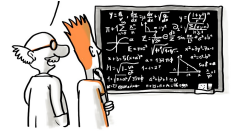


Ενότητα 1: Εισαγωγή στον Α/Σ Προγραμματισμό

Διάλεξη 7 - 13/2/24

UML

Σύνοψη



- **Τι έχουμε μάθει μέχρι τώρα;**
 - Βασικά στοιχεία Α/Σ Π και προγραμματισμού Java. Σχεδιασμός κλάσεων.
- **Τι θα δούμε στη συνέχεια;**
 - UML
 - Σχεδιάγραμμα/πρότυπα λογισμικού (design patterns)

Introduction to UML

- **UML** και **patterns** είναι δύο εργαλεία σχεδιασμού λογισμικού, τα οποία μπορεί να χρησιμοποιηθούν στο πλαίσιο οποιασδήποτε γλώσσας Α/Σ Π.
- Η UML είναι γραφική γλώσσα που χρησιμοποιείται για το **σχεδιασμό** και την **τεκμηρίωση** λογισμικού OOP.
- Ένα μοτίβο (σχεδιάγραμμα, μοτίβο, πρότυπο, **pattern**) είναι ένα είδος **προτύπου** ή **περίγραμμα** μιας εργασίας λογισμικού
 - Ένα μοτίβο μπορεί να υλοποιηθεί ως διαφορετικός κώδικας σε διαφορετικές, αλλά παρόμοιες, εφαρμογές.

Unified Modelling Language - Σύντομη Ανασκόπηση

- **Source:**
 - Nelson Padua-Perez & William Pugh
 - Department of Computer Science
 - University of Maryland, College Park
 - Chapter 12, Savitch

Unified Modelling Language

- Ένα σύνολο από διαγραμματικές συμβάσεις, οι οποίες μπορούν να χρησιμοποιηθούν για την περιγραφή σχεδίων λογισμικού.
- Υπάρχουν διάφοροι τύποι διαγραμμάτων UML
 - Κάθε τύπος χρησιμοποιείται για περιγραφή διαφορετικών θεμάτων
 - Πχ: class diagrams, state diagrams, sequence diagrams

Unified Modelling Language

- **Pseudocode - ψευδοκώδικας:** είναι ένας τρόπος αναπαράστασης ενός προγράμματος με γραμμικό και αλγεβρικό τρόπο
 - Απλοποιεί τον σχεδιασμό εξαλείφοντας τις λεπτομέρειες της σύνταξης της γλώσσας προγραμματισμού
- **Συστήματα γραφικής αναπαράστασης** έχουν επίσης χρησιμοποιηθεί για το σχεδιασμό προγραμμάτων
 - Για παράδειγμα, διαγράμματα ροής και δομικά διαγράμματα
- **Unified Modelling Language (UML):** φορμαλισμός γραφικής αναπαράστασης, σχεδιασμένος για να αντικατοπτρίζει και να χρησιμοποιείται με τη φιλοσοφία του Α/Σ. Π.

Ιστορία της UML

- As OOP has developed, different groups have developed graphical or other representations for OOP design
- In 1996, Brady Booch, Ivar Jacobson, and James Rumbaugh released an early version of UML
 - Its purpose was to produce a standardized graphical representation language for object-oriented design and documentation
- Since then, UML has been developed and revised in response to feedback from the OOP community
 - Today, the UML standard is maintained and certified by the **Object Management Group (OMG)**

Διαγράμματα Κλάσεων UML

- Classes are central to OOP, and the **class diagram** is the easiest of the UML graphical representations to understand and use
- A class diagram is divided up into three sections
 - The top section contains the **class name**
 - The middle section contains the **data specification** for the class
 - The bottom section contains the **actions or methods** of the class

Διαγράμματα Κλάσεων UML

- The **data specification** for each piece of data in a UML diagram consists of its name, followed by a **colon**, followed by its **type**
- Each name is preceded by a character that specifies its access type:
 - A **minus sign (-)** indicates private access
 - A **plus sign (+)** indicates public access
 - A **sharp (#)** indicates protected access
 - A **tilde (~)** indicates package access

