Αντικειμενοστρεφής Προγραμματισμός

Στην ενότητα αυτή θα μελετηθούν:

Εισαγωγή στον αντικειμενοστρεφή προγραμματισμό

Η έννοια της κλάσης και του αντικειμένου

Εισαγωγή στη C++

Τεχνολογία Λογισμικού

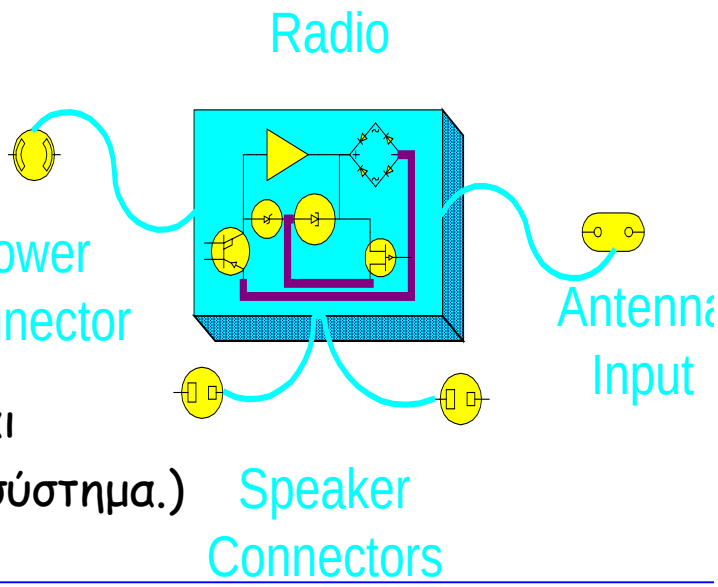
- Περιοχή της πληροφορικής που ασχολείται με το κτίσιμο μεγάλων λογισμικών συστημάτων.
- Στόχοι: Δημιουργία λογισμικών συστημάτων που είναι αξιόπιστα, ευκατανόητα, αποδοτικά ως προς το κόστος τους, ευπροσάρμοστα και επαναχρησιμοποιήσημα.
- Αρχές Τεχνολογίας Λογισμικού
  - Αφαιρετικότητα / Abstraction
  - Ενθυλάκωση / Encapsulation
  - Τμηματικότητα / Modularity
  - Ιεραρχικότητα / Hierarchy

Αφαιρετικότητα

- Διαδικασία εξαγωγής μόνο των σχετικών ιδιοτήτων ενός "αντικειμένου". Οι ιδιότητες αυτές ορίζουν μια άποψη του αντικειμένου.
- Παράδειγμα:
  - Ένας πωλητής αυτοκινήτων ασχολείται με τα στοιχεία που αφορούν την πώληση ενός αυτοκινήτου όπως τιμή, χρόνος εγγύησης, χρώμα, επιπρόσθετος εξοπλισμός κλπ.
  - Ένας μηχανικός βλέπει το αυτοκίνητο από την άποψη της συντήρησης του συστήματος, όπως είδος λαδιού, μέγεθος φίλτρου λαδιού κλπ.

Ενθυλάκωση

- Διάσπαση ενός αντικειμένου σε υπομέρη, αποκρύπτοντας και προστατεύοντας ουσιαστικές πληροφορίες και προσφέροντας κατάλληλες μεθόδους για επεξεργασία των πληροφοριών με ελεγμένο και χρήσιμο τρόπο.
- Με την απόκρυψη (της ακριβής μορφής) των πληροφοριών η παράσταση και το περιεχόμενο του αντικειμένου μπορεί να αλλάξει χωρίς να επηρεάζονται άλλα μέρη του συστήματος.
- Παράδειγμα - ράδιο αυτοκινήτου
  - Το interface αποτελείται από τα σημεία διασύνδεσης του ραδίου με το αυτοκίνητο.
  - Οι λεπτομέρειες σχετικά με το πως δουλεύει το ράδιο αποκρύπτονται. (Για την εγκατάσταση ενός ραδίου δεν χρειάζεται να γνωρίζουμε τίποτα για το ηλεκτρικό του σύστημα.)



Τμηματικότητα

- Διαμερισμός ενός αντικειμένου σε μικρότερα κομμάτια (modules) έτσι ώστε τα κομμάτια αυτά να κρατούν χρήσιμες και "αυτόνομες" πληροφορίες και το αντικείμενο είναι πιο ευκατανόητο και ευκολο στη διαχείρησή του.
- Παράδειγμα: Ένα αυτοκίνητο μπορεί να διαχωριστεί σε υποσυστήματα.
  - Cooling system
    - Radiator
    - Thermostat
    - Water pump
  - Ignition system
    - Battery
    - Starter
    - Spark plugs

Ιεραρχικότητα

- Ταξινόμηση αντικειμένων σύμφωνα με τις μεταξύ τους σχέσεις.
- Οι ιεραρχίες μας βοηθούν να κατανοήσουμε καλύτερα πολύπλοκα συστήματα.
  - Παράδειγμα: η ιεραρχία σε μια εταιρεία βοηθεί τους εργαζόμενους να κατανοήσουν καλύτερα την εταιρεία και την θέση τους σ'αυτήν.
- Ένας χρήσιμος τρόπος για ιεράρχηση παρόμοιων οντοτήτων είναι από το γενικότερο στο ειδικότερο.
  - Οι επιστήμονες χρησιμοποιούν αυτή τη τεχνική για την αναγνώριση και κατάταξη των ειδών.

