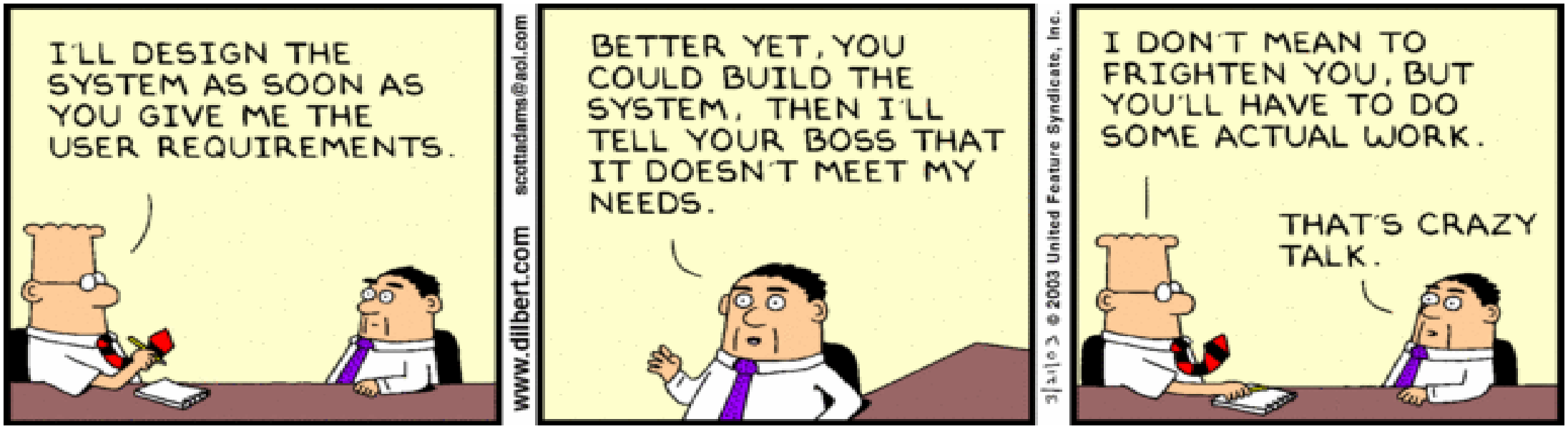
DIS501 Ανάλυση και Σχεδίαση Πληροφοριακών Συστημάτων

Lecture 5: Εισαγωγή στην Αντικειμενοστρεφή προσέγγιση

BY: ΔΡ ΑΥΓΟΥΣΤΑ ΚΥΡΙΑΚΙΔΟΥ ΖΑΧΑΡΟΥΔΙΟΥ

ΜΑΘΗΣΙΑΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

1. Εισαγωγή στην αντικειμενοστραφή ανάλυση και σχεδιασμό
2. Συγκρίνετε την αντικειμενοστρεφή προσέγγιση με άλλες μεθοδολογίες όπως SSAD, αξιολογώντας τα πλεονεκτήματα και τους περιορισμούς της για την ανάπτυξη σύγχρονων πληροφοριακών συστημάτων.
3. Επεξήγηση των αρχών APIE (Abstraction, Polymorphism, Information Hiding, Encapsulation)
4. Εισαγωγή στο UML modelling



ΚΑΘΟΡΙΣΜΟΣ ΠΡΟΔΙΑΓΡΑΦΩΝ ΑΠΑΙΤΗΣΕΩΝ
SPECIFYING REQUIREMENTS

Εισαγωγή στην Αντικειμενοστρεφή ανάλυση και σχεδιασμό (OOAD)

ΠΩΣ ΣΧΕΔΙΑΖΟΥΜΕ/ΜΟΝΤΕΛΟΠΟΙΟΥΜΕ ΤΟ ΠΡΟΙΟΝ?

Συμβατικές απαντήσεις από την SSAD

- Μοντέλο διαδικασίας Process model - το διάγραμμα ροής δεδομένων και το minispec
- Μοντέλο δεδομένων Data model - διαγράμματα οντοτήτων - συσχετίσεων
- Μοντέλο συμπεριφοράς - entity life history

Each model taken separately and with an orderly transition (decomposition)

Πιο μοντέρνες απαντήσεις από το OOAD

- Μοντέλο περίπτωσης χρήσης – Use case diagrams
- Μοντέλο Αντικειμένων (Object model) – Class diagrams
- Δυναμικό Μοντέλο – Object sequence diagrams, State chart diagrams
- Διάγραμμα Κλάσεων Σχεδιασμού - Design Class Diagram
- Περιπτώσεις χρήσης, κλάσεις και αντικείμενα, χαρακτηριστικά, μεταβίβαση μηνυμάτων και συμπεριφορά. Use Cases, Classes and Objects, attributes, message passing and behaviour

  όλα σε ένα ολοκληρωμένο μοντέλο, η μοντελοποίηση γίνεται με επαναληπτικό και σταδιακό τρόπο, υποστηρίζοντας την αφαίρετικότητα (abstraction) και όχι την αποσύνθεση(***decomposition)***

WHAT IS OBJECT ORIENTATION - VIDEO

OOAD VS SSAD

Διαφέρουν στον τρόπο μοντελοποίησης και υλοποίησης του συστήματος (πώς βλέπουμε το σύστημα και πώς το οργανώνουμε**):**

- Structured Systems Analysis & Design **(Σκέψου το σαν μια συνταγή μαγειρικής 📜)**
  - Σχεδιάζουμε το σύστημα βήμα-βήμα, σαν μια συνταγή που ακολουθεί συγκεκριμένες διαδικασίες.
  - Το σύστημα **χωρίζεται** σε διαδικασίες/processes (τι κάνει) και δεδομένα/data (τι αποθηκεύει).
  - Χρησιμοποιεί διαγράμματα ροής για να δείξει πώς μετακινούνται τα δεδομένα και πώς γίνονται οι λειτουργίες (ροή πληροφοριών (data flow).
  - Εξελίχθηκαν από έναν κύκλο ζωής γραμμικών έργων (linear projects) – αποσυνδεδεμένα από το περιβάλλον τους
    - Waterfall model idea at heart
    - a **data-oriented approach** στην μοντελοποίηση
    - Καλύτερο για παραδοσιακά, στατικά συστήματα που έχουν σαφώς ορισμένες διαδικασίες.
      - π.χ database-driven systems, τραπεζικά συστήματα και λογιστικά προγράμματα, large bureaucratic government systems, critical systems

OOAD VS SSAD

Διαφέρουν στον τρόπο μοντελοποίησης και υλοποίησης του συστήματος (πώς βλέπουμε το σύστημα και πώς το οργανώνουμε):

- Object Oriented Analysis & Design (Σκέψου το σαν LEGO 🧱):
  - βλέπει το σύστημα ως μια συλλογή αλληλοεπιδρώντων αντικειμένων - **objects** που έχουν τα δικά τους χαρακτηριστικά-δεδομένα (attributes) και τη δική τους λειτουργία/συμπεριφορά (methods)
  - Δίνει έμφαση στο **encapsulation (βάζει πίσω μαζί τα data και το behaviour/processes)** και στην **επαναχρησιμοποίηση - reusability**
    - **Ιδανικό για σύγχρονες εφαρμογές όπως παιχνίδια, εφαρμογές κινητών, τεχνητή νοημοσύνη.**

OOAD VS SSAD

	SSAD (Δομημένη Ανάλυση & Σχεδίαση Συστημάτων)	OOAD (Αντικειμενοστραφής Ανάλυση & Σχεδίαση)
Βασική Ιδέα	Βασίζεται στη **διαίρεση του συστήματος** σε διαδικασίες και ροές δεδομένων.	Βασίζεται στη **μοντελοποίηση του συστήματος** ως σύνολο αντικειμένων που αλληλεπιδρούν.
Τρόπος Ανάλυσης	Χωρίζει το σύστημα σε **λειτουργίες και διαδικασίες** (process-driven).	Χωρίζει το σύστημα σε **αντικείμενα με δεδομένα και συμπεριφορές** (object-driven).
Δομή του Συστήματος	Το σύστημα περιγράφεται με **Δομές Δεδομένων, Διαγράμματα Ροής Δεδομένων (DFDs) και Καταστάσεις**.	Το σύστημα περιγράφεται με **κλάσεις, αντικείμενα και σχέσεις μεταξύ τους**.
Κύριες Τεχνικές	Χρήση **Διαγραμμάτων Ροής Δεδομένων (DFD)**, **Δομές Απόφασης**, και **ER Διαγράμματα**.	Χρήση **Διαγραμμάτων UML**, όπως **Use Case Diagrams, Class Diagrams, Sequence Diagrams**.
Εστίαση	Εστιάζει περισσότερο στο **ΤΙ κάνει το σύστημα** (λειτουργικότητα).	Εστιάζει στο **ΠΩΣ τα αντικείμενα αλληλεπιδρούν και εξελίσσονται**.
Επαναχρησιμοποίηση Κώδικα	Χαμηλή επαναχρησιμοποίηση (ο κώδικας γράφεται ξανά για νέα έργα).	Υψηλή επαναχρησιμοποίηση μέσω **κληρονομικότητας (inheritance) και πολυμορφισμού (polymorphism)**.
Καταλληλότητα	Καλύτερο για **παραδοσιακά, στατικά συστήματα** που έχουν σαφώς ορισμένες διαδικασίες.	Ιδανικό για **πολύπλοκα, δυναμικά συστήματα** όπου η αλληλεπίδραση μεταξύ αντικειμένων είναι σημαντική.

