλωσης Algol
  - <αναγνωριστικό>:<τύπος> για ιδιότητες
  - <αναγνωριστικό>(<όνομα παραμέτρου>:<τύπος>):<τύπος επιστροφής> για λειτουργίες
- Επιπλέον, μπορούν να καταγράφονται μεταδεδομένα (metadata) με τη μορφή στερεότυπων που περιλαμβάνονται σε διπλά εισαγωγικά (guillemets)
  - π.χ. «constructor», «NotNull», «abstract», …



Κώδικας για απεικόνιση κλάσης σε διάγραμμα κλάσεων

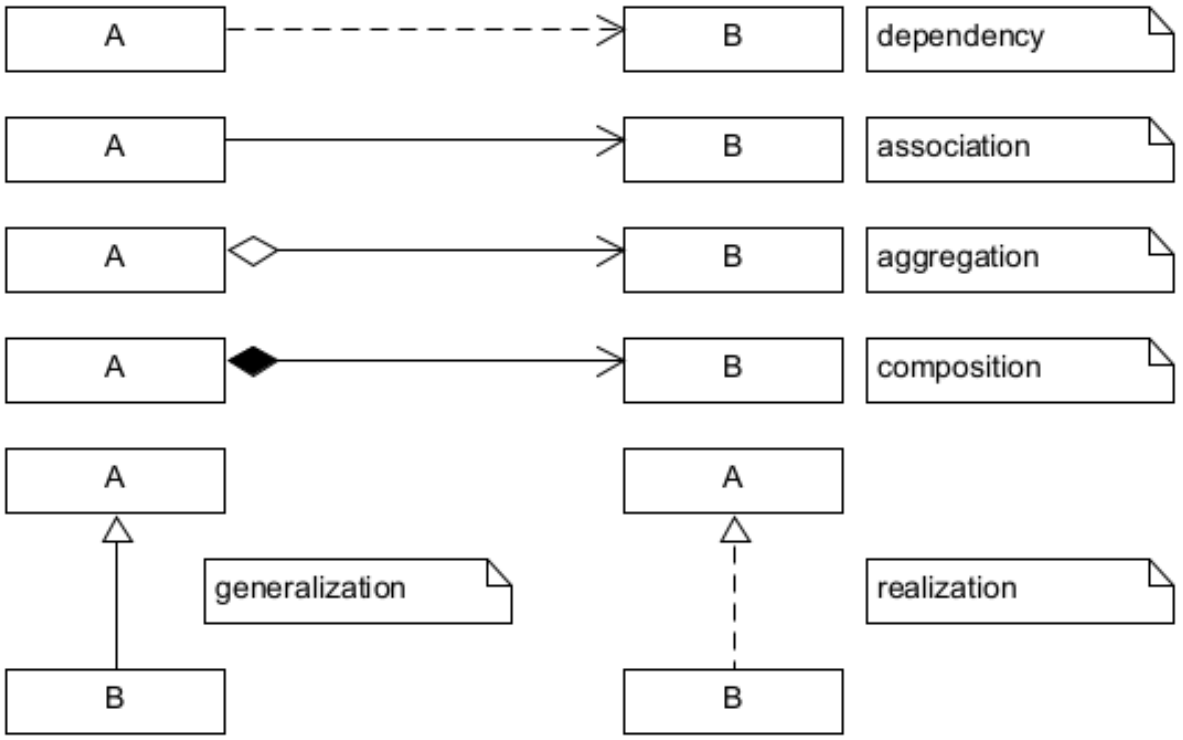
```
class Example
{
private:
    int x;
    void foo(int x){…}

protected:
    int y;
    int bar(int y, int z){…}

public:
    Example(){…}
    int z;
};
```



https://github.com/chgogos/oop/blob/master/uml/example1.cpp

Σχέσεις (relationships)