Διαγράμματα Κλάσεων UML

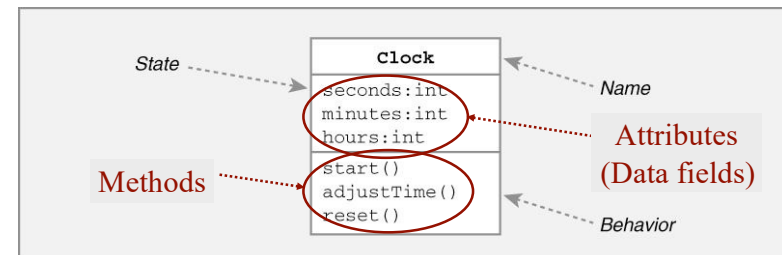
- Each method in a UML diagram is indicated by the name of the method, followed by its parenthesized parameter list, a colon, and its return type
- The access type of each method is indicated in the same way as for data

Διαγράμματα Κλάσεων UML

- A class diagram **need not give a complete description of the class**
 - If a given analysis does not require that all the class members be represented, then those members are not listed in the class diagram
 - Missing members are indicated with an ellipsis (three dots)

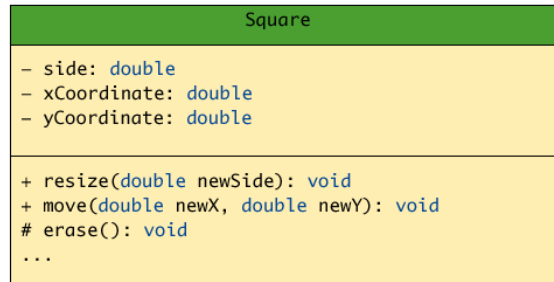
Διαγράμματα Κλάσεων UML

- Represent the (static) structure of the system
- General In Java
 - Name Name
 - State Data Fields / Instance Variables / Class Attributes
 - Behaviour Methods



Διαγράμματα Κλάσεων UML








Display 12.1 A UML Class Diagram



Συσχετίσεις μεταξύ κλάσεων

- Αντί να εμφανίζουν απλώς τη διαπροσωπεία μιας κλάσης, τα διαγράμματα κλάσεων έχουν σχεδιαστεί κυρίως για να δείχνουν τις αλληλεπιδράσεις μεταξύ των κλάσεων.
- Η UML έχει διάφορους τρόπους για να υποδεικνύει τη ροή πληροφοριών από ένα αντικείμενο κλάσης σε άλλο χρησιμοποιώντας διαφορετικά είδη σχολιασμένων βελών (annotated arrows).
- Η UML έχει σχολιασμούς για ομαδοποιήσεις κλάσεων σε πακέτα, για κληρονομικότητα και για άλλες αλληλεπιδράσεις.
- Επιπλέον, η UML είναι επεκτάσιμη.

Συσχετίσεις μεταξύ κλάσεων (Relationships Between Classes)

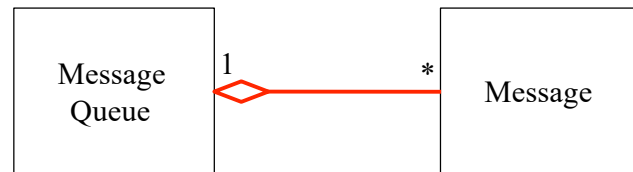
- Dependency - εξάρτηση
 - Temporary, "uses a"
- Aggregation - συσσωμάτωση
- Composition - σύνθεση
- Association - συνεταιρισμός
 - Permanent, structural, "has a"
 - Solid line (arrowhead optional) OR 
- Inheritance - κληρονομικότητα
 - Inheritance, "is a"
 - Solid line with open (triangular) arrowhead
- Implementation - υλοποίηση
 - Dotted line with open (triangular) arrowhead

Πολλαπλότητα Συσχετίσεων (Multiplicity of Associations)

- Κάποιες σχέσεις μπορούν να ποσοτικοποιηθούν
- Η πολλαπλότητα υποδηλώνει με πόσα αντικείμενα μπορεί να συσχετισθεί με έγκυρο τρόπο το αντικείμενο αφετηρίας (Multiplicity denotes how many objects the source object can legitimately reference)
- Σημειογραφία (Notation)
 - * \Rightarrow 0, 1, or more
 - 5 \Rightarrow 5 exactly
 - 5..8 \Rightarrow between 5 and 8, inclusive
 - 5..* \Rightarrow 5 or more