DR AVGOUSTA KYRIAKIDOU

ΣΥΓΚΡΙΣΗ ΜΕ ΕΝΑ ΠΑΡΑΔΕΙΓΜΑ 🚙

Σχεδιάζουμε ένα **σύστημα για ένα αυτοκίνητο**

📝 **Με SSAD:**

- Θα το σπάσουμε σε διαδικασίες και ροή δεδομένων: **"Ενεργοποίησε κινητήρα" → "Έλεγξε καύσιμα" → "Προώθησε κίνηση" κτλ.**

📌 **Όλα αυτά καταγράφονται με ένα Διάγραμμα Ροής Δεδομένων (DFD), που δείχνει τι συμβαίνει πρώτο, τι δεύτερο κ.λπ.**

🧱 **Με OOAD:**

- Το σύστημα θα αποτελείται από **αντικείμενα** που συνεργάζονται σαν ένα lego όπως:
  - **Αυτοκίνητο** 🚗 (Χαρακτηριστικά: χρώμα, μάρκα, μοντέλο – Λειτουργίες: Επιτάχυνση, επιβράδυνση, αλλαγή ταχύτητας)
  - **Κινητήρας** 🛠️ (Χαρακτηριστικά: Ισχύς, τύπος καυσίμου– Λειτουργίες: Ξεκινά, σταματά)
  - **Τροχοί** 🛞 (Χαρακτηριστικά: Μέγεθος, υλικό – Λειτουργίες: Περιστροφή)
  - **Οδηγός** 👨‍✈️ (Χαρακτηριστικά: Όνομα, ηλικία, επίπεδο εμπειρίας – Λειτουργίες: Οδηγεί)
- Κάθε αντικείμενο ξέρει **τι να κάνει από μόνο του και συνεργάζονται μεταξύ τους.**
- Όταν ο **οδηγός** δώσει εντολή για εκκίνηση, το **αυτοκίνητο** καλεί τη μέθοδο **"ξεκινά"** του **κινητήρα**.
- Ο **κινητήρας** στέλνει μήνυα στους **τροχούς**, που αρχίζουν να περιστρέφονται.

ΣΥΓΚΡΙΣΗ SSAD VS OOAD ΣΤΟ ΑΥΤΟΚΊΝΗΤΟ 🚗

	SSAD (Δομημένη)	**OOAD** (Αντικειμενοστρεφές)
Προσέγγιση	Σκέφτεται το σύστημα ως **μια σειρά από βήματα**	Σκέφτεται το σύστημα ως **ένα σύνολο αντικειμένων που συνεργάζονται**
Δομή	Χωρίζεται σε **διαδικασίες και ροές δεδομένων**	Χωρίζεται σε **αντικείμενα με ιδιότητες και λειτουργίες**
Αλλαγές & Επεκτασιμότητα	Δύσκολο να αλλάξει – αν αλλάξει κάτι, πρέπει να ξανασχεδιάσουμε όλο το σύστημα	Εύκολο να αλλάξει – μπορούμε να αντικαταστήσουμε αντικείμενα χωρίς να πειράξουμε τα υπόλοιπα
Παράδειγμα	"Γύρνα το κλειδί" → "Έλεγξε καύσιμα" → "Ενεργοποίησε κινητήρα" → "Κινήσου"	"Οδηγός" ζητά από το "Αυτοκίνητο" να κινηθεί → Το "Αυτοκίνητο" ζητά από τον "Κινητήρα" να ξεκινήσει → Οι "Τροχοί" περιστρέφονται

INTRODUCING THE OO APPROACH

OOAD: Μια τεχνική για την ανάλυση, τον προσδιορισμό και τη μοντελοποίηση των λειτουργικών απαιτήσεων ενός συστήματος.

Object-oriented Analysis

- Focus is on understanding the **problem -** Έμφαση δίνεται στην κατανόηση του προβλήματος
- **What** the system does - **Τι** κάνει το σύστημα
- **Logical (abstract model)**: πτυχές του συστήματος που μπορούν να σχεδιαστούν χωρίς γνώση της πλατφόρμας υλοποίησης (implementation platform)
- Διαδικασία διερεύνησης και μοντελοποίησης των απαιτήσεων

INTRODUCING THE OO APPROACH

Object oriented Design

- Focus is on understanding the **solution - Έμφαση δίνεται στην κατανόηση της λύσης**
- **How** the system does it - **Πώς** το κάνει το σύστημα
- **Physical**: πτυχές του συστήματος που εξαρτώνται από την πλατφόρμα υλοποίησης (implementation platform) που θα χρησιμοποιηθεί
- Επεξεργάζεται και βελτιώνει περεταίρω τα μοντέλα/διαγράμματα ανάλυσης και τα μετατρέπει σε μοντέλα σχεδιασμού προκειμένου να υλοποιηθούν οι απαιτήσεις.

Object oriented Programming: μετατρέπει τα μοντέλα σχεδιασμού σε κώδικα.

OBJECT ORIENTED ANALYSIS AND DESIGN

- **Περισσότερα από 30 χρόνια ιστορίας της ΟΟ ανάλυσης (τέλη της δεκαετίας του 1980)** (μεγαλύτερη ιστορία της ΟΟ σχεδίασης και του προγραμματισμού).
  - Υπήρχε ανάγκη για πιο συνεκτικό τρόπο περιγραφής συστημάτων που συνέδεαν στενά τα δεδομένα (data) και συμπεριφορά (behaviour/processes) σε ένα αντικείμενο (object).

- Προσπάθεια να επανασυνδεθούν τα **data, process and behaviour** ξανά.
  - Κρατώντας τα τρία στοιχεία μαζί, είναι ευκολότερο να μοντελοποιηθεί το πραγματικό πρόβλημα.
  - Η ανάλυση (OOA), η σχεδίαση (OOD) και ο προγραμματισμός (OOP) στις προσεγγίσεις ΟΟ συγχωνεύονται μεταξύ τους. → Καλύτερη επικοινωνία μεταξύ των developers.

Σε αντίθεση με το SSAD που κρατάει τα δεδομένα (data) και το behaviour (process) χωριστά (χρησιμοποιώντας διαγράμματα ροής δεδομένων για τη μοντελοποίηση του behaviour/process και ERDs για τη μοντελοποίηση των δεδομένων (data).

Όχι πλέον, όπως στην SSAD, μια προσέγγιση από πάνω προς τα κάτω, διαίρει και βασίλευε (top-down, divide-and-conquer), αλλά διατηρώντας αυτά τα βασικά στοιχεία μαζί με έναν τρόπο που είναι πιο κοντά στη μοντελοποίηση του πραγματικού προβλήματος.

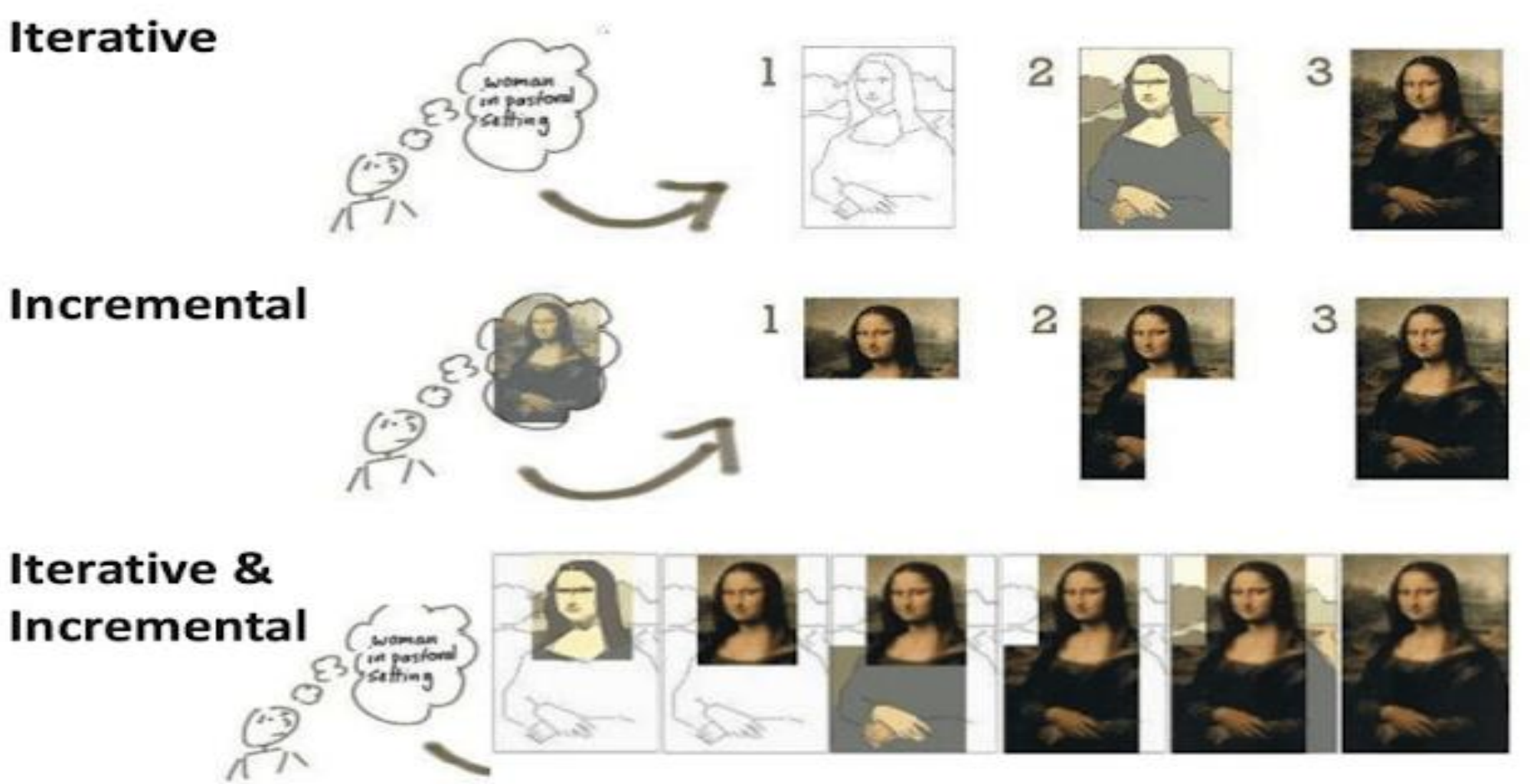
OBJECT ORIENTED ANALYSIS AND DESIGN

- Το πρόβλημα αντιμετωπίζεται ως αντικείμενα (πράγματα) **objects (things) with certain characteristics** με συγκεκριμένα χαρακτηριστικά που αλληλεπιδρούν μεταξύ τους.
- Το σύστημα διαμορφώνεται με τη μορφή ιεραρχιών αντικειμένων (objects) που μοιράζονται/κληροδοτούν γενικά χαρακτηριστικά, αλλά αναγνωρίζουν και τα ιδιαίτερα χαρακτηριστικά του κάθε αντικειμένου.

