

Η ΓΛΩΣΣΑ C++

Μάθημα 10:

Κληρονομικότητα

Δημήτρης Ψούνης



www.psounis.gr



Περιεχόμενα Μαθήματος

A. Θεωρία

1. Το νόημα της κληρονομικότητας

1. Τι είναι
2. Τι κάνουμε: Αναπαράσταση σχέσης «είναι» (is-a)
3. Πως σκεφτόμαστε

2. Δημόσια Κληρονομικότητα

1. Ορισμός και Πρόσβαση
2. Παράδειγμα
3. Προστατευμένα (protected) χαρακτηριστικά

3. Κατασκευαστές – Καταστροφείς

1. Κατασκευαστές
2. Καταστροφείς

4. Παρατηρήσεις

1. “Είναι” και “Έχει”
2. Οντολογίες και Διαγράμματα Κλάσεων
3. Ιεραρχία κλάσεων
4. Περαιτέρω Θέματα στην Κληρονομικότητα

B. Ασκήσεις

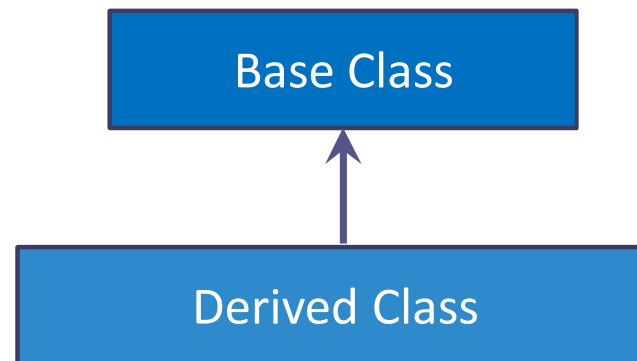


A. Θεωρία

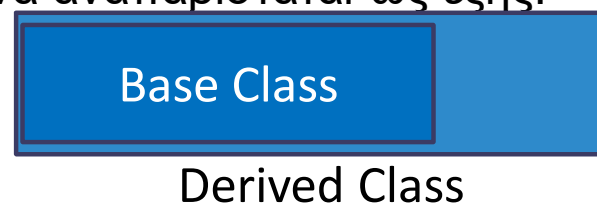
1. Το νόημα της κληρονομικότητας

1. Τι είναι

- Η κληρονομικότητα (inheritance):
 - Παίρνει μία υφιστάμενη κλάση
 - την επαυξάνει σε μια καινούργια, προσθέτοντας σε αυτήν χαρακτηριστικά (μέλη – μεθόδους).
- Σχηματικά συνηθίζεται να αναπαρίσταται ως εξής:



- Η παραγόμενη κλάση (derived class) περιέχει όλα τα μέλη και τις μεθόδους της βασικής κλάσης
 - και λέμε ότι κληρονομεί (inherits) τη βασική κλάση (base class)
- Ίσως θα ήταν καλύτερα να αναπαρίσταται ως εξής:



- Για να φαίνεται ότι επαυξάνει (επεκτείνει) τη βασική κλάση.