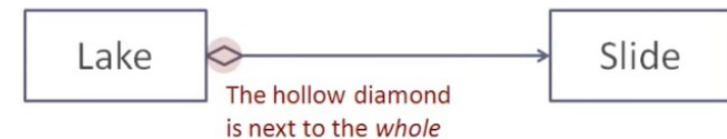
Συσσωμάτωση - Aggregation

- Aggregation (συσσωμάτωση): προκύπτει όταν αντικείμενα μιας κλάσης περιέχουν αντικείμενα κάποιας άλλης κλάσης για μια χρονική περίοδο
 - Ειδική κατηγορία συσχέτισης (“**has-a**”)
 - Ένα πεδίο δεδομένων μιας κλάσης μπορεί να περιγραφεί είτε με συνάθροιση είτε σαν χαρακτηριστικό (attribute) της κλάσης.



Aggregation

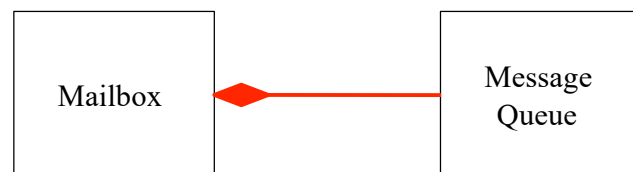
Example:



The slide is part of the lake

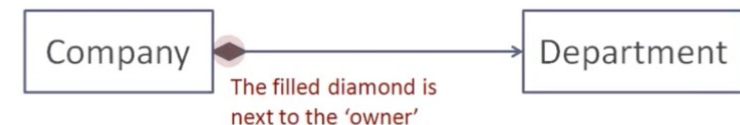
Σύνθεση - Composition

- Πιό ισχυρή μορφή συσσωμάτωσης
 - τα συσσωματούμενα αντικείμενα **δεν έχουν ανεξάρτητη ύπαρξη** έξω από το αντικείμενο που τα συσσωματώνει
- Κάνετε χρήση συσσωμάτωσης ή σύνθεσης μόνο όταν η μια κλάση *διαχειρίζεται* τα αντικείμενα της άλλης



Composition

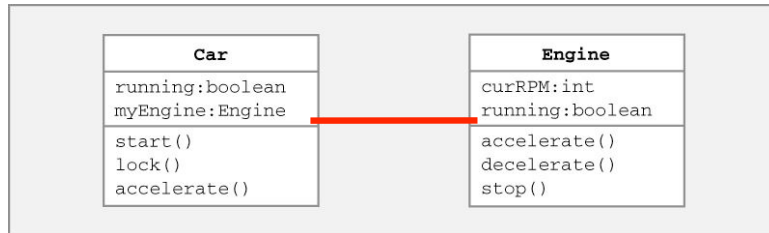
Example:



A department is part of a company. The construction and destruction of a department depends on the company.

Συνεταιρισμός - Association

- Γενικότερη σχέση από την συνάθροιση και την σύνθεση
- **Μόνιμη, δομική** συσχέτιση
- Η κατάσταση της κλάσης A συμπεριλαμβάνει αντικείμενα κλάσης B
- Represented by solid line (arrowhead optional)



Car and Engine classes know about each other

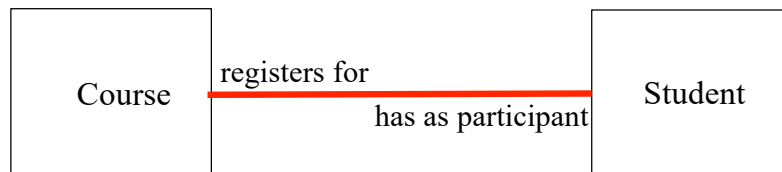
Binary Association

-
-
-
-



ClassC knows about ClassD and
ClassD knows about ClassC

Συνεταιρισμός με ρόλους



For two objects, 'Foo' and 'Bar' the relationships can be defined

305

Association - I have a relationship with an object. 'Foo' uses 'Bar'

```
public class Foo {
    void Baz(Bar bar) {
    }
};
```

Composition - I own an object and I am responsible for its lifetime. When 'Foo' dies, so does 'Bar'

```
public class Foo {
    private Bar bar = new Bar();
}
```

Aggregation - I have an object which I've borrowed from someone else. When 'Foo' dies, 'Bar' may live on.