- Παρέχει ένα καθαρότερο και καλά ολοκληρωμένο μοντέλο της συμπεριφοράς του συστήματος.

- **Εξομαλύνει τη μετάβαση μεταξύ ανάλυσης, σχεδίασης και προγραμματισμού.**
  - Υποστηρίζει την επαναληπτική (επαναλαμβανόμενοι κύκλοι) και την αυξητική(σε μικρότερα τμήματα κάθε φορά) διαδικασία ανάπτυξης (**iterative and incremental development process.)**
    - η ανάλυση, ο σχεδιασμός και η ανάπτυξη συμβαίνουν σε επαναλαμβανόμενους κύκλους και σε μικρότερα τμήματα κάθε φορά.

ITERATIVE AND INCREMENTAL DEVELOPMENT – ΕΠΑΝΑΛΗΠΤΙΚΗ/ΑΥΞΗΤΙΚΗ ΑΝΑΠΤΥΞΗ



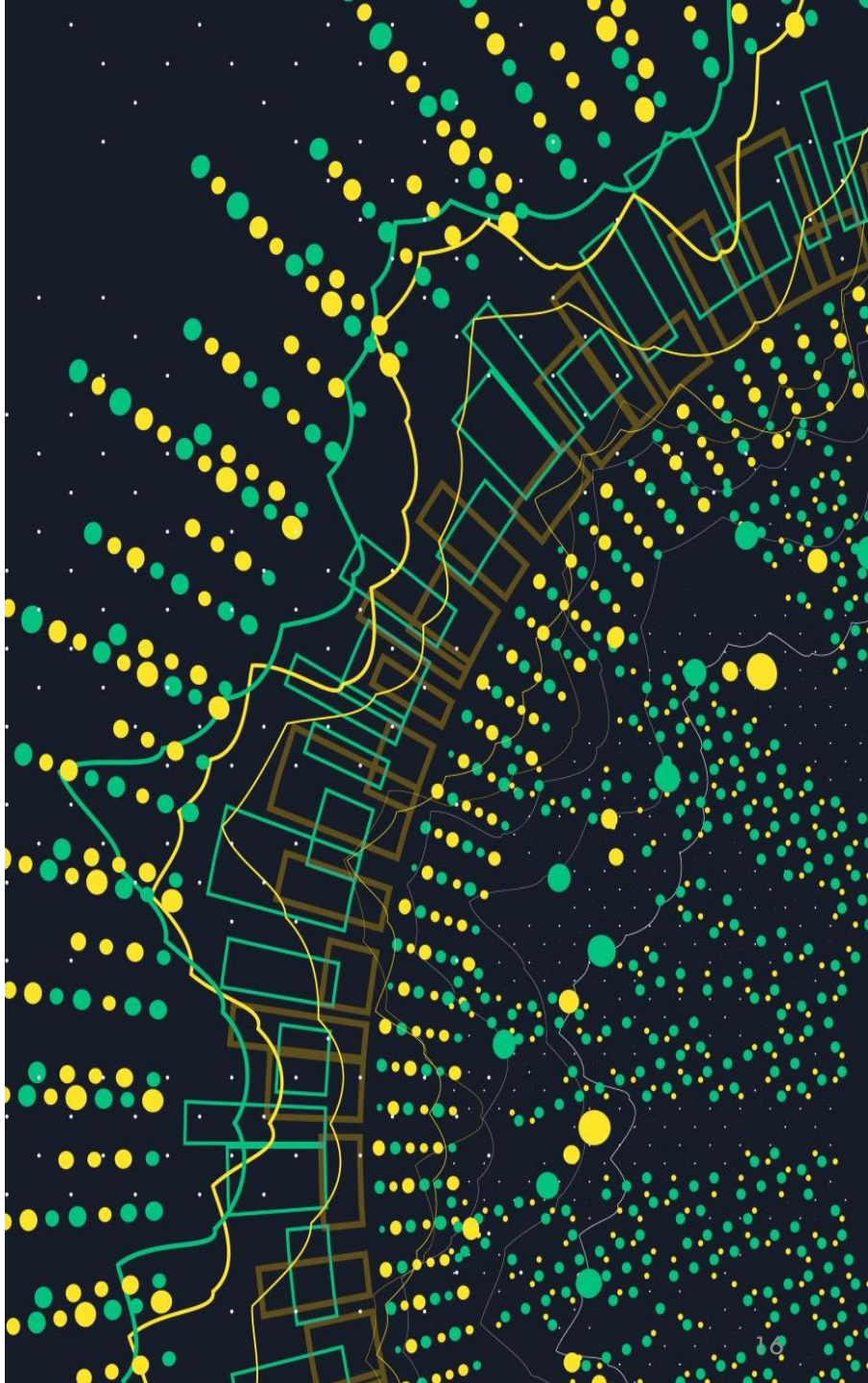
OBJECT ORIENTED ANALYSIS AND DESIGN

Iterative:

- Επιτρέπει στους αναλυτές να επωφεληθούν από όσα έμαθαν κατά την προηγούμενη ανάλυση.
- Οποιεσδήποτε αλλαγές γίνουν στις προδιαγραφές/απαιτήσεις μπορούν εύκολα να γίνουν χωρίς μεγάλες επιβαρύνσεις.

Incremental:

- Συχνότεροι έλεγχοι στο πως τα διάφορα μέρη του συστήματος δουλεύουν μαζί. Περισσότερο confident ότι το έργο είναι στο σωστό δρόμο.
- Στόχος είναι η υλοποίηση ενός μέρους του συστήματος την φορά αλλά σου επιτρέπει επίσης να προβλέψεις πρόσθετες λειτουργίες και να ανακαλύψεις σφάλματα η επιπλέον ανάγκες για το επόμενο σπριντ.
- Ελαχιστοποιείται η προσπάθεια για έλεγχο και testing αφού αυτό γίνεται συνεχώς.
- **Διαγράμματα πιο εμπλουτισμένα με πληροφορίες**
  - μπορούν ευκολότερα να μετατραπούν σε κώδικα
  - Η συντήρηση(maintenance) είναι ευκολότερη.



OBJECT ORIENTED MODELLING

- **Identifying objects (Αναγνώριση αντικειμένων):**
  χρησιμοποιώντας encapsulation concepts, CRC cards, etc.

- **Organising the objects (Οργάνωση των αντικειμένων):**
  ταξινόμηση των αντικειμένων που έχουν αναγνωριστεί, έτσι ώστε παρόμοια αντικείμενα να μπορούν μετά να οριστούν στην ίδια κλάση (class)

- **Identifying relationships between objects (Προσδιορισμός σχέσεων μεταξύ αντικειμένων):**
  πως τα αντικείμενα επικοινωνούν μεταξύ τους και τα μηνύματα που ανταλλάζουν

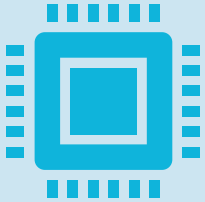
- **Defining behaviours (operations) of objects (Καθορισμός των συμπεριφορών (μεθόδων των αντικειμένων):**
  ο τρόπος με τον οποίο τα δεδομένα επεξεργάζονται εντός ενός αντικειμένου

- **Defining objects internally (Εσωτερικός ορισμός αντικειμένων):**
  Τι πληροφορίες, δεδομένα (data/attributes) αποθηκεύονται εντός των αντικειμένων

ΑΛΛΑ ΤΙ ΕΙΝΑΙ ΈΝΑ ΑΝΤΙΚΕΙΜΕΝΟ - OBJECT?

Η αντικειμενοστρεφής προσέγγιση αποσκοπεί στο να φέρει τον προγραμματισμό ενός συστήματος πιο κοντά στο να σκεφτόμαστε τον πραγματικό κόσμο. (make thinking about programming closer to thinking about the real world)

Τι είναι ένα αντικείμενο στον προγραμματισμό πρέπει πρώτα να αναρωτηθούμε τι είναι ένα αντικείμενο στον πραγματικό κόσμο.



Object: Τα αντικείμενα είναι οντότητες ενός συστήματος λογισμικού οι οποίες αντιπροσωπεύουν στιγμιότυπα πραγματικών οντοτήτων και οντοτήτων του συστήματος. Κάθε αντικείμενο χαρακτηρίζεται από ένα σύνολο ιδιοτήτων (attributes) και συμπεριφοράς (methods)

WHAT IS AN OBJECT?

BUT WHAT IS A THING?

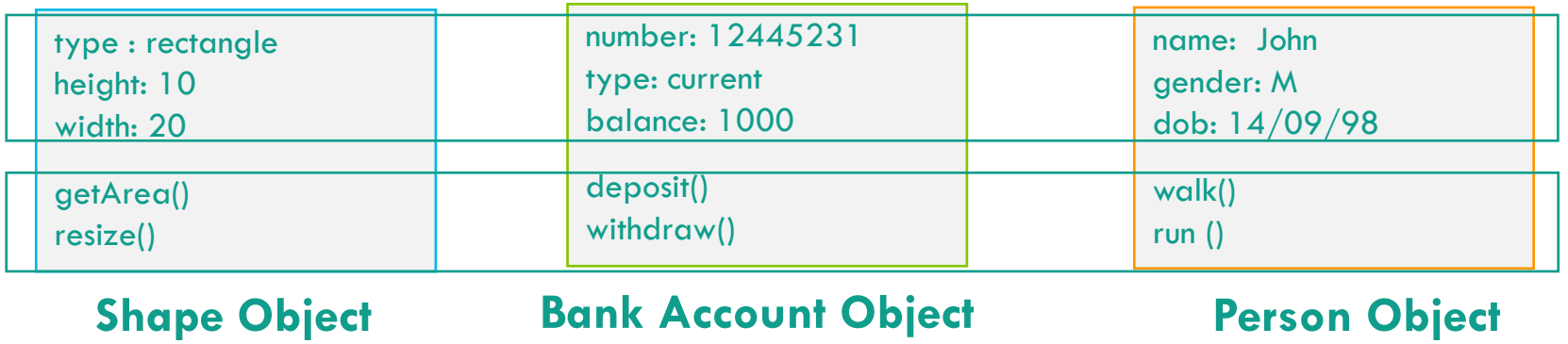
Types of objects may include a person, place, tangible things or event:

Person objects: student, customer, inspector	**Place objects:** warehouse, building, room	**Things objects:** product, vehicle, equipment	**Event objects:** Order, payment, invoice, reservation, registration, application,

These things exist within the system's environment, and equally within the system.