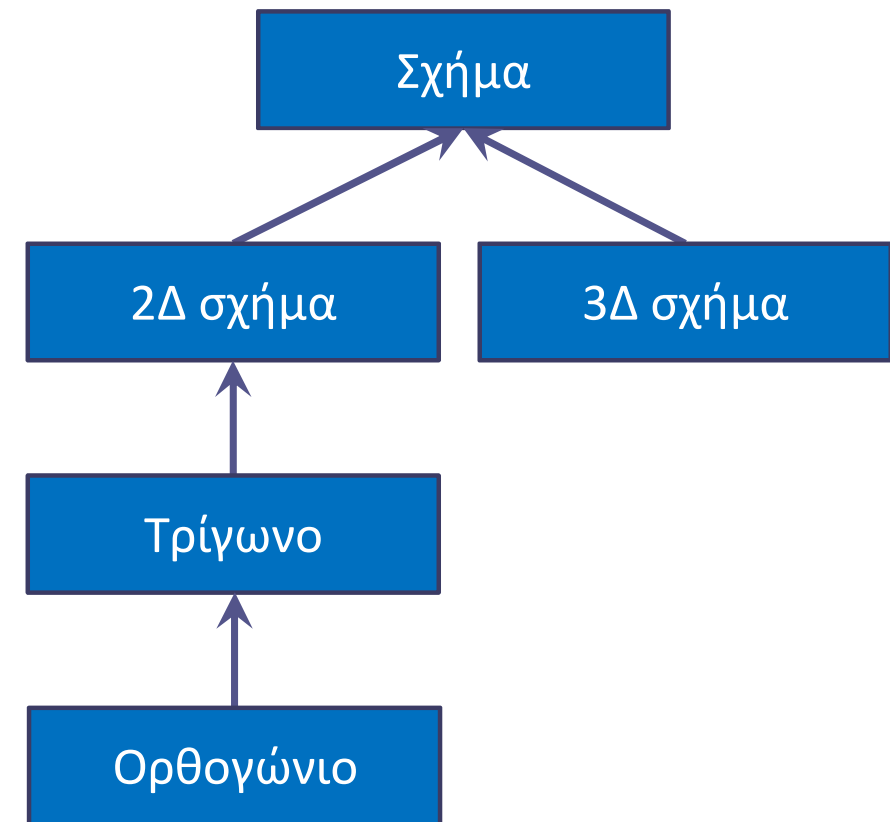
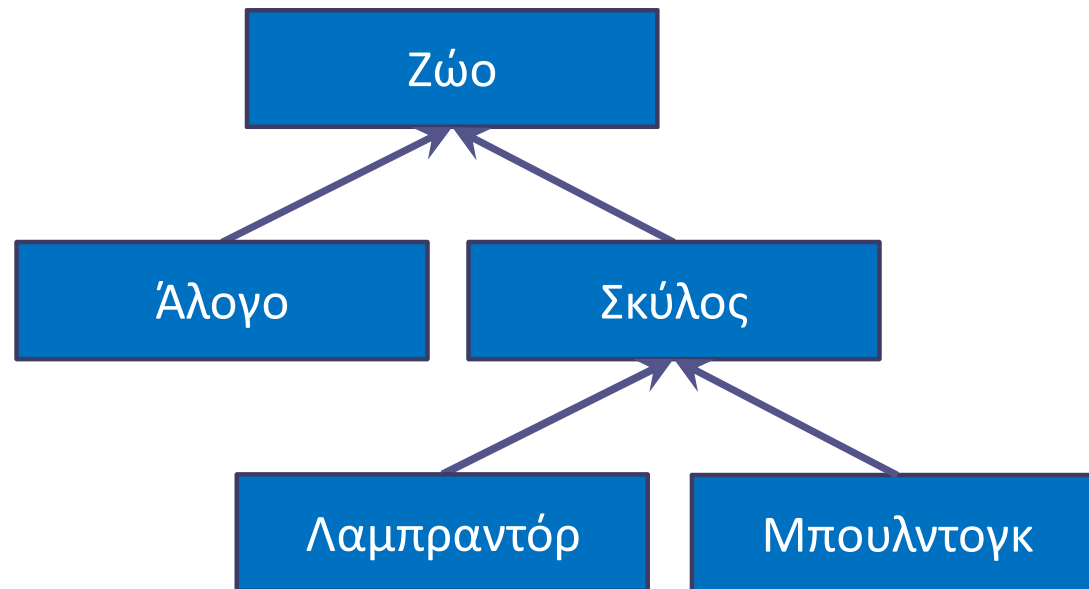


Α. Θεωρία

1. Το νόημα της κληρονομικότητας

2. Τι κάνουμε: Αναπαράσταση σχέσης «είναι» (is-a)

- Η βασική σχέση που αναπαρίσταται με την κληρονομικότητα είναι η «**είναι-ένα**» (is-a)
- Παραδείγματα οντολογιών (κλάσεις και υπο-κλάσεις):



Παρατήρηση:

- Οι σχέσεις «είναι» (is-a) και «εχει» (has-a: Αναπαρίσταται με αντικείμενο που περιέχεται σε κλάση) είναι τα βασικά εργαλεία για την αναπαράσταση καταστάσεων στη C++