Αντικειμενοστρεφής Προγραμματισμός

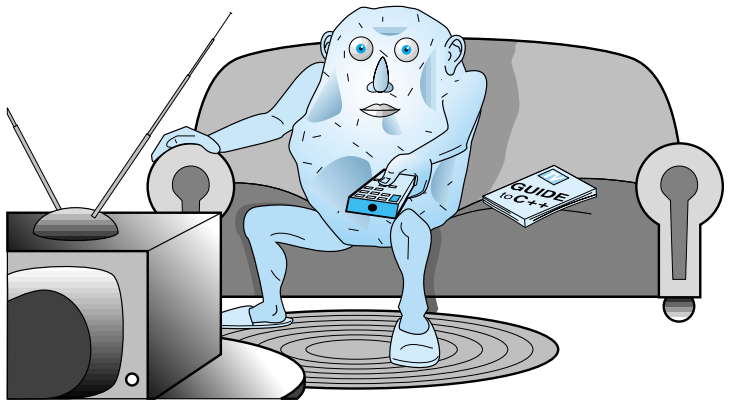
- Οι παραδοσιακές "διαδικαστικές" γλώσσες προγραμματισμού (π.χ. C, Modula-2, Fortran) αφαιρετικοποιούν τις συμβολικές γλώσσες (και κατ'επέκταση αναπαραστούν το μοντέλο ενός υπολογιστή) και επιβάλουν στον προγραμματιστή να δημιουργήσει μια σύνδεση ανάμεσα στο μοντέλο μηχανής και στο μοντέλο του προβλήματος που προσπαθεί να επιλύσει.

- Στον αντικειμενοστρεφή προγραμματισμό υπάρχει η δυνατότητα περιγραφής και αναπαράστασης των στοιχείων του πραγματικού χώρου ως αντικείμενα έτσι ώστε η επίλυση ενός προβλήματος να χρησιμοποιεί έννοιες οι οποίες πηγάζουν άμεσα από το πρόβλημα. Έτσι η περιγραφή του προβλήματος μέσω του προγράμματος γίνεται μέ τους όρους του προβλήματος και όχι της μηχανής.

Αντικείμενα

- **Αντικείμενα (objects):** κάθε οντότητα που υπάρχει στον κόσμο μας χαρακτηρίζεται από
  - Χαρακτηριστικά (attributes): όπως για παράδειγμα το μέγεθος, το χρώμα, το βάρος κλπ.
  - Συμπεριφορές (behaviours): τι δηλαδή μπορεί να κάνει ένα αντικείμενο, πως ανταποκρίνεται και πως λειτουργεί.

- Παράδειγμα:
  - το τηλεχειριστήριο και η τηλερόραση είναι αντικείμενα με χαρακτηριστικά και συμπεριφορές.
  - όταν πατήσουμε ένα κουμπί στο τηλεχειριστήριο αυτό θα στείλει το κατάλληλο μήνυμα προς την τηλεόραση και η τηλεόραση θα επιδείξει την κατάλληλη συμπεριφορά.



Αντικείμενα και κλάσεις

- Στον αντικειμενοστρεφή προγραμματισμό όλα είναι αντικείμενα και χρησιμοποιούνται για αποθήκευση δεδομένων όπως και για την εκτέλεση υπολογισμών πάνω σε δεδομένα.
- Τα αντικείμενα λένε το ένα στο άλλο τι να κάνει μέσω ανταλλαγής μηνυμάτων. *Για να εγερθεί ένα αίτημα σε κάποιο αντικείμενο στέλνουμε ένα μήνυμα προς αυτό με αποτέλεσμα την κλήση μιας συνάρτησης του αντικειμένου.*
- Κάθε αντικείμενο έχει τη δική του μνήμη η οποία αποτελείται από άλλα αντικείμενα.
- Κάθε αντικείμενο έχει τύπο:
  - Κάθε αντικείμενο είναι στιγμιότυπο μιας κλάσης (κλάση ≈ τύπος)
- Η κλάση καθορίζει τα χαρακτηριστικά και τις συμπεριφορές αντικειμένων: αντικείμενα της ίδιας κλάσης μπορούν να αποδεχθούν τα ίδια μηνύματα.