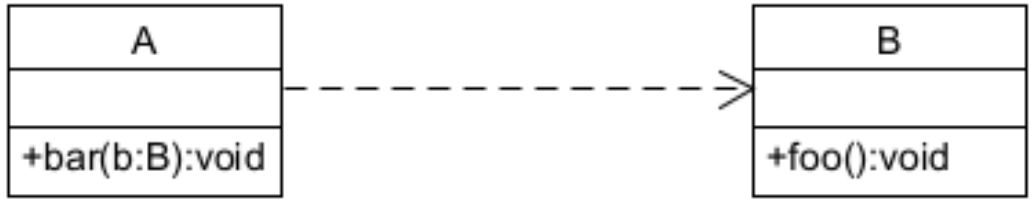
- Τύποι σχέσεων
  - Εξάρτηση (dependency)
  - Συσχέτιση (association)
  - Συνάθροιση (aggregation)
  - Σύνθεση (composition)
  - Γενίκευση (generalization)
  - Υλοποίηση (realization)
- Για κάθε τύπο σχέσης χρησιμοποιείται διαφορετικός οπτικά τύπος σύνδεσης



Σχέση εξάρτησης (dependency)

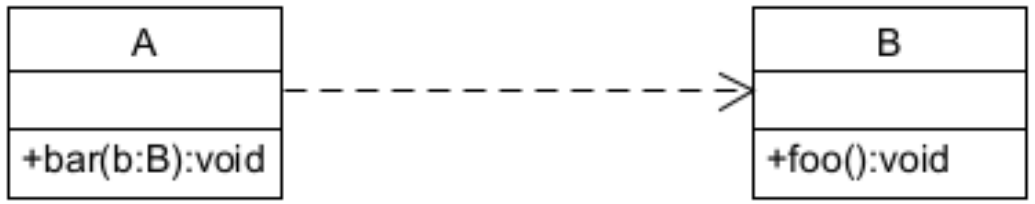
- Το Α έχει εξάρτηση από το Β
  - Αν ένα αντικείμενο της κλάσης Β περνά ως παράμετρος σε συνάρτηση της κλάσης Α

  ή
  - Αν ένα αντικείμενο της κλάσης Β εκχωρείται σε τοπική μεταβλητή μιας συνάρτησης της κλάσης Α
- Η σχέση εξάρτησης είναι γνωστή και ως σχέση “uses a” (χρησιμοποιεί ένα)



Παράδειγμα με σχέση εξάρτησης (κώδικας)

```
class B
{
public:
    void foo(){}
};

class A
{
public:
    void bar(B b){b.foo();}
};

int main()
{
    B obj1; A obj2;
    obj2.bar(obj1);
}
```



https://github.com/chgogos/oop/blob/master/uml/dependency1.cpp

https://github.com/chgogos/oop/blob/master/uml/dependency2.cpp

Σχέση συσχέτισης (association)

- Ένα A έχει πρόσβαση σε ένα B
- Η σχέση συσχέτισης είναι γνωστή και ως σχέση “has-a” (έχει ένα) ή “knows-a” (γνωρίζει ένα)

Παράδειγμα με σχέση συσχέτισης (κώδικας)

```
class B
{
public:
    void foo(){}
};
class A
{
private:
    B *b_ptr;
public:
    A(B *b) : b_ptr(b){}
    void bar(){ b_ptr->foo();}
};
int main()
{
    B obj1; A obj2(&obj1);
    obj2.bar();
}
```

https://github.com/chgogos/oop/blob/master/uml/association1.cpp

https://github.com/chgogos/oop/blob/master/uml/association2.cpp

Σχέση συνάθροισης (aggregation)

- Το Α έχει ως μέρος του το Β, αν και η διάρκεια ζωής τους μπορεί να διαφοροποιείται καθώς το Β μπορεί να μοιράζεται σε πολλά Α
- Η σχέση συνάθροισης είναι παρόμοια με τη σχέση συσχέτισης, **αλλά υποδηλώνει σαφώς** ότι το Α δεν έχει έλεγχο πάνω στη διάρκεια ζωής του Β
- Το σύμβολο της συνάθροισης (ο ρόμβος) τοποθετείται από την πλευρά της κλάσης που αναπαριστά το **όλο** στη σχέση **όλο-μέρος**