A. Θεωρία

1. Το νόημα της κληρονομικότητας

3. Πως σκεφτόμαστε

- Για την οργάνωση μιας οντολογίας κάνουμε κάποιες ερωτήσεις.
- Π.χ. στο παράδειγμα με τα ζώα:
 - Ποια είναι τα κοινά χαρακτηριστικά των αλόγων και των σκύλων;
 - Έχουν βάρος και ύψος (μέλη)
 - Τρώνε, τρέχουν (μέθοδοι)
 - Αυτά θα οριστούν στην βασική κλάση και θα κληρονομούνται από κάθε άλογο και σκύλο.
 - Τι κάνει έναν σκύλο διαφορετικό από ένα άλογο; Τι τον συγκεκριμενοποιεί σε αντίθεση με άλλα ζώα;
 - Γαβγίζει
 - Φυλάει το σπίτι
 - Αυτά θα οριστούν στην παραγόμενη κλάση «σκύλος»
- Ένα αντικείμενο της κλάσης «Σκύλος» θα περιέχει όλα τα χαρακτηριστικά της κλάσης «Ζώο» καθώς και τα καινούργια χαρακτηριστικά που θα ορίσει.

Παρατήρηση:

- Η εμπειρία στην μοντελοποίηση είναι αντικείμενο της ανάλυσης συστημάτων και υπάρχουν ειδικές γλώσσες που περιγράφουν προδιαγραφές, όπως π.χ. η UML



A. Θεωρία

2. Δημόσια Κληρονομικότητα

1. Ορισμός και Πρόσβαση

- Ορίζουμε ότι η κλάση D κληρονομεί την κλάση B με δημόσιο τρόπο ως εξής:

```
class D: public B {  
    ...  
};
```

- Τα χαρακτηριστικά της κλάσης B μεταφέρονται στην κλάση D.
- Συγκεκριμένα:
 - Τα δημόσια μέλη της κλάσης B είναι δημόσια και στην κλάση D
 - Τα ιδιωτικά μέλη της κλάσης B, υπάρχουν στην κλάση D, αλλά δεν είναι προσβάσιμα από τις νέες μεθόδους της B

Σημείωση:

- Εκτός από τη δημόσια κληρονομικότητα, υπάρχουν η προστατευμένη και η ιδιωτική κληρονομικότητα, που θα μελετήσουμε σε επόμενο μάθημα.
- Ωστόσο η πιο συνηθισμένη κληρονομικότητα, είναι η δημόσια κληρονομικότητα.



A. Θεωρία

2. Δημόσια Κληρονομικότητα

2. Παράδειγμα

- Βασική κλάση animal

```
class animal {  
    public:  
        int get_weight() const;  
        int get_height() const;  
        void set_weight(int in_weight);  
        void set_height(int in_height);  
    private:  
        int weight;  
        int height;  
};
```

```
void animal::set_weight(int in_weight)  
{  
    weight = in_weight;  
}
```

```
void animal::set_height(int in_height)  
{  
    height = in_height;  
}
```

```
int animal::get_weight() const  
{  
    return weight;  
}
```

```
int animal::get_height() const  
{  
    return height;  
}
```



A. Θεωρία

2. Δημόσια Κληρονομικότητα

2. Παράδειγμα

- Κλάση Σκύλος: Κληρονομεί την animal

```
class dog: public animal {  
public:  
    void bark();  
    void set_house_address(string  
                           in_house_address);  
    string get_house_address() const;  
    friend ostream &operator<<(ostream &left,  
                               const dog &right);  
private:  
    string house_address;  
};
```

```
void dog::bark()  
{  
    cout<<"arf arf!"<<endl;  
}  
void dog::set_house_address(string in_house_address)  
{  
    house_address = in_house_address;  
}  
string dog::get_house_address() const  
{  
    return house_address;  
}  
ostream &operator<<(ostream &left, const dog &right)  
{  
    //left<<"weight: "<<weight; //doesn't work  
    left<<"weight: "<<right.get_weight()<<endl;  
    //left<<"height: "<<height; //doesn't work  
    left<<"height: "<<right.get_height()<<endl;  
    left<<"address: "<<right.house_address;  
}
```




A. Θεωρία

2. Δημόσια Κληρονομικότητα

2. Παράδειγμα

- Συνάρτηση main

```
int main()
{
    dog piko;

    piko.set_house_address("Left Hill 154");
    piko.set_weight(10);
    piko.set_height(35);

    piko.bark();
    cout<<piko;

    return 0;
}
```



A. Θεωρία

2. Δημόσια Κληρονομικότητα

3. Προστατευμένα (protected) Χαρακτηριστικά

- Ως τώρα ξέρουμε ότι μία κλάση έχει:
 - Δημόσια (public) μέλη - μεθόδους
 - Ιδιωτικά (private) μέλη – μεθόδους
- Και προσθέτουμε τα προστατευμένα (protected) μέλη – μεθόδους:

Ένα **προστατευμένο (protected)** χαρακτηριστικό μιας κλάσης:

- Συμπεριφέρεται ως ιδιωτικό (private) χαρακτηριστικό στην κλάση.
- Αλλά παραμένει προστατευμένο (protected) στις παραγόμενες κλάσεις.