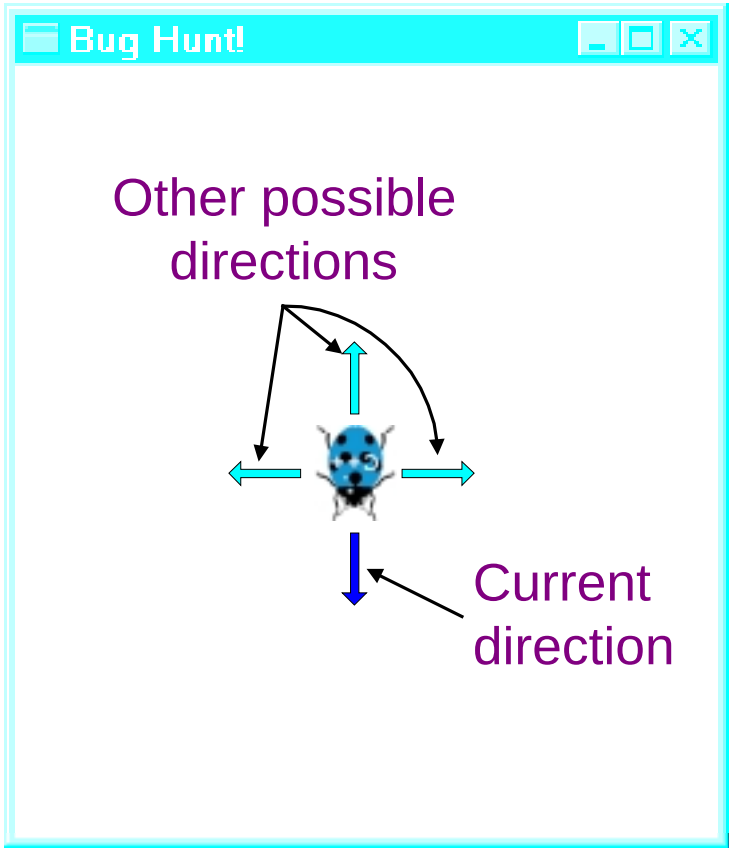
Αντικειμενοστρεφής Προγραμματισμός

- Παράδειγμα
  - Να προτείνετε δομή για πρόγραμμα που να υλοποιεί ένα απλό παιχνίδι με το όνομα Bug Hunt
  - στόχος του οποίου είναι η "εξόντωση" ζωυφίων από την οθόνη.
  - Χαρακτηριστικά του παιχνιδιού:
    - Ένα κινούμενο ζωύφιο εμφανίζεται σε ένα παράθυρο στην οθόνη
    - Το ζωύφιο μπορεί να αλλάξει ανά πάσα στιγμή την διεύθυνση κίνησής του.
    - Ένα ζωύφιο εξοντώνεται από τον παίκτη αν "κτυπηθεί" από τον παίκτη μέσω του ποντικιού κάποιο προκαθορισμένο αριθμό φορών.
    - Όταν ένα ζωύφιο εξοντωθεί ένα γρηγορότερο ζωύφιο παίρνει την θέση του.
    - Αν ένα κλικ του ποντικιού χάσει το στόχο του τότε ο παίκτης χάνει και το παιχνίδι τελειώνει.

Bug Hunt

- Βήμα πρώτο
  - Προσδιορισμός των αντικειμένων
    - Ποντίκι
    - Παράθυρο
    - Ζωύφιο
    - Ελεγκτής παιχνιδιού
- Ας επικεντρώσουμε την προσοχή μας στο αντικείμενο που αντιστοιχεί στο ζωύφιο.

Τμηματικότητα



Bug Hunt

- Για υλοποίηση του παιχνιδιού το ζωύφιο πρέπει να εφοδιαστεί με τα πιο κάτω χαρακτηριστικά:
  - Θέση στο παράθυρο
  - Μορφή (display image)
  - Ταχύτητα
  - Διεύθυνση κίνησης
  - Δύναμη (ο αριθμός των "κτυπημάτων" που απομένουν για εξόντωσή του)

- Επίσης το ζωύφιο πρέπει να ανταποκρίνεται στα εξής μηνύματα:
  - Draw
  - Move
  - Change direction
  - Hit
  - Kill
  - Is-pointed-at