Παράδειγμα με σχέση συνάθροισης (κώδικας)

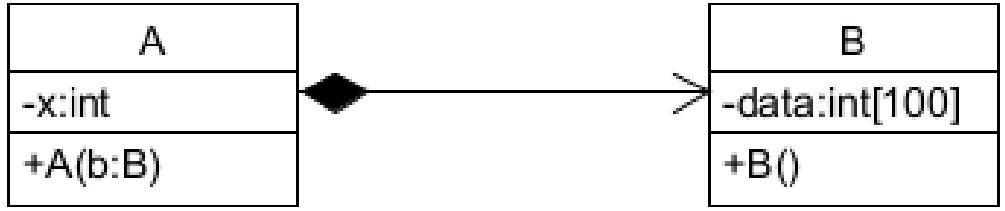
```
class B
{
private:
    int data[100];
public:
    B(){}
};
class A
{
private:
    int x;
    B &b; // aggregation (weak containment)
public:
    A(B &rb) : b(rb){}
};
int main() {
    B b_obj; A obj1(b_obj); A obj2(b_obj);
}
```

https://github.com/chgogos/oop/blob/master/uml/aggregation1.cpp

https://github.com/chgogos/oop/blob/master/uml/aggregation2.cpp

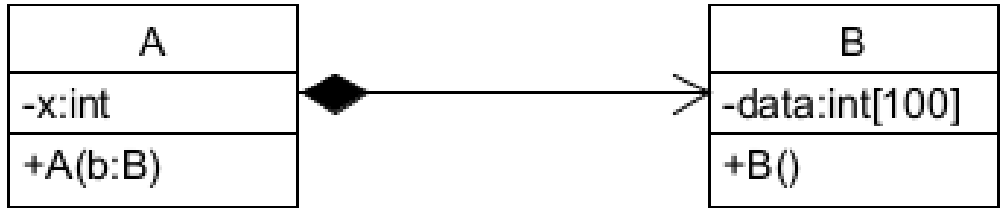
Σχέση σύνθεσης (composition)

- Το Α έχει ως μέρος του το Β, και η διάρκεια ζωής του Β ελέγχεται από το Α (καθώς το Β περιέχεται στο Α)
- Αν καταστραφεί το Α τότε καταστρέφεται και το Β
- Το σύμβολο της σύνθεσης (ο μαύρος ρόμβος) τοποθετείται από την πλευρά της κλάσης που αναπαριστά το **όλο** στη σχέση **όλο-μέρος**



Παράδειγμα με σχέση σύνθεσης (κώδικας)

```
class B
{
private:
    int data[100];
public:
    B(){}
};
class A
{
private:
    int x;
    B b; // composition (strong containment)
public:
    A(){}
};
int main()
{
    A obj1, obj2;
}
```

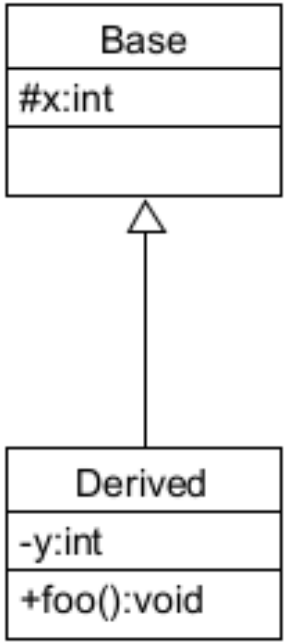


https://github.com/chgogos/oop/blob/master/uml/composition1.cpp

https://github.com/chgogos/oop/blob/master/uml/composition2.cpp

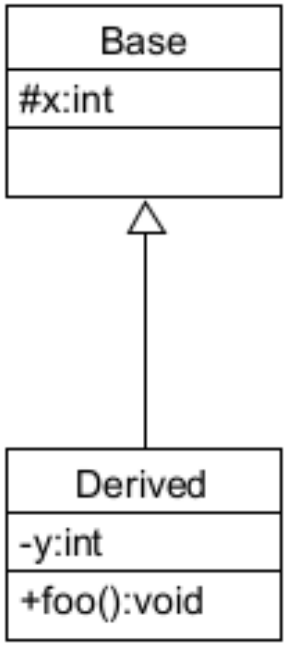
Σχέση γενίκευσης (generalization)