- Έτσι σε αντίθεση με τα private χαρακτηριστικά:
 - Ένα protected χαρακτηριστικό θα είναι προσβάσιμο από τις μεθόδους της παραγόμενης κλάσης.



A. Θεωρία

2. Δημόσια Κληρονομικότητα

3. Προστατευμένα (protected) Χαρακτηριστικά

- Συνεπώς:
 - Έστω μία βασική κλάση με χαρακτηριστικά (μέλη-μεθόδους) είτε private είτε public είτε private.
 - Τα αντικείμενα βασικής κλάσης έχουν πρόσβαση σε όλα τα χαρακτηριστικά
 - Τα αντικείμενα παραγόμενης κλάσης δεν έχουν πρόσβαση στα private χαρακτηριστικά
 - Έξω από την κλάση υπάρχει πρόσβαση μόνο στα public χαρακτηριστικά

Πρόσβαση στη βασική κλάση	public	protected	private
Βασική κλάση	✓	✓	✓
Παραγόμενη κλάση	✓	✓	✗
Έξω από την κλάση	✓	✗	✗



A. Θεωρία

2. Δημόσια Κληρονομικότητα

3. Προστατευμένα Μέλη (protected) Χαρακτηριστικά

- Με το ακόλουθο παράδειγμα (cpp10.protected_members.cpp) αναδεικνύεται η προσβασιμότητα στα μέλη. Αφαιρώντας κάποιο σχόλιο, ο μεταγλωττιστής θα διαμαρτυρηθεί.

```
class B {  
    public:  
        int pub;  
        void f();  
    protected:  
        int pro;  
    private:  
        int pri;  
};  
  
class D: public B {  
    public:  
        void g();  
};
```

```
void B::f()  
{  
    pub=1;  
    pro=1;  
    pri=1;  
}  
  
void D::g()  
{  
    pub=1;  
    pro=1;  
    //pri=1; //no access  
}
```

```
int main()  
{  
    B b;  
    b.pub=1;  
    //b.pro=1; //no access  
    //b.pri=1; //no access  
    D d;  
    d.pub=1;  
    //d.bpro=1; //no access  
    //d.bpri=1; //no access  
}
```



A. Θεωρία

3. Κατασκευαστές – Καταστροφείς

1. Κατασκευαστές

Σειρά εκτέλεσης των **κατασκευαστών** σε ένα αντικείμενο παραγόμενης κλάσης:

- Πρώτα τρέχει ο κατασκευαστής της βασικής κλάσης
 - Έπειτα τρέχει ο κατασκευαστής της παραγόμενης κλάσης
-
- Συνεπώς:
 - Είτε θα έχουμε default κατασκευαστές στις δύο κλάσεις
 - Είτε θα χρησιμοποιήσουμε τις λίστες αρχικοποίησης στον κατασκευαστή της παραγόμενης κλάσης
 - Και θα διοχετεύει πρώτα τα ορίσματα που αφορούν τον κατασκευαστή της βασικής κλάσης μέσω αυτών.
 - Παρατηρήστε στο σχέδιο:
 - Ο constructor της βασικής κλάσης παίρνει π.χ. δύο ακέραια ορίσματα
 - Ο constructor της παραγόμενης κλάσης αρχικοποιεί πρώτα το βασικό αντικείμενο, και έπειτα ασχολείται με ενέργειες που αφορούν το ίδιο το αντικείμενο.

```
B::B(int Bx, int By)
{
    ....
}
```

```
D::D(int Bx, int By, int Dz, int Dw):
    B(Bx,By)
{
    ....
}
```



A. Θεωρία

3. Κατασκευαστές – Καταστροφείς

1. Κατασκευαστές

- Επεκτείνουμε το προηγούμενο παράδειγμα ώστε να χρησιμοποιεί κατασκευαστές:

```
class animal {
public:
    animal(int in_weight, int in_height);
    ...
};

class dog: public animal {
public:
    dog(int in_weight, int in_height, string
        in_house_address);
    ...
};

int main()
{
    dog piko(10,35,"Left Hill 154");
    piko.bark();
    cout<<piko;
    return 0;
}
```

```
animal::animal(int in_weight, int in_height)
{
    cout<<"Constructing Animal..."<<endl;
    weight = in_weight;
    height = in_height;
}

...

dog::dog(int in_weight, int in_height, string in_house_address):
    animal(in_weight, in_height)
{
    cout<<"Constructing Dog..."<<endl;
    house_address=in_house_address;
}

...
```