OBJECTS

- **Οντότητες που κάνουν encapsulate data and behaviour**
  - Στα αντικείμενα οι χρήστες κάνουν store data and associate behaviour
- **Τα δεδομένα (Data) είναι τα χαρακτηριστικά ενός αντικειμένου**
  - ιδιότητες που τα περιγράφουν (π.χ. ένα δοχείο μπορεί να είναι γεμάτο ή άδειο, μια λάμπα μπορεί να είναι αναμμένη ή σβηστή, ένα μήλο μπορεί να είναι κίτρινο ή κόκκινο
  - αυτά είναι τα **χαρακτηριστικά** κάθε αντικειμένου, πράγματα όπως το **χρώμα, το μέγεθος και το βάρος** - περιγράφουν την **τρέχουσα κατάσταση (state)** ενός αντικειμένου - η κατάσταση (state) ενός αντικειμένου είναι ανεξάρτητη από την κατάσταση του άλλου



OBJECTS

- **Τα αντικείμενα δεν είναι πάντα φυσικά αντικείμενα ή ορατά αντικείμενα, π.χ. αυτοκίνητο, σπίτι, προϊόν, μήλο.**
  - A date, a bank account, timer can be an object – they still meet our idea of an object

- **Κάθε αντικείμενο έχει ταυτότητα (identity)**
  - it can be referred to individually - μπορεί να αναφέρεται ξεχωριστά
  - it is distinguishable from other objects - διακρίνεται από άλλα αντικείμενα

- **Μπόρείτε να βάλετε την λέξη "το" "the" μπροστά τους?**
  - The mug, the apple, the bank account, the date.
  - Δεν θα λέγαμε the saving, the printing, αυτά είναι ρήματα (verbs) και δείχνουν συμπεριφορά (behaviours) των αντικειμένων
  - Τα **αντικείμενα** συνήθως αναπαραστούνται από **ουσιαστικά** (nouns) ενώ η συμπεριφορά τους (**operations/methods)** του αντικειμένου από **ρήματα** (verbs)

CLASSES - ΚΛΑΣΕΙΣ

- **Τα αντικείμενα και οι κλάσεις (classes) πάνε χέρι-χέρι.**
  - Δεν μπορούμε να μιλήσουμε για το ένα χωρίς να μιλήσουμε για το άλλο.
  - Το OOAD αφορά τις κλάσεις επειδή χρησιμοποιούμε κλάσεις για να δημιουργήσουμε αντικείμενα

- Μια **κλάση** είναι ένα σχέδιο ή ένα πρότυπο ή ένα σύνολο οδηγιών που χρησιμοποιούνται για τη δημιουργία ενός συγκεκριμένου τύπου αντικειμένου (object instances) - **περιγράφει τι θα είναι ένα αντικείμενο, αλλά δεν είναι το ίδιο το αντικείμενο.**

- Μια κλάση έρχεται πάντα πρώτη

- Μια αφαιρετικότητα δεδομένων (abstraction) η οποία καθορίζει τα attributes /behaviour ενός συνόλου αντικειμένων
  - Κάθε κλάση, περιγράφει ένα σύνολο αντικειμένων με: ***κοινά χαρακτηριστικά*** (attributes), ***κοινή συμπεριφορά*** (operations/methods), ***κοινές σχέσεις*** με άλλα αντικείμενα και ***κοινή σημασιολογία***
  - ***Όλα τα αντικείμενα μιας κλάσης είναι πανομοιότυπα ως προς τη δομή (χαρακτηριστικά) και τη συμπεριφορά, αλλά περιέχουν διαφορετικά δεδομένα στα χαρακτηριστικά τους.***

CLASSES

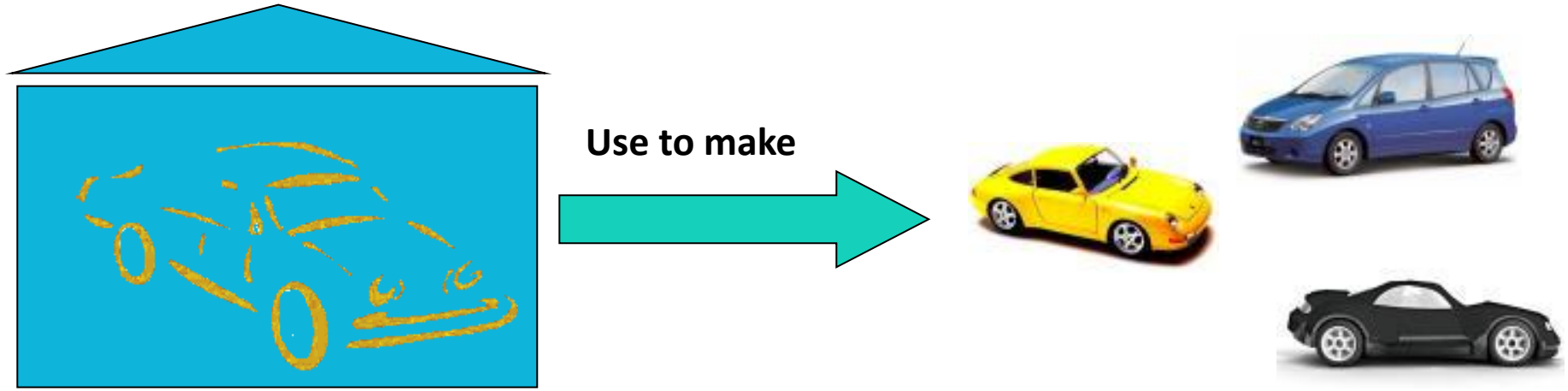
- **Μια καλή κλάση καταγράφει μία και μόνο μία αφαιρετικότητα (abstraction). Θα πρέπει να έχει ένα κύριο θέμα**
  - Π.χ. αντιπροσωπεύει ένα είδος προσώπου, τόπου, γεγονότος ή πράγματος για το οποίο το σύστημα θα πρέπει να καταγράφει και να αποθηκεύει πληροφορίες.
  - Κάθε αντικείμενο είναι μια περίπτωση κάποιας κλάσης (instance of a class) και τα αντικείμενα δεν μπορούν να είναι περιπτώσεις (instances) περισσότερων από μία κλάσεων

THE DIFFERENCE BETWEEN OBJECTS AND CLASSES

- Μια κλάση είναι ένα σχέδιο, ή πρωτότυπο, που ορίζει τα χαρακτηριστικά και τις μεθόδους που είναι κοινές για όλα τα αντικείμενα ενός συγκεκριμένου είδους.

Class car:

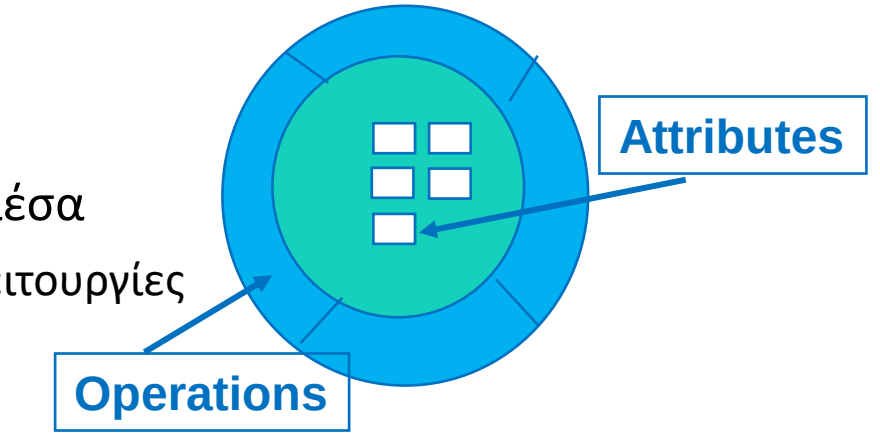
- Attributes: make, colour, type, weight etc.
- Behaviour: start_engine(); drive()



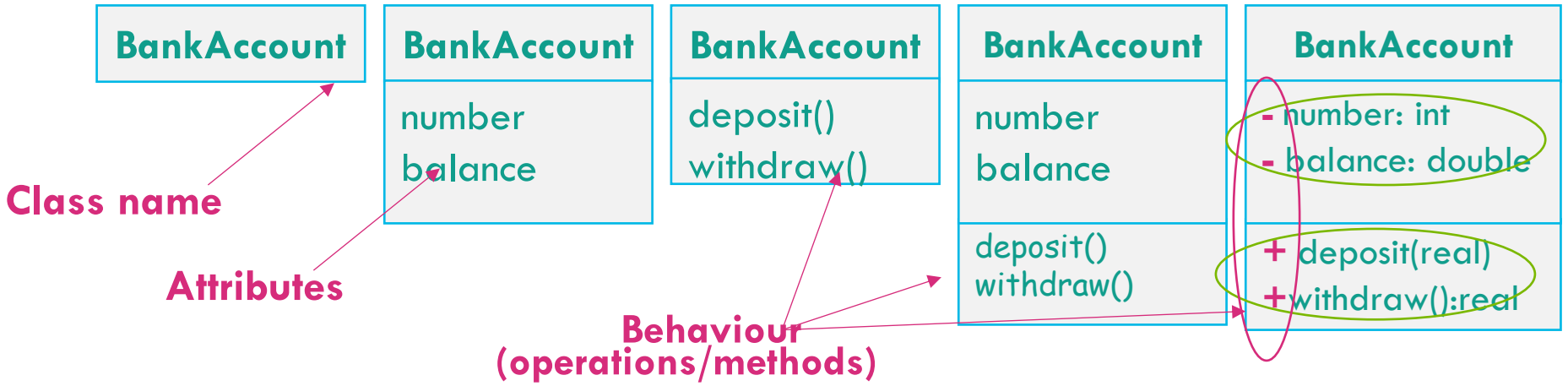
Class (the blueprint)