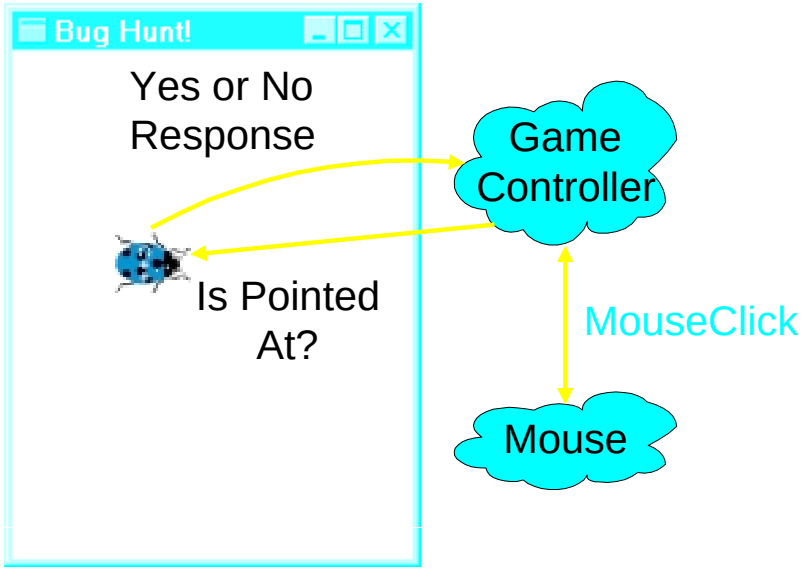
Bug Hunt

- Άλλα σημαντικά αντικείμενα είναι
  - αυτό που ρυθμίζει το παιχνίδι, ο Ελεγκτής Παιχνιδιού (ΕΠ) και
  - αυτό που αντιπροσωπεύει το ποντίκι.
- Τα τρία αυτά αντικείμενα θα επικοινωνούν και θα συνεργάζονται μεταξύ τους κατά την εξέλιξη του παιχνιδιού και θα εγείρουν μηνύματα το ένα προς το άλλο. Για παράδειγμα:
  - Κατά την εκκίνηση του παιχνιδιού το αντικείμενο ΕΠ θα πρέπει να δημιουργεί τα αντικείμενα παράθυρο και ζωύφιο.
  - Κάθε φορά που ο χρήστης κτυπά το ποντίκι σε κάποιο σημείο στην οθόνη το αντικείμενο Ποντίκι θα πρέπει να στέλνει μήνυμα στο αντικείμενο ΕΠ ενημερώνοντας σχετικά με τη θέση του κτυπήματος. Με τη σειρά του ο ΕΠ θα στέλνει μήνυμα στο αντικείμενο Ζωύφιο για να μάθει κατά πόσο το ζωύφιο έχει κτυπηθεί. Αν όχι ο ΕΠ θα πρέπει να τερματίσει το παιχνίδι. Αν ναι, θα πρέπει να σταλεί μήνυμα στο Ζωύφιο για εκτέλεση της συμπεριφοράς Hit και στη συνέχεια, αν η δύναμη του Ζωύφιο έχει φτάσει το 0, ο ΕΠ θα πρέπει να σκοτώσει το ζωύφιο και να δημιουργήσει ένα γρηγορότερο τοποθετώντας το στο παράθυρο του παιχνιδιού.

Bug Hunt - Αντικείμενα

Παρατηρούμε ότι …

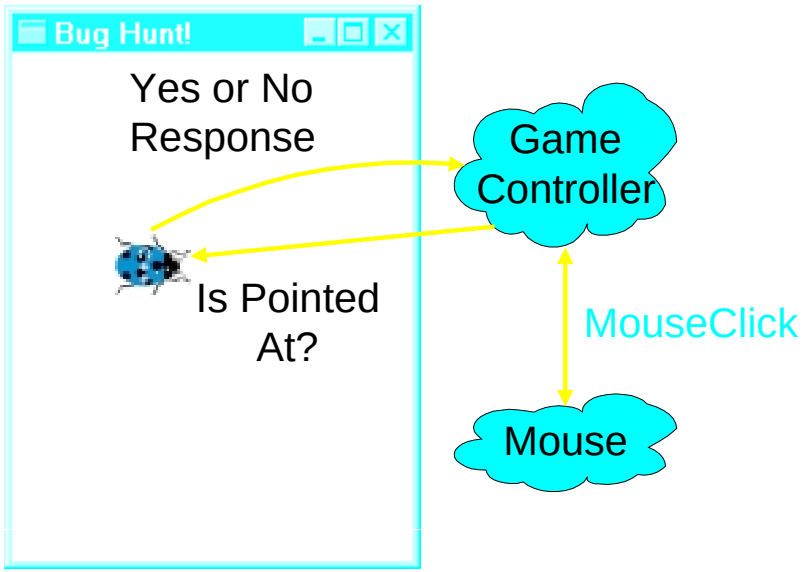
- Κάθε τι στο πρόγραμμά μας είναι αντικείμενο.
- Τα αντικείμενα εκτελούν υπολογισμούς εγείροντας μεταξύ τους αιτήματα μέσω μηνύματων.
- Ο αποδέκτης ενός μηνύματος είναι το αντικείμενο εκείνο που θα το διεκπεραιώσει, πιθανόν εγείροντας με τη σειρά του αιτήματα σε άλλα αντικείμενα.
- Τα αντικείμενα συνεργάζονται μεταξύ τους και η σωστή εξέλιξη του προγράμματος εξαρτάται από την καλή συνεργασία των αντίκειμένων.