Ολοκληρωμένο το πρόγραμμα είναι το:
«cpp10.inheritance_constructors.cpp»



A. Θεωρία

3. Κατασκευαστές – Καταστροφείς

2. Καταστροφείς

Σειρά εκτέλεσης των **καταστροφέων** σε ένα αντικείμενο παραγόμενης κλάσης:

- Πρώτα τρέχει ο καταστροφέας της παραγόμενης κλάσης
 - Έπειτα τρέχει ο καταστροφέας της βασικής κλάσης
-
- Η σειρά εκτέλεσης των καταστροφέων είναι και πάλι αντίστροφη από τη σειρά εκτέλεσης των κατασκευαστών



A. Θεωρία

3. Κατασκευαστές – Καταστροφείς

2. Καταστροφείς

- Επεκτείνουμε το προηγούμενο παράδειγμα ώστε να χρησιμοποιεί κατασκευαστές:

```
class animal {
public:
    ~animal();
    ...
};

class dog: public animal {
public:
    ~dog();
    ...
};

int main()
{
    dog piko(10,35,"Left Hill 154");
    piko.bark();
    cout<<piko;
    return 0;
}
```

```
animal::animal(int in_weight, int in_height)
{
    cout<<"Constructing Animal..."<<endl;
    weight = in_weight;
    height = in_height;
}

...

dog::dog(int in_weight, int in_height, string in_house_address):
    animal(in_weight, in_height)
{
    cout<<"Constructing Dog..."<<endl;
    house_address=in_house_address;
}

...
```

Ολοκληρωμένο το πρόγραμμα είναι το:
«cpp10.inheritance_destructors.cpp»



A. Θεωρία

4. Παρατηρήσεις

1. Είναι και Έχει

- Τα βασικά εργαλεία μοντελοποίησης στη C++ είναι η μοντελοποίηση των σχέσεων «είναι» και «έχει»:
 - «Είναι» μεταξύ δύο εννοιών
 - π.χ. Η φεράρι είναι αυτοκίνητο. Ο ποδοσφαιριστής είναι αθλητής. Ο πολιτικός είναι ψεύτης.
 - Μοντελοποιείται με την κληρονομικότητα.
 - Προσοχή! Δεν θα πρέπει να συγχέεται με το «είναι» μεταξύ ενός πραγματικού όντος και μίας έννοιας.
 - Π.χ. Ο Σωκράτης είναι άνθρωπος
 - Εδώ η κλάση είναι ο άνθρωπος και ο Σωκράτης είναι αντικείμενο της κλάσης.
 - «Έχει» μεταξύ δύο εννοιών
 - π.χ. Η καρέκλα έχει 4 πόδια. Μια ομάδα ποδοσφαίρου έχει 11 αθλητές. Η βουλή έχει 300 βουλευτές.
 - Μοντελοποιείται με κλάση που περιέχει ως μέλη αντικείμενα άλλης κλάσης.