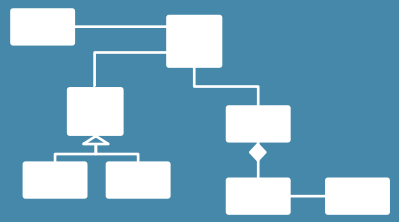
Objects

UML CLASS NOTATION



- Απλά αναπαρίσταται ως ένα κουτί με το όνομα της κλάσης μέσα
  - το οποίο δείχνει επίσης τα χαρακτηριστικά (attributes) και τις λειτουργίες (attributes and operations)
  - the complete signature of an attribute /operation is:
    - <attributeName>:<type>
    - <operationName(parameterName:paramType…)>:<returnType>
    - **Visibility**: private (-), public (+), protected (#), or package(~)





UML NOTATION
CLASS VS OBJECTS

Class

Book
title: String author : String isbn: String publisher: String price: Real
getQuantity() get_bookDetails() set_bookDetails() deletebook()

Object – Instance of Book

Book : myBook
title = “Software Engineering” author = “Pressman” isbn = “2-5345-53455-6” publisher = “McGrawHill” price = 40.00
getQuantity() get_bookDetails() set_bookDetails() deletebook()

OBJECT-ORIENTED OBJECT/CLASS PRINCIPLES

Identity - ένα αντικείμενο γνωρίζει ποιο είναι

Classification - ένα αντικείμενο γνωρίζει την κλάση (class) στην οποία ανήκει
classify objects to later create similar objects as needed

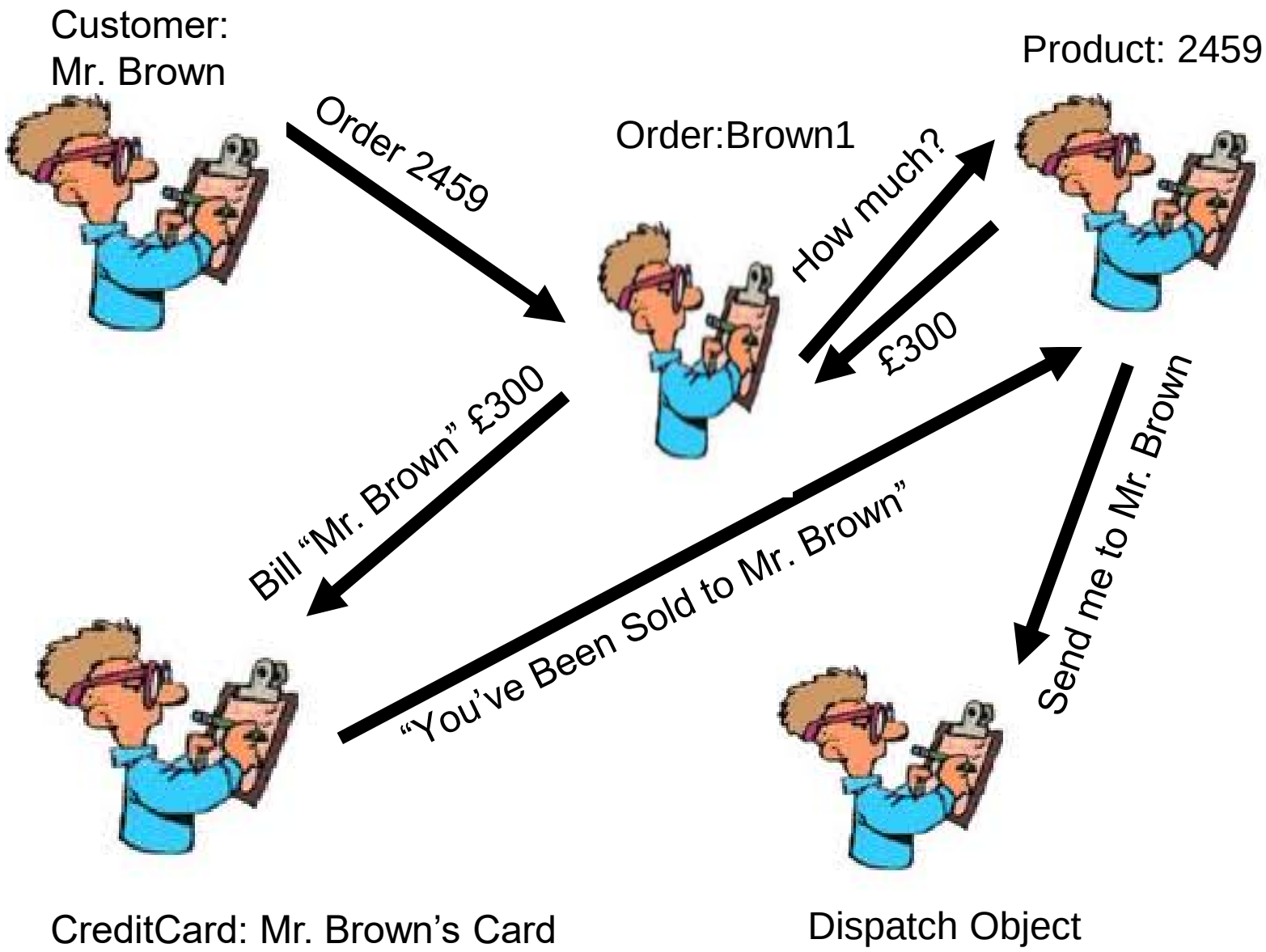
Relationships - ένα αντικείμενο ξέρει με ποια αλλά αντικείμενα πρέπει να αλληλοεπιδρά
this helps to determine the inputs and outputs of an object

Abstract behaviour - τα αντικείμενα εκτελούν ενέργειες/λειτουργίες, έχουν συμπεριφορά

the way data is processed within an object

Για να ορίσετε τα αντικείμενα, κάντε ερωτήσεις όπως

- Ποια είναι τα πράγματα (things) (classes/objects) τα οποία εμπλέκονται στο σύστημα?
- Τι πρέπει να γνωρίζω για να είμαι σε θέση να εκτελέσω το ρόλο μου; ? Τι δεδομένα (data) πρέπει να κατέχω ? **(attributes)**
- Τι πρέπει να κάνω για να μπορώ να εκτελώ το ρόλο μου; Τι λειτουργίες πρέπει να μπορώ να εκτελώ? Ποιο είναι το behaviour μου? **(methods)**
- Ποιον πρέπει να γνωρίζω για να μπορώ να εκτελώ το ρόλο μου; Με ποιες άλλες κλάσεις πρέπει να συνδέομαι/συνεργάζομαι? **(relations/visibility)**


Customer:
Mr. Brown
Order 2459
Order:Brown1
How much?
Product: 2459
£300
Bill "Mr. Brown" £300
"You've Been Sold to Mr. Brown"
Send me to Mr. Brown
CreditCard: Mr. Brown's Card
Dispatch Object

APIE OBJECT ORIENTED PRINCIPLES

Υπάρχουν διάφορες έννοιες στην αντικειμενοστρεφή προσέγγιση που μας βοηθούν να αντιμετωπίσουμε τα προβλήματα.

Το OOAD βασίζεται σε ένα σύνολο αρχών που χρησιμεύουν στη διαχείριση και τη μείωση της πολυπλοκότητας του συστήματος.

APIE - Abstraction, Polymorphism, Inheritance, Encapsulation

ABSTRACTION - ΑΦΑΙΡΕΤΙΚΟΤΗΤΑ

Abstraction - Αφαιρετικότητα: αποτύπωση βασικών πτυχών, αγνόηση άσχετων λεπτομερειών

- Περιλάβετε μόνο τα σχετικά και σημαντικά στοιχεία σε μια κλάση
- Υποστηρίζεται από δυο κύριες ιδέες:
  - **Information Hiding (απόκρυψη πληροφοριών)**
  - **Encapsulation (ενθυλάκωση)**

INFORMATION HIDING - ΑΠΟΚΡΥΨΗ ΠΛΗΡΟΦΟΡΙΩΝ

Information Hiding

- **Μια κλάση θα πρέπει να αποκρύπτει από άλλες κλάσεις τις λεπτομέρειες του τρόπου με τον οποίο υλοποιείται εσωτερικά ως κώδικας υπολογιστή.**

- Επιτρέπει τα δεδομένα (attributes) να μην είναι ορατά στον έξω κόσμο, **έτσι οι μέθοδοι (methods) είναι αυτές που επιτρέπουν σε άλλα αντικείμενα να έχουν πρόσβαση στα δεδομένα με ελεγχόμενο τρόπο.**

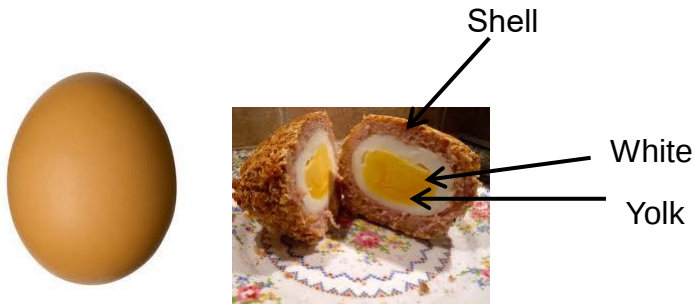
- Η απόκρυψη πληροφοριών είναι χρήσιμη καθώς οι κλάσεις και τα δεδομένα τους **δεν μπορούν να αλλοιωθούν από άλλες κλάσεις που δεν πρέπει να τις χρησιμοποιούν**.

Clipboard example: δεν μπορούμε να δούμε τις πληροφορίες στο σημειωματάριο (clipboard) του άλλου. Πρέπει να ζητήσουμε άδεια για να έχουμε πρόσβαση σε αυτές τις πληροφορίες.