- Η κλάση Base κληρονομεί στην κλάση Derived
- Η κλάση Derived κληρονομεί από την κλάση Base
- Η κλάση Base αποτελεί μια γενίκευση της κλάσης Derived
- Η κλάση Derived αποτελεί μια εξειδίκευση (specialization) της κλάσης Base
- Τα αντικείμενα της κλάσης Derived θεωρούνται αντικείμενα και της κλάσης Base
- Τα αντικείμενα της κλάσης Base δεν θεωρούνται αντικείμενα της κλάσης Derived
- Η σχέση γενίκευσης είναι γνωστή ως σχέση “IS-A”



Παράδειγμα με σχέση γενίκευσης (κώδικας)

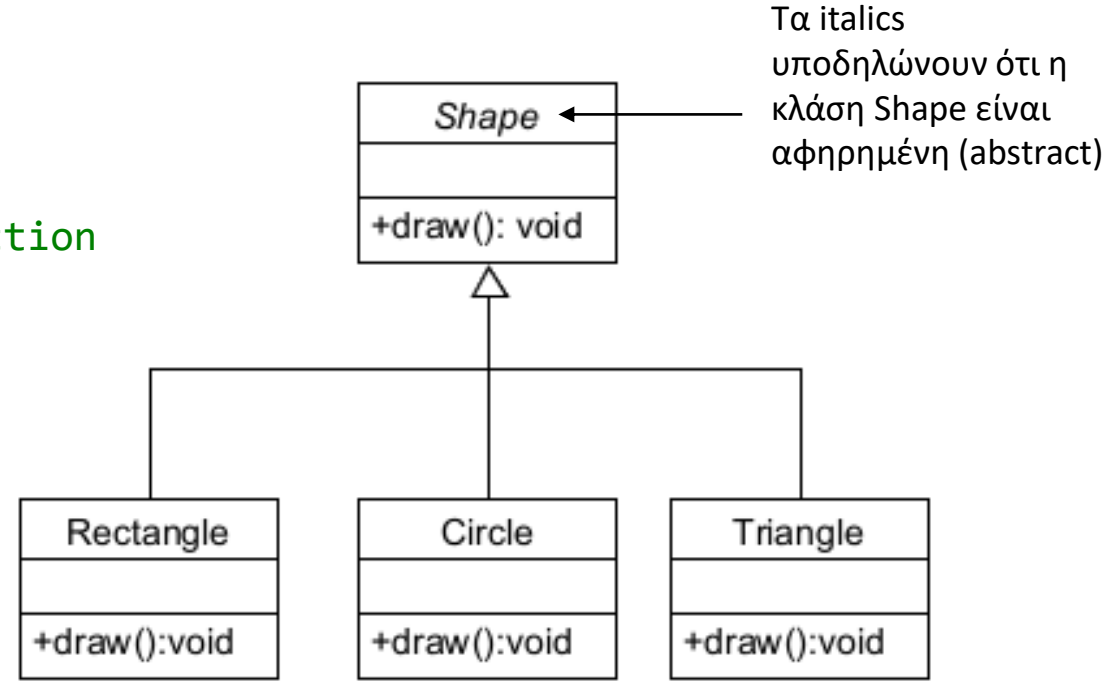
```
class Base
{
protected:
    int x;
};

class Derived : public Base
{
private:
    int y;
public:
    void foo() {…}
};

int main()
{
    Derived obj;
    obj.foo();
}
```



https://github.com/chgogos/oop/blob/master/uml/generalization1.cpp

Γενίκευση με αφηρημένη κλάση (κώδικας)