Bug Hunt - Μηνύματα

- Οι ενέργειες προκύπτουν ως αποτελέσματα αιτημάτων για δράση.
- Το μήνυμα κωδικοποιεί την απαίτηση για την εκτέλεση κάποιας ενέργειας παρέχοντας οποιεσδήποτε πληροφορίες που τυχόν χρειάζονται για διεκπεραίωση της ενέργειας.
- Ο αποδέκτης ενός μηνύματος ανταποκρίνεται σε αυτό με την πραγματοποίηση της κατάλληλης ενέργειας (μεθόδου) και επιστρέφοντας κάποια τιμή στον αποστολέα του μηνύματος.
- Ο αποστολέας του μηνύματος δεν χρειάζεται να ξέρει λεπτομέρειες για τη μέθοδο που θα εκτελέσει ο παραλήπτης.

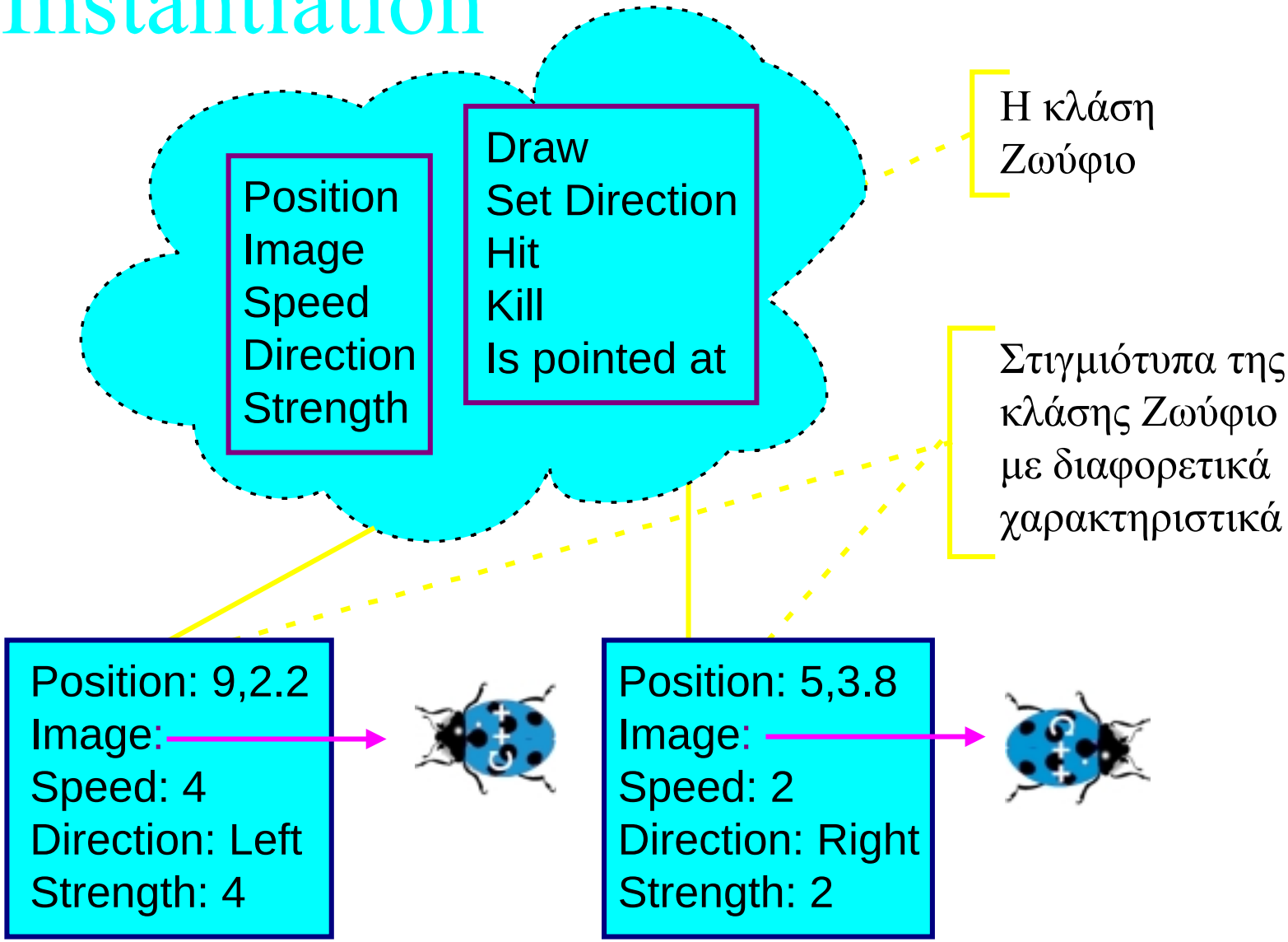
ενθυλάκωση



Bug Hunt -Κλάσεις

- Η συμπεριφορά που αναμένουμε από ένα αργό και ένα γρήγορο ζωύφιο καθορίζεται από τη γενική ιδέα που έχουμε σχετικά με τα ζωύφια του παιχνιδιού, όπως περιγράφετε στη διαφάνεια 12.
- Έτσι ένα αργό ή ένα γρήγορο ζωύφιο αποτελούν κάποια εκδοχή (στιγμιότυπο) της κλάσης (κατηγορίας) Ζωυφίου.
- Μια κλάση είναι μια δομή στην οποία αποθηκεύονται τα δυνατά χαρακτηριστικά και συμπεριφορές (μεθόδοι) των αντικειμένων που ανήκουν στην κλάση.
- Όλα τα αντικείμενα είναι στιγμιότυπα κάποιας κλάσης.
- Οι δυνατές συμπεριφορές ενός αντικειμένου προκαθορίζονται πλήρως από την κλάση στην οποία ανήκει.

Bug Instantiation



Κληρονομικότητα

- Κληρονομικότητα είναι η βασική αρχή του αντικειμενοστρεφή προγραμματισμού σύμφωνα με την οποία τα χαρακτηριστικά και συμπεριφορές μιας γενικότερης κλάσης ισχύουν και εφαρμόζονται (κληρονομούνται) σε ειδικότερες κλάσεις.