ENCAPSULATION - ΕΝΘΥΛΑΚΩΣΗ

Encapsulation:

- **Συνδυασμός τόσο των χαρακτηριστικών (attributes) όσο και της συμπεριφοράς (methods) σε μια κλάση και απόκρυψη της μη σχετικής λεπτομέρειας σε μια δεδομένη κλίμακα**
  - Εάν βάλεις τα δεδομένα και την συμπεριφορά πίσω μαζί σε ένα αντικείμενο τότε υποστηρίζεται η αφαιρετικότητα (abstraction) επειδή τα αντικείμενα μπορούν να χρησιμοποιηθούν σαν **building blocks** που μπορούν να συναρμολογηθούν για να δημιουργήσουν το σύστημα.
  - **Encapsulation allows for a more robust model, easier to reuse**



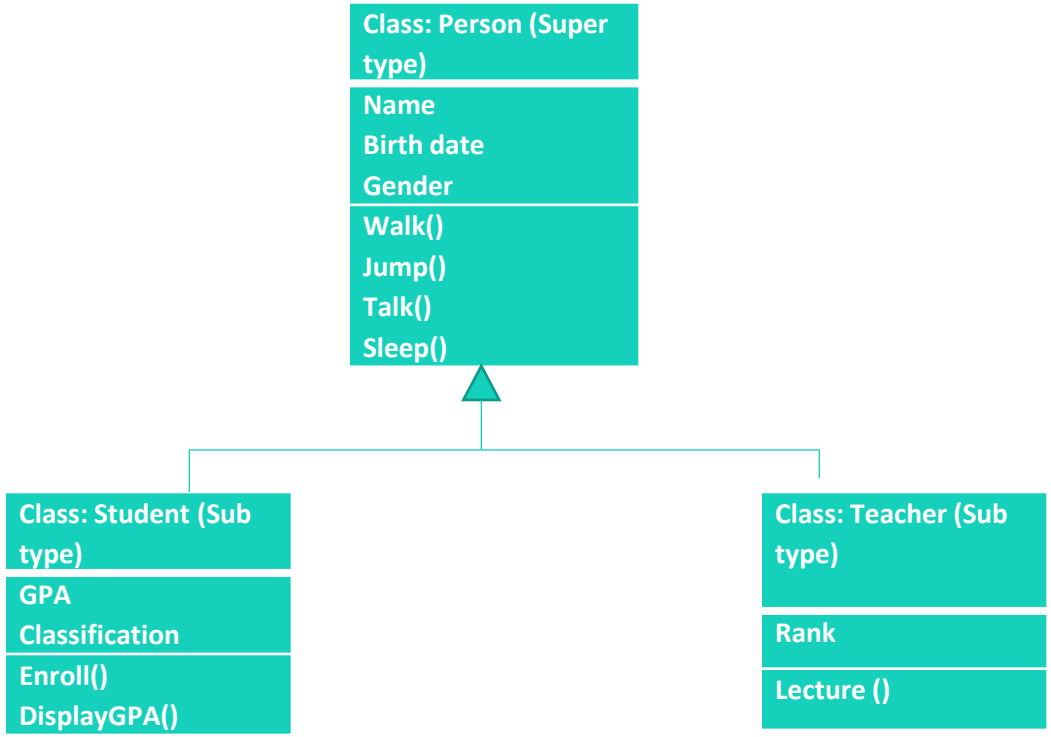
Example: Object is like an Egg

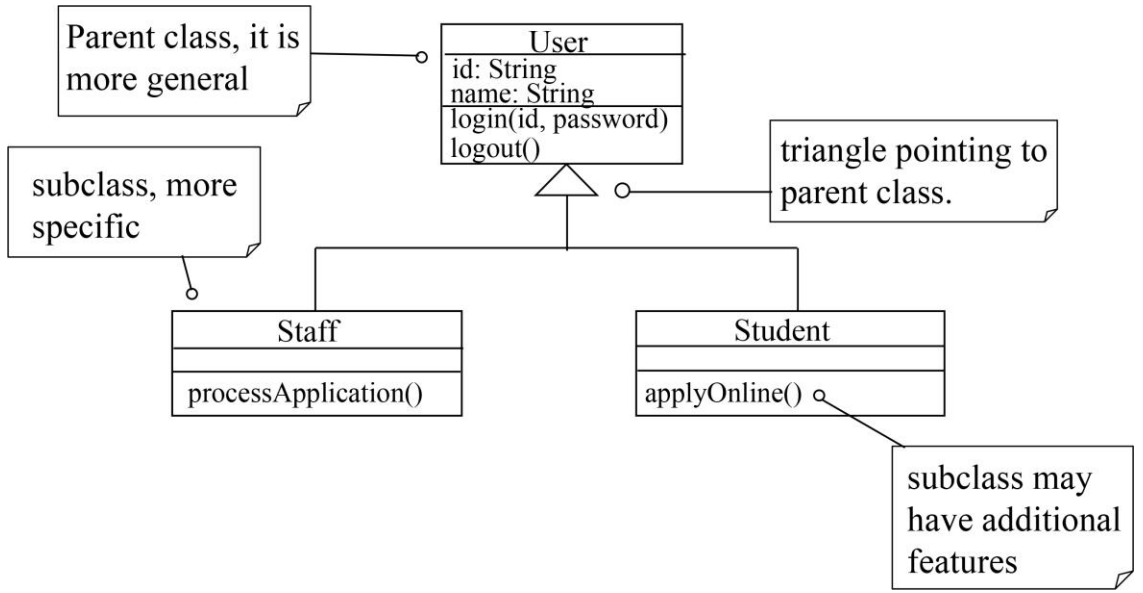
- Ο κρόκος είναι τα δεδομένα (**attributes**) που περιβάλλονται από το ασπράδι. Το ασπράδι είναι οι μέθοδοι (methods) που ενεργούν στα δεδομένα..
- Το κέλυφος περιβάλλει όλο το αυγό και το κρατάει ενωμένο ενώ κρύβει το εσωτερικό του από τον εξωτερικό κόσμο. Το κέλυφος είναι η διεπαφή και το μόνο πράγμα που φαίνεται.
- **Με την ενθυλάκωση και την απόκρυψη πληροφοριών, η επιβάρυνση κατά τη συντήρηση μειώνεται, καθώς είστε σίγουροι ότι όταν διορθώνετε ένα πρόβλημα σε μια κλάση, επηρεάζεται μόνο ο κώδικας της συγκεκριμένης κλάσης.**

INHERITANCE - ΚΛΗΡΟΝΟΜΙΚΟΤΗΤΑ

Inheritance

- Ένα αντικείμενο από μια πιο συγκεκριμένη κλάση μπορεί να κληρονομήσει τα χαρακτηριστικά(attributes) και τις μεθόδους (methods) από μια πιο γενική κλάση.



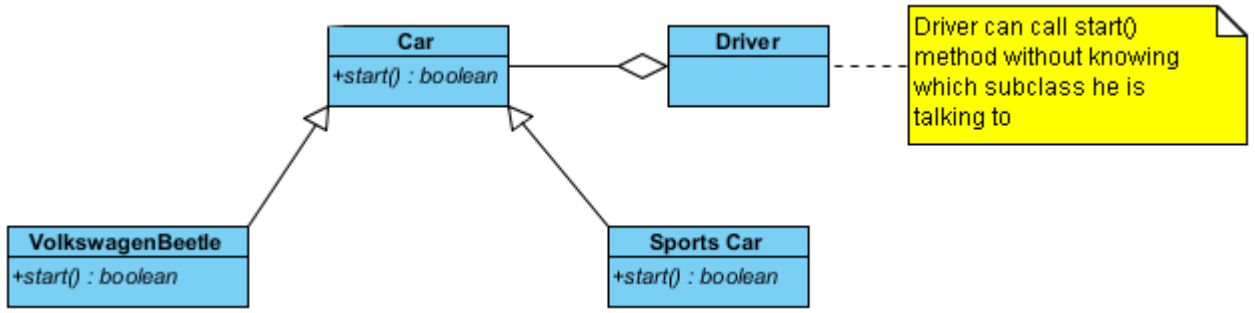


POLYMORPHISM - ΠΟΛΥΜΟΡΦΙΣΜΟΣ

Polymorphism

- Τα γενικά χαρακτηριστικά μιας κλάσης μπορούν να κληρονομηθούν αλλά να αντικατασταθούν αν χρειαστεί.
- Επιτρέπει σε μεθόδους διαφορετικών κλάσεων να έχουν το ίδιο όνομα, ενώ οι λεπτομέρειες της πραγματικής λειτουργίας που εκτελείται μπορεί να είναι διαφορετικές.

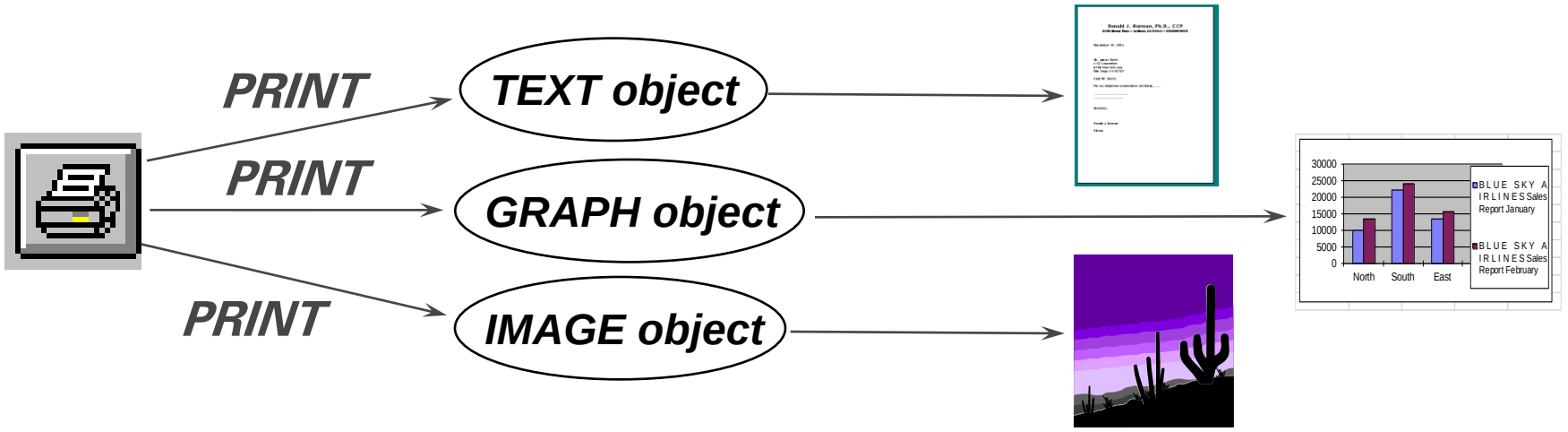
E.g. "Close method" a door can "swing shut" a window can "slide downwards"



Πώς σχετίζεται ο πολυμορφισμός με την αποστολή μηνυμάτων μεταξύ αντικειμένων;

Το αιτούμενο αντικείμενο γνωρίζει ποια συμπεριφορά (method) να ζητήσει και από ποιο αντικείμενο, αλλά δεν χρειάζεται να ανησυχεί για τον τρόπο με τον οποίο επιτυγχάνεται αυτή η συμπεριφορά.