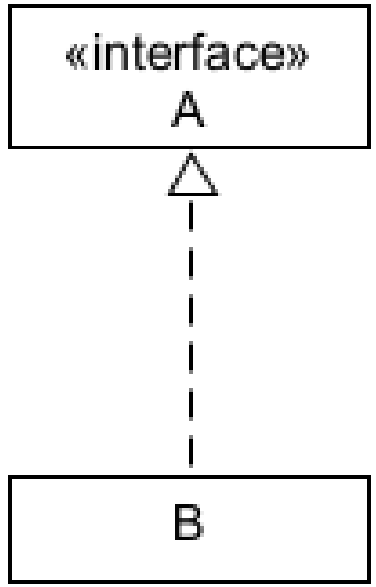
```
class Shape
{
  public:
    virtual ~Shape() {}
    virtual void draw() = 0; // pure virtual function
};
class Rectangle : public Shape
{
  public:
    void draw(){…}
};
class Triangle : public Shape
{…};
class Circle : public Shape
{…};
int main()
{
    Shape *a[] = {new Rectangle, new Triangle, new Circle};
    for (int i = 0; i < 3; i++)  a[i]->draw();
    for (int i = 0; i < 3; i++)  delete a[i];
}
```



https://github.com/chgogos/oop/blob/master/cpp_playground/ex022/shapes.cpp

Σχέση υλοποίησης (realization)

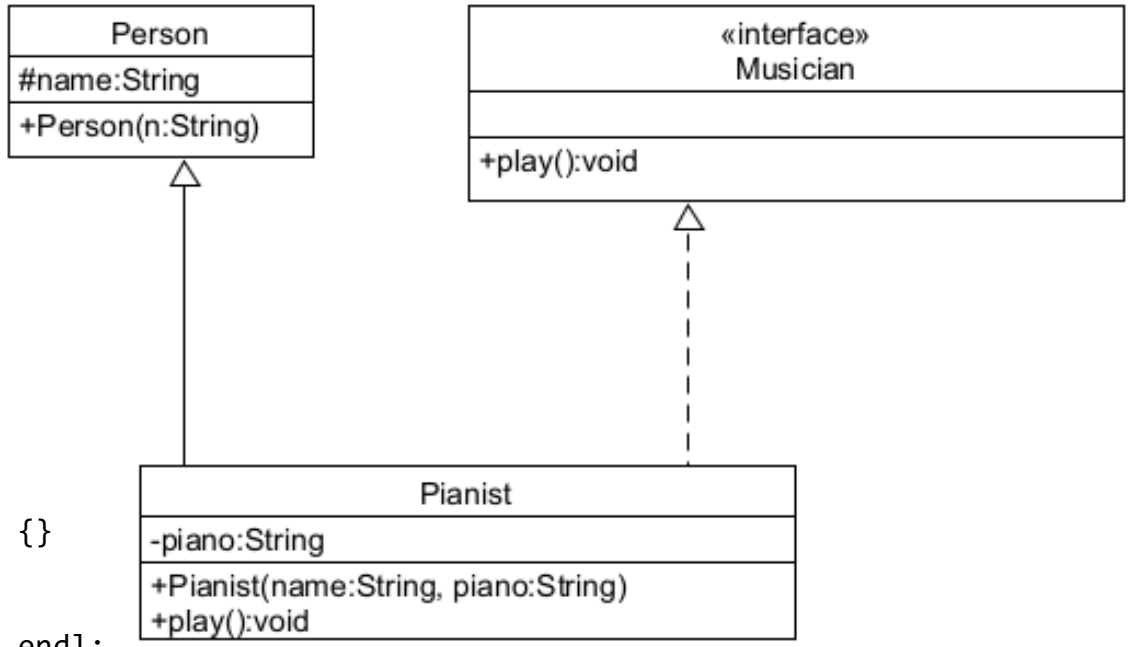
- Η κλάση Β υλοποιεί τη διεπαφή που ορίζεται στο Α
- Ως διεπαφή ορίζεται ένα σύνολο από λειτουργίες
- Χρησιμοποιείται όταν θέλουμε να δείξουμε ότι μια κλάση υλοποιεί λειτουργίες που ορίζονται σε μια διεπαφή (συχνά χρησιμοποιείται η σήμανση «interface»)