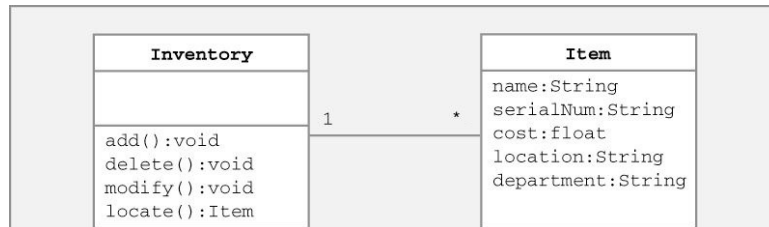
```
public class Foo {
    private Bar bar;
    Foo(Bar bar) {
        this.bar = bar;
    }
}
```

Πολλαπλότητα Συνεταιρισμών (Multiplicity of Associations)

- Many-to-one
 - Bank has many ATMs, ATM knows only 1 bank

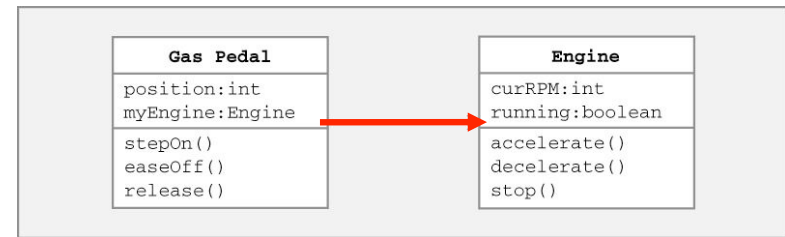


- One-to-many
 - Inventory has many items, items know 1 inventory



Associations w/ Navigation Information

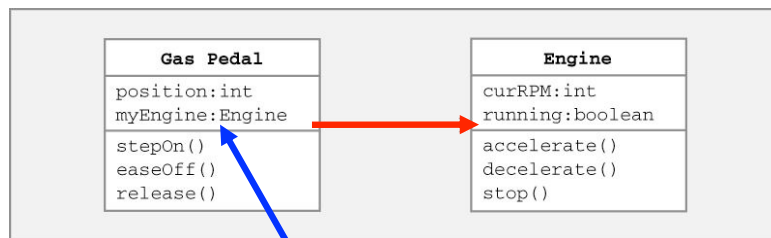
- Υποδηλώνει «κατεύθυνση» στον συνεταιρισμό
- Υποδηλώνει συσχέτιση “has-a” μεταξύ των κλάσεων
 - Represented by solid line with arrowhead



Gas Pedal class knows about Engine class
Engine class doesn't know about Gas Pedal class

Associations w/ Navigation Information

- Denotes “has-a” relationship between classes
- “Gas Pedal” has a “Engine”



State of Gas Pedal class contains instance of Engine class ⇒ can invoke its methods

Εξάρτηση - Dependency

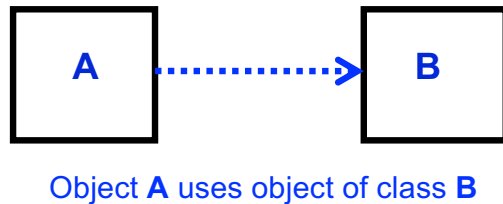
- Υποδηλώνει εξάρτηση (dependence) μεταξύ κλάσεων
- Πάντοτε κατευθυνόμενη (κλάση A εξαρτάται από τη B)
- Αναπαρίσταται με εστιγμένο βέλος