POLYMORPHISM EXAMPLE



- **Η κληρονομικότητα και ο πολυμορφισμός μπορούν να βελτιώσουν την επαναχρησιμοποίηση.**
- **Αυτό μειώνει το κόστος σχεδιασμού, προγραμματισμού και testing.**

ΠΟΙΟ ΕΙΝΑΙ ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΗΣ ΑΝΤΙΚΕΙΜΕΝΟΣΤΡΕΦΟΥΣ ΑΝΑΛΥΣΗΣ

Το αποτέλεσμα της αντικειμενοστρεφούς ανάλυσης του συστήματος είναι διάφορα μοντέλα.

A model should:

- να χρησιμοποιεί μια τυποποιημένη γλώσσα μοντελοποίησης (θα χρησιμοποιήσουμε τη UML)
- να είναι κατανοητό από τους πελάτες και τους χρήστες
- να οδηγεί τους μηχανικούς λογισμικού σε γνώσεις σχετικά με το σύστημα
- να παρέχει αφαίρετικοτητα

USE CASE MODEL: USE CASE DIAGRAM
ΜΟΝΤΕΛΟ ΠΕΡΙΠΤΩΣΕΩΝ ΧΡΗΣΤΗ

- Οδηγεί την ανάλυση - Προσδιορίζει και περιγράφει τις λειτουργίες του συστήματος από την οπτική γωνία του εξωτερικού χρήστη, χρησιμοποιώντας ένα εργαλείο που ονομάζεται περιπτώσεις χρήσης (Use cases).

- Αποσκοπεί στην παροχή μιας υψηλού επιπέδου και προσανατολισμένης προς τον χρήστη περιγραφής ενός συστήματος – χωρίς να εμβαθύνουμε στην υλοποίηση του συστήματος.

  - Προσδιορίζουμε τους **Actors** του συστήματος – **πιθανοί χρήστες** ή **άλλα συστήματα** που πρέπει να **αλληλοεπιδρούν με το δικό μας**.
  - Προσδιορίζουμε τα **Use case (περιτπώσεις χρήσεις)** σαν την λειτουργία του συστήματος την οποία χρειάζεται ο χρήστης.
  - Δείχνουμε σχεδιαστικά την αλληλεπίδραση των δυο
  - Δημιουργούμε scenarios για να εξηγήσουμε αυτή την αλληλεπίδραση

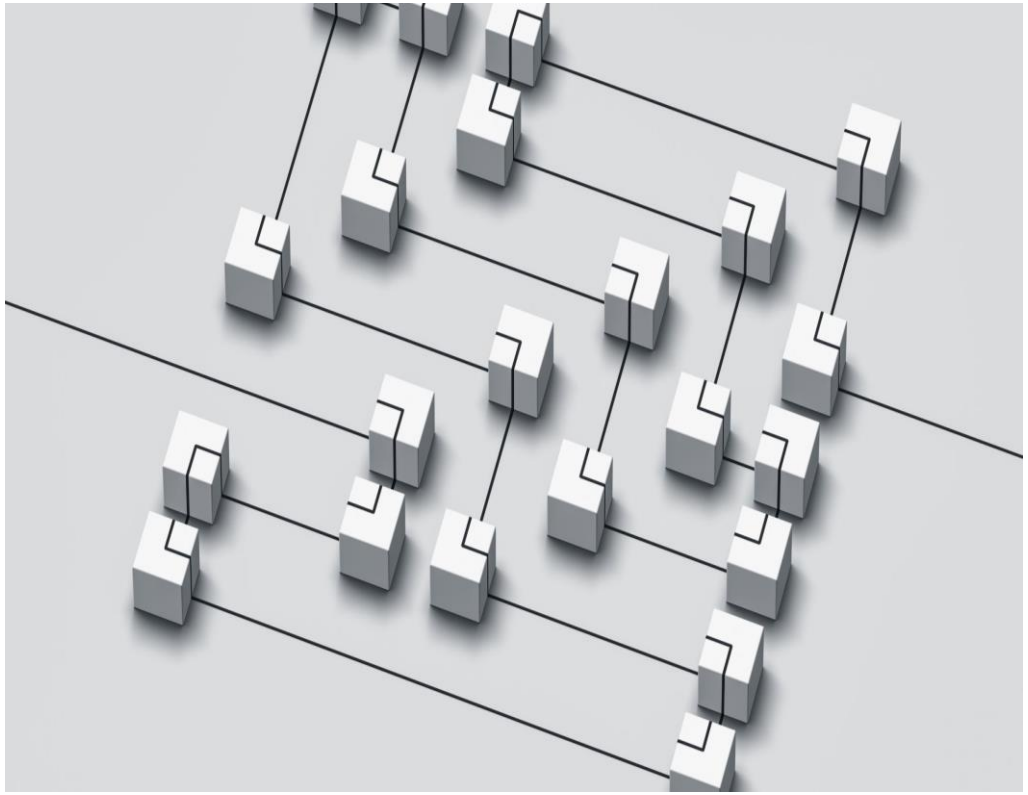
OBJECT MODEL - CLASS DIAGRAM - **ΜΟΝΤΕΛΟ ΚΛΑΣΕΩΝ**
(CONCEPTUAL VS DESIGN CLASS DIAGRAM)

Προσδιορίζει τις κλάσεις(Classes):

- Names
- Attributes
- Visibility
- Obvious behaviour (methods)

Και τις σχέσεις μεταξύ τους

Μοντελοποιεί την **στατική εικόνα (static structure)** του συστήματος καθώς μοντελοποιεί το περιβάλλον εντός του οποίου λειτουργεί



DYNAMIC MODELS: OBJECT SEQUENCE DIAGRAMS AND STATE CHARTS

Object Sequence Diagram – Ακολουθιακό διάγραμμα

- Αναπαράσταση της **δυναμικής (dynamic)** συμπεριφοράς
- πώς συνεργάζονται τα αντικείμενα στο σύστημα (τι μηνύματα στέλνουν για να ζητήσουν την εκτέλεση των μεθόδων άλλων αντικειμένων)
- συνδέει τις περιπτώσεις χρήσης (use case) και το class diagrams.
- Μοντελοποιείται η αλληλεπίδραση των κλάσεων οι οποίες συνεργάζονται για να επιτευχθεί μια περίπτωση χρήσης (use case) - τι μηνύματα στέλνονται και ποιες μεθόδοι ενεργοποιούνται

State transition diagram – Διάγραμμα κατάστασης

Τι συμβαίνει στη ζωή ενός αντικειμένου σε πολλές περιπτώσεις χρήσης. Λεπτομερέστερη προβολή των κλάσεων. Βοηθά στον προσδιορισμό της περαιτέρω συμπεριφοράς των αντικειμένων – More detailed view of a class

WHICH DIAGRAM COMES FIRST?...

“Chicken and Egg” dilemma?

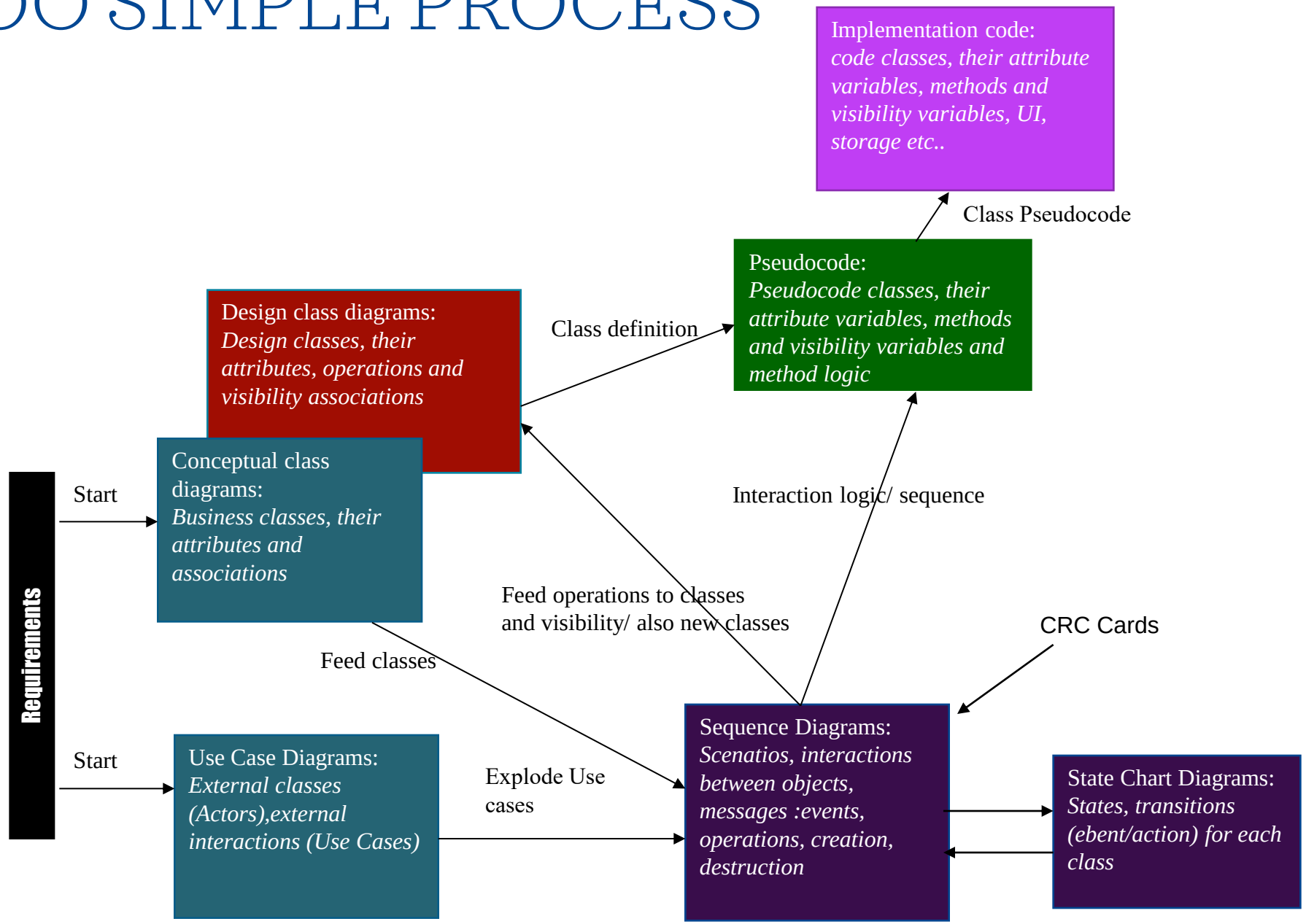
Ποιο δημιουργείται πρώτα - Το κοτόπουλο ή το αυγό;