Παράδειγμα με σχέση υλοποίησης (κώδικας)

```
class Person
{
protected:
    string name;
public:
    Person(string n) : name(n) {}
};
class Musician
{
public:
    virtual void play() = 0;
};
class Pianist : public Person, public Musician
{
private:
    string piano;
public:
    Pianist(string name, string piano) : Person(name), piano(piano) {}
    void play()
    {
        cout << "Musician: " << name << " Instrument: " << piano << endl;
    }
};

int main(){
    Pianist p("Nikolaos", "Steinway & Sons");
    p.play();
}
```

https://github.com/chgogos/oop/blob/master/uml/realization1.cpp

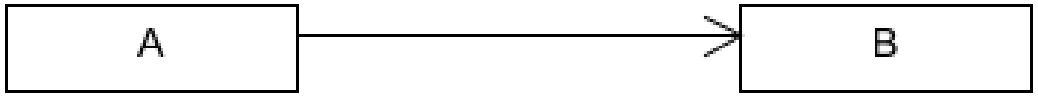


Στη C++ μια διεπαφή ορίζεται ως αφηρημένη κλάση (στο παράδειγμα η κλάση Musician)

Πλοηγησιμότητα συσχετίσεων (navigability)

- Τοποθετώντας βελάκια στα άκρα των συσχετίσεων μπορούμε να διευκρινίσουμε τη δυνατότητα μετάβασης από ένα αντικείμενο μιας κλάσης σε αντικείμενα της συσχετιζόμενης κλάσης

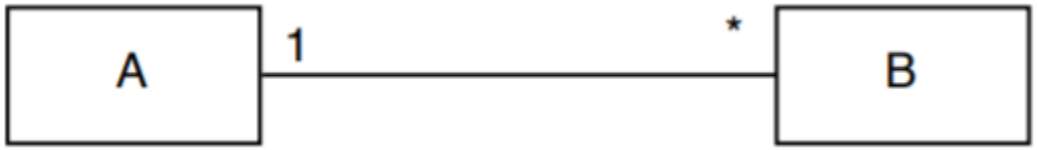
- Το B μπορεί να προσπελαστεί από το A
- Το διάγραμμα δεν υποδηλώνει ότι το A μπορεί να προσπελαστεί από το B



Πολλαπλότητα συσχετίσεων (multiplicity)