A depends on B

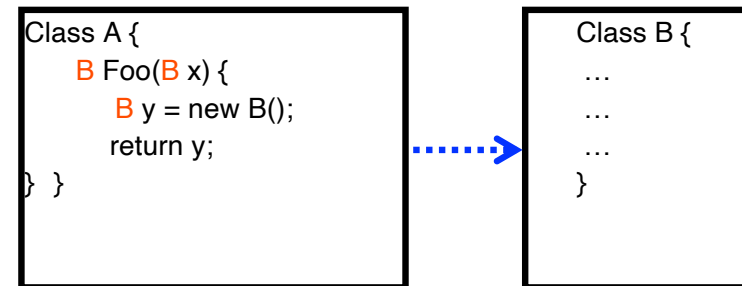
Εξάρτηση - Dependency

- Προκύπτει από μεθόδους της κλάσης
- Μέθοδος κλάσης **A** προσωρινά “**χρησιμοποιεί**” αντικείμενο κλάσης **B**
- Αλλαγή στην κλάση **B** μπορεί να επηρεάσει την **A**

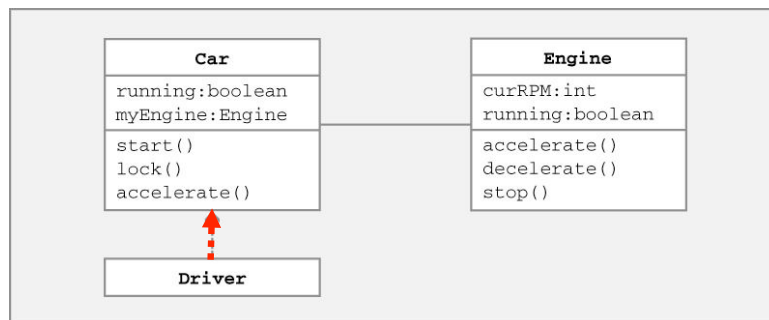


Εξάρτηση - Dependency

- Η εξάρτηση μπορεί να προκληθεί από
 - Τοπική μεταβλητή
 - Τυπική Παράμετρο
 - Επιστρεφόμενη τιμή
- Παράδειγμα:



Εξάρτηση



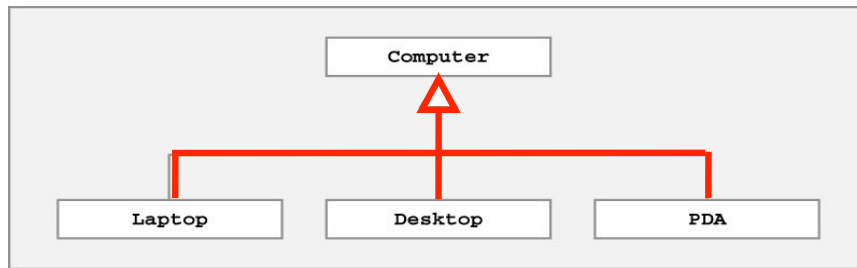
Class Driver depends on Class Car

Διαγράμματα Κληρονομικότητας - Γενίκευσης Inheritance Diagrams

- Ένα διάγραμμα κληρονομικότητας δείχνει τη σχέση μεταξύ μιας **βασικής κλάσης** (κλάση βάσης) και των **παραγόμενων κλάσεων** της (κλάσεις κληρονόμοι)
 - Κανονικά, στο διάγραμμα εμφανίζονται μόνο όσες κλάσεις χρειάζονται (από τις κλάσεις βάσεις ή τους κληρονόμους)
 - Κάθε παραγόμενη κλάση μπορεί να χρησιμεύσει ως η βασική κλάση άλλων παραγόμενων κλάσεων.
- Κάθε βασική κλάση σχεδιάζεται ψηλότερα από τις παραγόμενες κλάσεις της.
 - Ένα βέλος που δείχνει προς τα πάνω σχεδιάζεται ανάμεσά τους για να υποδείξει τη σχέση κληρονομικότητας