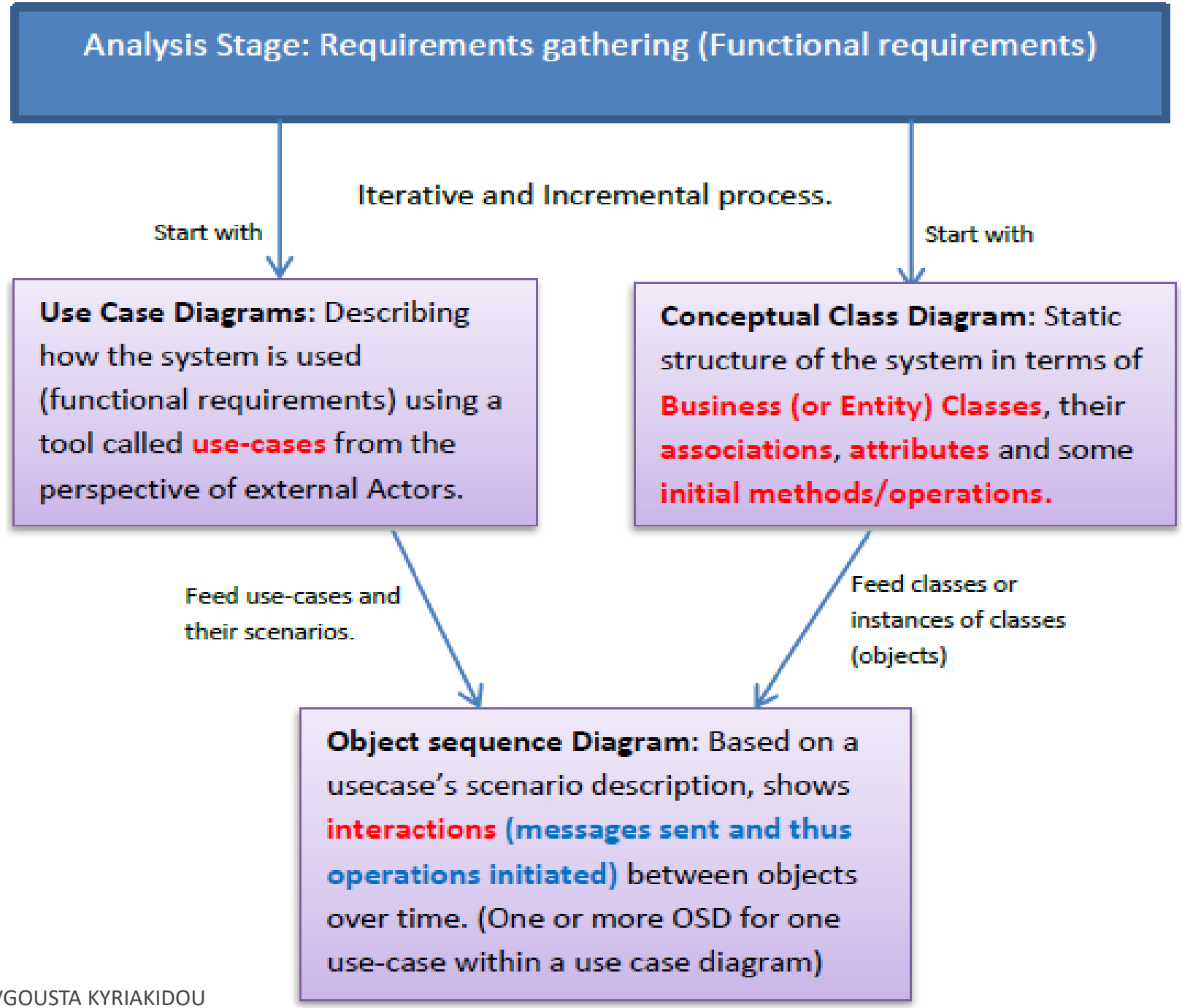
Ποιο διάγραμμα έρχεται πρώτο? Είναι μια βηματική προσέγγιση το ένα μετά το άλλο?

Στην πραγματικότητα:

- Μια επαναληπτική διαδικασία
- Χρειάζεται εμπειρία

AN OO SIMPLE PROCESS




Analysis Stage: Requirements gathering (Functional requirements)
Iterative and Incremental process.
Start with
Start with
Use Case Diagrams: Describing how the system is used (functional requirements) using a tool called use-cases from the perspective of external Actors.
Conceptual Class Diagram: Static structure of the system in terms of Business (or Entity) Classes, their associations, attributes and some initial methods/operations.
Feed use-cases and their scenarios.
Feed classes or instances of classes (objects)
Object sequence Diagram: Based on a usecase's scenario description, shows interactions (messages sent and thus operations initiated) between objects over time. (One or more OSD for one use-case within a use case diagram)

SUMMARY: KEY CONCEPTS OF OBJECT ORIENTATION

Επανασυνδέει τα δεδομένα/χαρακτηριστικά (attributes) με την συμπεριφορά/μεθόδους.

Επαναληπτική και αυξητική ανάπτυξη, επομένως ομαλή μετάβαση από την ανάλυση στον σχεδιασμό και την κωδικοποίηση

Καλύτερη επικοινωνία μεταξύ της ομάδας ανάπτυξης, όχι καταμερισμός της προσπάθειας.

Το αντικείμενο αποτελεί μια διαφορετική (ισχυρότερη) εννοιολογική βάση για τη σύλληψη του συστήματος, τη δημιουργία μοντέλων και την ανάπτυξη συστημάτων.

Όχι μόνο για την αποθήκευση δεδομένων, αλλά ενεργεί, έχει συμπεριφορά)

Άμεση εστίαση στη διαχείριση της πολυπλοκότητας και υποστήριξη της επαναχρησιμοποίησης (αφαιρετικότητα, απόκρυψη πληροφοριών, ενθυλάκωση, κληρονομικότητα κ.λπ.)

Δυναμικό μοντέλο αλλά κυρίως δυναμική μοντελοποίηση του συστήματος στο περιβάλλον του (διάγραμμα ακολουθίας αντικειμένων)

SUMMARY: KEY CONCEPTS OF OBJECT ORIENTATION

Η αντικειμενοστραφής προσέγγιση βασίζεται σε γλώσσες ΟΟ, όπως η Smalltalk, η Java, η C# ή η C++, καθώς και σε βιβλιοθήκες αντικειμένων και συστατικά (object libraries and components).

- Πιο κατάλληλη για την ανάπτυξη συστημάτων που πρέπει να είναι πιο διαδραστικά και ευέλικτα ή συστημάτων που βασίζονται στο διαδίκτυο.

Η ΟΟ μπορεί να χρησιμοποιηθεί σε συστήματα που απαιτούν συντήρηση, καθώς οι προσεγγίσεις ΟΟ επιτρέπουν ευκολότερη συντηρησιμότητα.

Η προσέγγιση ΟΟ θεωρεί δεδομένες ιδέες όπως

- spiral/risk driven προσεγγίσεις
- Δημιουργία πρωτοτύπων
- Επαναχρησιμοποίηση και component-based development
- Σταδιακή παράδοση του συστήματος (producing the right outputs in the right time)
- Εργασία σε μικρές και ευέλικτες ομάδες.

Ως εκ τούτου, η προσέγγιση αυτή μπορεί να είναι καταλληλότερη για ευέλικτα έργα.

ΠΩΣ ΤΑ ΠΗΓΑΜΕ ΑΥΤΗ ΤΗΝ ΕΒΔΟΜΑΔΑ? ΤΙ ΜΑΘΑΜΕ?

Reply at:
PollEv.com/avgoustakyri977

1. Go to **PollEv.com**
2. Enter **AVGOUSTAKYRI977**
3. Respond to activity

1. Text **AVGOUSTAKYRI977** to **22333**
2. Text in your message

WRAP UP

Requirements specification document: What it is, why it is important, introduced the techniques we will use (OOAD and SSAD) to specify functional requirements.

Key differences between OOAD and SSAD

What is an object, what is a class.

Key principles of object orientation - abstraction, information hiding, encapsulation, association, inheritance, polymorphism.

Introduced UML and the diagrams we will be discussing in the next lectures.

READINGS/REFERENCES

1. Kendall, K. E., και Kendall, J. E. Ανάλυση & Σχεδίαση Συστημάτων. 10η έκδοση, 2023 Εκδόσεις Μ. Γκιούρδας, Κεφάλαιο 10
2. Ε.Α. Γιακουμάκης, Ν. Α. Διαμαντίδης, Τεχνολογία Λογισμικού (2021), 2η έκδοση, Unibooks, Διαθέσιμο συμπληρωματικό υλικό online με άδεια χρήσης Creative Commons 4.0: Τεχνολογία Λογισμικού, Κεφάλαιο 5 και 12
3. Pressman, R.S. (2020). Software Engineering: A practitioner's approach, 9th Edition, Mc-Graw Hill – Chapter 7 and 8
4. Kung, D.C (2024) Software Engineering, 2nd Edition, McGraw Hill – Chapter 4 and 5