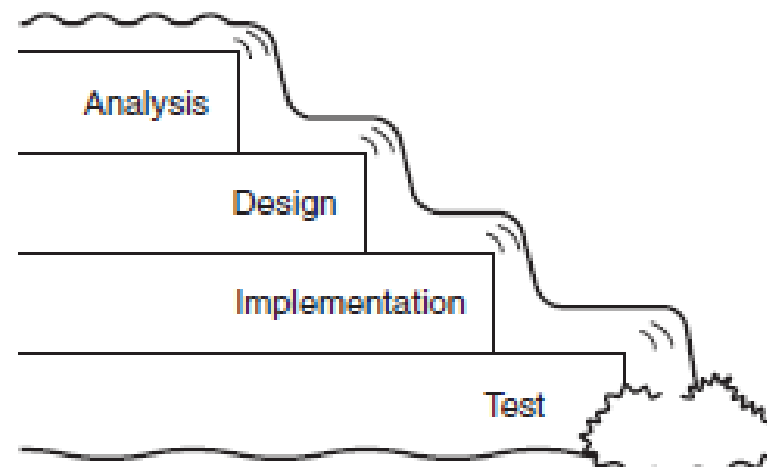


A. Θεωρία

4. Παρατηρήσεις

2. Οντολογίες και Διαγράμματα Κλάσεων

- Αν και ξεφεύγει κάπως από τα όρια της παρουσίασης της σύνταξης της C++ έχουν προταθεί πολλά μοντέλα για την κατασκευή μιας μεγάλης εφαρμογής.
- Ένα δημοφιλές μοντέλο είναι το μοντέλο του καταρράκτη, στο οποίο η ανάλυση και ο σχεδιασμός είναι τα πρώτα πράγματα που γίνονται προτού γραφεί έστω και μία γραμμή κώδικα.



- Η κατασκευή του διαγράμματος κλάσεων είναι μέρος της ανάλυσης και του σχεδιασμού που προηγείται της υλοποίησης σε κάποια γλώσσα προγραμματισμού.