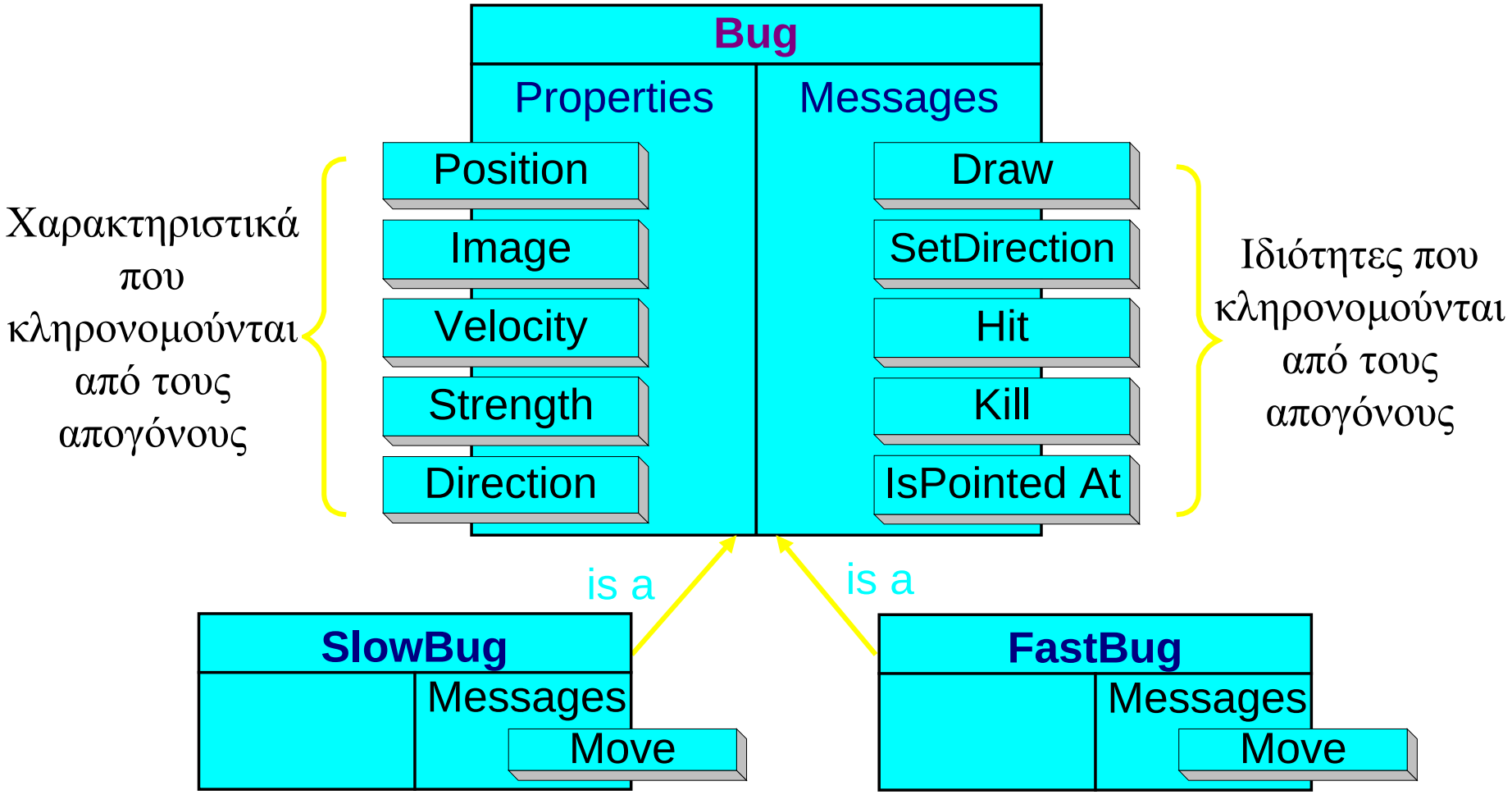
  Ιεραρχικότητα

- Παράδειγμα:

  Έστω ότι θέλουμε να ορίσουμε δύο διαφορετικά είδη ζωυφίων, το Αργό Ζωύφιο και το Γρήγορο Ζωύφιο που διαφέρουν ως προς τις συμπεριφορές τους ως εξής: Το ΑΖ, σε περίπτωση που κατά την κίνησή του κτυπήσει τα σύνορα του παραθύρου, αμέσως αντιστρέφει την κατεύθυνση κίνησης του (μέθοδος Move ΑΖ), ενώ το ΓΖ, διαπερνά το σύνορο και ξαναεμφανίζεται στην απέναντι πλευρά του παραθύρου (μέθοδος Move ΓΖ).

  Τότε μπορούμε να δημιουργήσουμε δύο κλάσεις FastBug και SlowBug οι οποίες κληρονομούν από τη γενική κλάση Bug τα χαρακτηριστικά και συμπεριφορές της ενώ επιπρόσθετα ορίζουν η κάθε μια την δική της μέθοδο κίνησης (υποκλάσεις - υπερκλάσεις) :

Κληρονομικότητα



Εισαγωγή στη C++

- Η γλώσσα C++ μετεξελίχθηκε από τη γλώσσα προγραμματισμού C.
- Οι δύο γλώσσες έχουν πολλά κοινά όπως πολλές εντολές ροής (if/else, while, for, etc).
- Η C++ όμως επεκτείνει την C με στοιχεία αντικειμενοστρεφή προγραμματισμού.

Κλάσεις

- Οι κλάσεις στη C++ είναι ένας νέος τύπος περιγραφής δεδομένων και λειτουργιών που μπορεί να ορίσει ο προγραμματιστής ή να χρησιμοποιήσει έτοιμες από ήδη υπάρχουσες στις βιβλιοθήκες της C++.