Γενίκευση - Generalisation

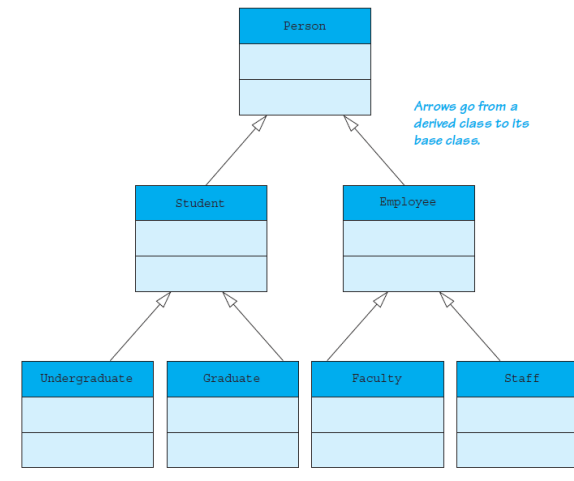
- Denotes **inheritance** (κληρονομικότητα) between classes
- Can view as “**is-a**” relationship
- Represented by line ending in (open) triangle



Laptop, Desktop, PDA inherit state
& behavior from Computers

A Class Hierarchy in UML Notation

Display 12.2 A Class Hierarchy in UML Notation

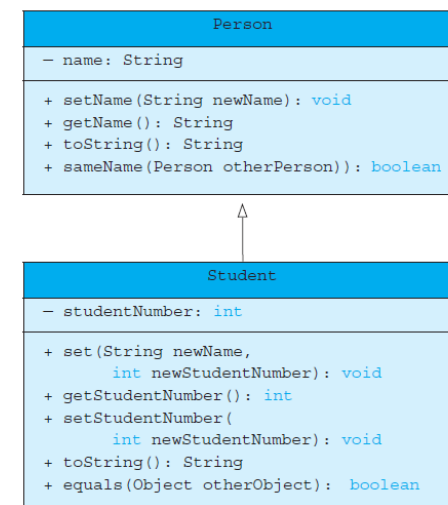


Διαγράμματα Κληρονομικότητας Inheritance Diagrams

- The arrows also help in **locating method definitions**
- To look for a method definition for a class:
 - Examine the class definition first
 - If the method is not found, the path of connecting arrows will show the order and direction in which to search
 - Examine the parent class indicated by the connecting arrow
 - If the method is still not found, then examine this parent's parent class indicated by the connecting arrow
 - Continue until the method is found, or until the top base class is reached

Some Details of a UML Class Hierarchy

Display 12.3 Some Details of a UML Class Hierarchy

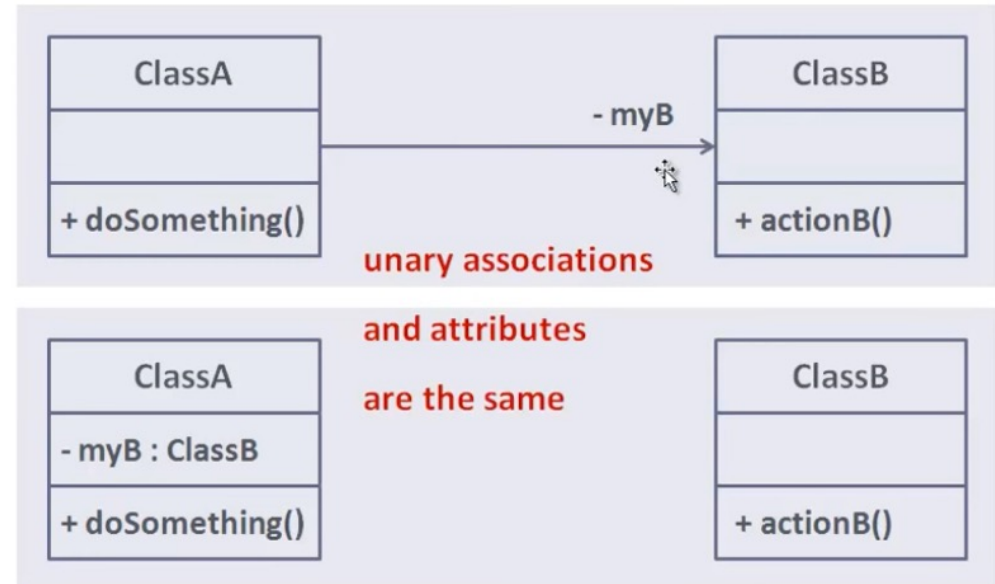


Implementation - υλοποίηση

- Denotes class **implements** Java **interface**
- Represented by dotted line ending in (open) triangle



A implements interface B



Example: Unary Association