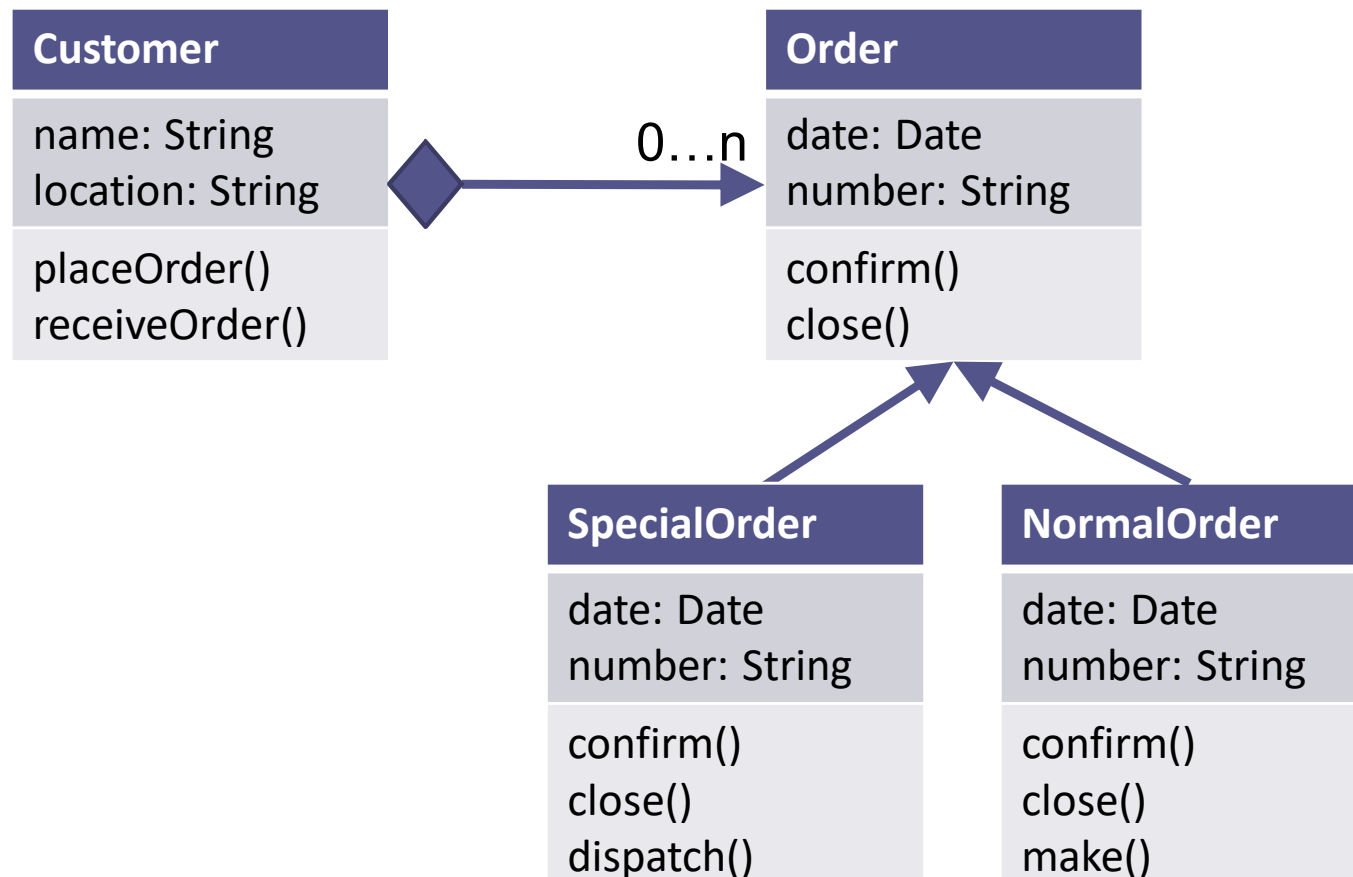


A. Θεωρία

4. Παρατηρήσεις

2. Οντολογίες και Διαγράμματα Κλασεων

- Η ανάλυση συστημάτων και οι γλώσσες προδιαγραφών (π.χ. UML) και, μας λένε ότι η σωστή πολιτική, είναι να σχεδιάζουμε πρώτα ένα διάγραμμα κλάσεων στο οποίο να φαίνονται οι σχέσεις μεταξύ τους:
- π.χ.:



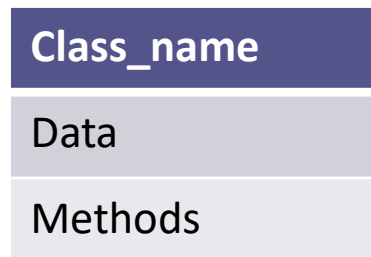


A. Θεωρία

4. Παρατηρήσεις

2. Οντολογίες και Διαγράμματα Κλάσεων

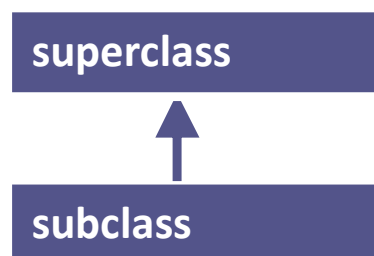
- Τα (πολύ) βασικά στοιχεία του διαγράμματος κλάσεων είναι:
 - Μία κλάση έχει 3 μέρη: Το όνομα, τα μέλη της και τις μεθόδους της και αναπαρίσταται σε ένα ορθογώνιο:



- Η σχέση «έχει» στην οποία υποδεικνύεται και η σχέση πλήθους (στο π.χ. ένας πελάτης έχει πολλές παραγγελίες)



- (Υπονοείται από το συμβολισμό, ότι ο πελάτης θα περιλαμβάνει έναν τρόπο να αποθηκεύει τις παραγγελίες, π.χ. έναν πίνακα)
- Η σχέση «είναι» (κληρονομικότητας) που απεικονίζει τη σχέση κλάσης – υποκλάσης



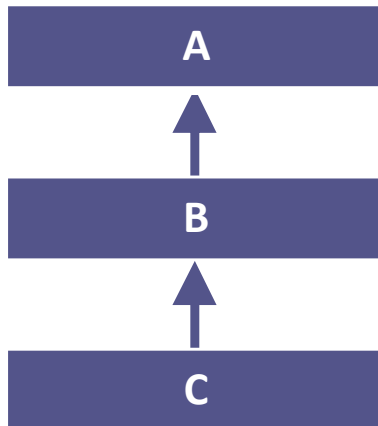


A. Θεωρία

4. Παρατηρήσεις

3. Ιεραρχία κλάσεων

- Με τους ορισμούς που μελετήσαμε, μπορούμε να ορίσουμε ιεραρχίες που εκτείνονται σε παραπάνω από δύο επίπεδα:
- π.χ.



```
class A {  
    ...  
};  
  
class B: public A {  
    ...  
};  
  
class C: public B {  
    ...  
};
```



A. Θεωρία

4. Παρατηρήσεις

4. Περαιτέρω Θέματα στην Κληρονομικότητα

- Η κληρονομικότητα δεν σταματάει εδώ.
- Μία συνηθισμένη λειτουργία που συναντάμε στην κληρονομικότητα είναι:
 - Η υπερκλάση να έχει μία μέθοδο
 - και έπειτα οι υποκλάσεις να αλλάζουν την λειτουργικότητα αυτή
 - (π.χ. στο παράδειγμα που είδαμε η παραγγελία γίνεται π.χ. μέσω ίντερνετ, αλλά η ειδική παραγγελία γίνεται π.χ. μέσω τηλεφώνου)
 - Με αυτό το θέμα:
 - Δηλαδή η υπερκλάση να περιέχει μία μέθοδο, και οι υποκλάσεις να την επανα-ορίζουν θα ασχοληθούμε εκτενώς στα επόμενα μαθήματα.
- Μία ακόμη βασική λειτουργικότητα είναι η κλάση να κληρονομεί από δύο υπερκλάσεις (πολλαπλή κληρονομικότητα)
 - Θα αφιερώσουμε ένα ακόμη μάθημα για αυτήν την έννοια.