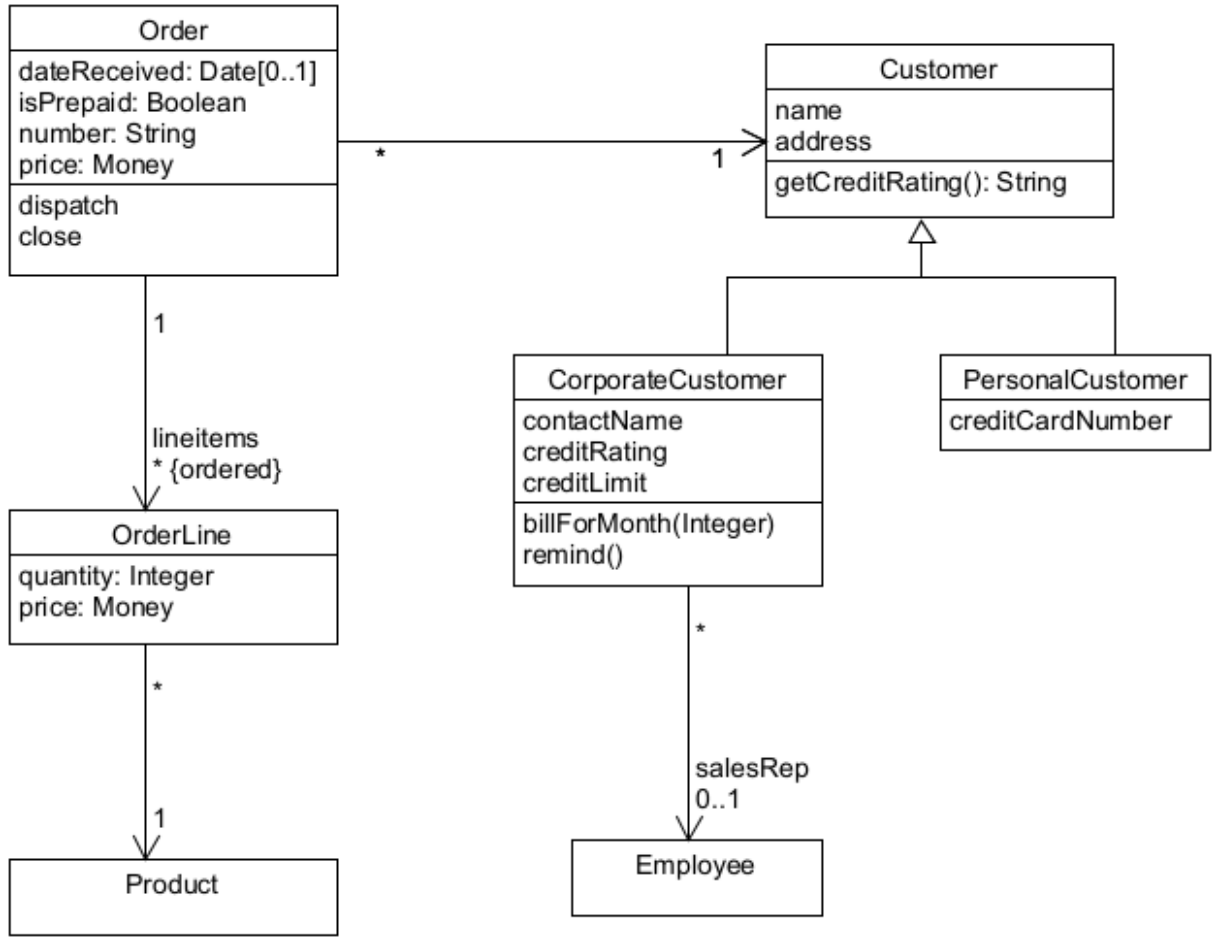
- * (οποιοσδήποτε αριθμός)
- 1 (ακριβώς ένα)
- n (ακριβώς n)
- 0..1 (μηδέν ή ένα)
- 1..* (ένα ή περισσότερα)
- n..m (από n μέχρι m)

- Κάθε αντικείμενο Α σχετίζεται με οποιοδήποτε αριθμό αντικειμένων Β
- Κάθε αντικείμενο Β σχετίζεται με ένα αντικείμενο Α



Παράδειγμα διαγράμματος κλάσεων

- Παρατηρείστε το διαφορετικό επίπεδο συμπλήρωσης λεπτομέρειας στα επιμέρους συστατικά του διαγράμματος (είναι φυσιολογικό)
- Εξετάζοντας τις συσχετίσεις προς την κατεύθυνση της πλοηγησιμότητας:
  - Μια παραγγελία αφορά έναν πελάτη
  - Ο πελάτης μπορεί να είναι εταιρικός ή απλός πελάτης
  - Σε περίπτωση που είναι εταιρικός πελάτης μπορεί να απευθύνεται σε κανένα ή ένα εμπορικό αντιπρόσωπο
  - Κάθε παραγγελία έχει καμία, μια ή περισσότερες γραμμές παραγγελίας
  - Κάθε γραμμή παραγγελίας αφορά ένα προϊόν
- Επιπλέον:
  - Ένας εμπορικός αντιπρόσωπος μπορεί να έχει πολλούς εταιρικούς πελάτες
  - Ένας πελάτης μπορεί να έχει πραγματοποιήσει καμία, μια ή περισσότερες παραγγελίες
  - Μια γραμμή παραγγελίας ανήκει σε μια παραγγελία
  - Ένα προϊόν μπορεί να έχει πολλές γραμμές παραγγελίας που το αφορούν



Αναφορές

- UML Distilled A Brief Guide to the Standard Object Modeling Language by Martin Fowler 2003
- http://www.cs.bsu.edu/homepages/pvgestwicki/misc/uml/