- Στη C η βασική μονάδα προγραμματισμού είναι η συνάρτηση. Στη C++ η βασική μονάδα προγραμματισμού είναι η κλάση.

- Κάθε κλάση περιέχει:
  - Χαρακτηριστικά/Attributes τα οποία θεωρούμε ότι είναι το τμήμα των δεδομένων της κλάσης (data members).
  - Συμπεριφορές/Behaviors: οι οποίες αποτελούν το συναρτησιακό τμήμα της κλάσης (member functions).

Παράδειγμα

```
class Time{
   public:
   //πρωτότυπα συναρτήσεων της κλάσης
        Time();                          //κατασκευαστής της κλάσης
        void setTime(int, int, int);
        void printTime();
   private:
        int hour, minute, second;
};
```

Παράδειγμα (συν.)

- Ο ορισμός μιας κλάσης ξεκινά με τη λέξη class.
- Το σώμα μιας κλάσης ορίζεται μεταξύ { }.
- Η συνάρτηση κλάσης με το ίδιο όνομα με αυτό της κλάσης ονομάζεται κατασκευαστής (constructor).
- Η συνάρτηση κατασκευαστής καλείται αυτόματα όταν ένα αντικείμενο αυτής της κλάσης δημιουργείται.
- Το τμήμα public μιας κλάσης (δεδομένα ή/και συναρτήσεις) είναι διαθέσιμο σε όλους που έχουν πρόσβαση στο αντικείμενο αυτής της κλάσης.
- Το τμήμα private μιας κλάσης (δεδομένα ή/και συναρτήσεις) είναι διαθέσιμο μόνο στις συναρτήσεις που είναι μέλη αυτής της κλάσης. Αυτό έχει σαν αποτέλεσμα την *απόκρυψη πληροφοριών* (information hiding).