B. Ασκήσεις

Άσκηση 1.1: Playlist μουσικής

Κατασκευάζουμε ένα πρόγραμμα που θα παρακολουθεί τα μουσικά μας playlist στο youtube.

- Ένα playlist θα αποτελείται από:
 - Το όνομα του
 - Την περιγραφή του
 - Την χρονική διάρκειά του
 - Τα βίντεο από τα οποία αποτελείται (το πολύ 100). Κάθε βίντεο έχει
 - το όνομα του καλλιτέχνη
 - το όνομα του κομματιού
 - Την χρονική του διάρκεια.
- Ορίστε κατάλληλα:
 - Κατασκευαστή του playlist (να παίρνει όλα τα στοιχεία, και να έχει 0 βίντεο)
 - Προσθήκη βίντεο (να ενημερώνεται κατάλληλα η διάρκεια του playlist)
 - Υπερφόρτωση της εκτύπωσης τόσο του βίντεο όσο και του playlist.
 - και όποια άλλη βοηθητική μέθοδο κρίνετε σκόπιμη.



B. Ασκήσεις

Άσκηση 1.2: Playlist μουσικής (συνέχεια)

Η κλάση `playlist` κληρονομείται από την `classical_playlist`

- Προσθέτει στην λειτουργικότητα ακόμη ένα πεδίο «περίοδος» (`period`) στο οποίο απεικονίζεται η χρονική περίοδος στην οποία ανήκουν τα κομμάτια του `playlist`.
 - π.χ. «μπαρόκ», «ροκοκό» κ.λπ.
- Γράψτε και μία συνάρτηση `main` η οποία θα κατασκευάζει 3 `playlist` `classical_playlist` και έπειτα θα τα τυπώνει στην οθόνη.



B. Ασκήσεις

Άσκηση 2.1: Magic the Gathering

Το Magic the Gathering είναι το δημοφιλέστερο επιτραπέζιο παιχνίδι καρτών.

- Ορίστε μία κλάση card:
 - έχει ένα όνομα
 - έχει ένα χρώμα (blue, green, red, white, black)
 - έχει σπανιότητα (common, uncommon, rare)
 - να έχει κατασκευαστή και υπερφόρτωση της εκτύπωσης.
- Ορίστε μία κλάση creature:
 - είναι μία κάρτα που προσθέτει στην λειτουργικότητα:
 - έναν δείκτη επίθεσης (ακέραιος)
 - έναν δείκτη άμυνας (ακέραιος)
 - είδος (συμβολοσειρά)
 - Ορίστε κατάλληλο κατασκευαστή και υπερφόρτωση της εκτύπωσης.





Β. Ασκήσεις

Άσκηση 2.2: Magic the Gathering

Ορίστε στη main τις ακόλουθες δύο κάρτες:

Είδος



Όνομα



Σπανιότητα

Επίθεση

Άμυνα



B. Ασκήσεις

Άσκηση 2.3: Magic the Gathering

Ένας άλλος τύπος κάρτας είναι η «land»

- Επεκτείνετε την κλάση card με την εξής λειτουργικότητα:
- Έχει μέλος το mana (το οποίο είναι 1 ή 2)
- Έχει μία περιγραφή
- Έχει μια μεταβλητή tap (η οποία δείχνει αν έχουμε χρησιμοποιήσει ή όχι την κάρτα στον συγκεκριμένο γύρο)

Κατασκευάστε στη main σας τις εξής κάρτες:



(όλες οι παραπάνω κάρτες έχουν 1 mana)



B. Ασκήσεις

Άσκηση 2.4: Magic the Gathering

- Σε ένα παιχνίδι, κάθε παίκτης τραβάει τυχαία 7 κάρτες.
- Προβληματίζεστε: Μπορούμε να απεικονίσουμε τις κάρτες που έχει τραβήξει ο παίκτης π.χ. με έναν πίνακα 7 θέσεων; Ποιον τρόπο θα επιλέγατε για να απεικονίσετε το «χέρι» του παίκτη;