Αντικείμενα

```
class Time{
   public:
   //πρωτότυπα συναρτήσεων της κλάσης
        Time();                          //κατασκευαστής της κλάσης
        void setTime(int, int, int);
        void printTime();
   private:
        int hour, minute, second;
};
...
Time t1, t2;
int i1, i2;
```

Αντικείμενα

- Τα στιγμιότυπα (instances) ενός κοινού τύπου δεδομένων (όπως τα i1 και i2 στην προηγούμενη σελίδα) ονομάζονται, από την εμπειρία μας με τη C, μεταβλητές. Οι i1 και i2 για παράδειγμα, είναι μεταβλητές τύπου int.
- Στιγμιότυπα (instances) μιας κλάσης ονομάζονται αντικείμενα (objects). Έτσι τα t1, t2 του παραδείγματος είναι αντικείμενα της κλάσης Time.
- Συνεπώς, τα αντικείμενα
  - περικλείουν (encapsulate) χαρακτηριστικά (attributes) και συμπεριφορές (behaviors),
  - έχουν την δυνατότητα απόκρυψης πληροφοριών (information hiding),
  - επικοινωνούν με το περιβάλλον τους μέσω ενός καλά ορισμένου τρόπου διασύνδεσης (interface).

Αντικείμενα

- Η συνάρτηση-μέλος (member function) μιας κλάσης που έχει το ίδιο όνομα με την κλάση ονομάζεται κατασκευαστής (constructor) και καλείται αυτόματα με τη δημιουργία ενός αντικειμένου.
- Για όλες τις συναρτήσεις-μέλη μιας κλάσης μπορούμε να χρησιμοποιήσουμε τα πρωτότυπα των συναρτήσεων και ο ορισμός του να γίνει αργότερα μετά την περιγραφή της κλάσης.
- Ο κώδικας των συναρτήσεων μπορεί να τοποθετηθεί και μέσα στην κλάση αλλά δεν προτείνεται για καλύτερη αναγνωσιμότητα και κομψότητα του παραγόμενου κώδικα.
- Τα μέλη δεδομένων (data members) δηλώνονται ακριβώς όπως δηλώνουμε απλές μεταβλητές γνωστών τύπων δεδομένων.
- Βέβαια μπορούμε να χρησιμοποιήσουμε και αντικείμενα ως μέλη-δεδομένων και τη δυνατότητα αυτή την ονομάζουμε σύνθεση (composition) αντικειμένων.

Παράδειγμα

```
class Time{
   public:
   //πρωτότυπα συναρτήσεων της κλάσης
        Time();
        void setTime(int, int, int);
        void printTime();
   private:
        int hour, minute, second;
};

Time :: Time()                    //Ορισμός κατασκευαστή της κλάσης
{hour = minute = second = 0;}
```

Παράδειγμα

```
void Time :: setTime(int h, int m, int s)
                          //Ορισμός της συνάρτησης setTime
 {
   if( h >= 0 && h < 24)
        hour = h;
   else
        hour = 0;
   if( m >= 0 && h < 59)
        minute = m;
   else
        minute = 0;
  ...
}

Time timeObject1, timeObject2;
Time *timePtr;
```

Ορισμός Αντικειμένων

- Τα αντικείμενα ορίζονται κάνοντας χρήσης του ονόματος της κλάσης από την οποία προέρχονται.
- Όπως οι μεταβλητές χρησιμοποιούνται μόνο αν έχουν προηγούμενα οριστεί έτσι και τα αντικείμενα πρέπει να ορίζονται πριν να χρησιμοποιηθούν.
- Για να προσπελάσουμε το δημόσιο τμήμα ενός αντικειμένου χρησιμοποιούμε τον τελεστή '.' :
  - `timeObject1.setTime(6, 30, 0);`
  - `timeObject2.printTime();`
- Το προσωπικό (private) τμήμα ενός αντικειμένου δεν πρέπει και δεν μπορεί να προσπελαστεί από το περιβάλλον του αντικειμένου.

Ορισμός Αντικειμένων

- Για να προσπελάσουμε ένα αντικείμενο μέσω ενός δείκτη χρησιμοποιούμε τον τελεστή -> :
  - `timePtr -> setTime(18, 25, 0);`
  - `timePtr -> printTime();`

- Ο κατασκευαστής μιας κλάσης μπορεί να υλοποιηθεί έτσι ώστε να τοποθετεί default τιμές σε μερικά ή όλα τα ορίσματά του…:

Παράδειγμα

```
class Time{
   public:
       Time(int=0, int=0, int=0);
       void setTime(int, int, int);
       void printTime();
   private:
       int hour, minute, second;
};

Time :: Time(int hr, int min, int sec)
{setTime(hr, min, sec);}
```

Default τιμές σε ορίσματα συναρτήσεων

- Συνεπώς, μπορούμε να δημιουργήσουμε αντικείμενα ως εξής με αποτελέσματα όπως φαίνονται πιο κάτω:

– `Time t1;`	**00:00:00**
– `Time t2(6);`	**06:00:00**
– `Time t3(6, 30);`	**06:30:00**
– `Time t4(6, 30, 5);`	**06:30:05**