JAVA VS C++
ΠΡΟΓΡΑΜΜΑΤΑ ΜΕ ΠΟΛΛΑ ΑΡΧΕΙΑ
ΣΥΝΘΕΣΗ ΚΑΙ ΣΥΝΑΘΡΟΙΣΗ

Java v.s. C++

C++

```
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <ctime>

using namespace std;

const int SIZE = 2;

……

void Dimension::RandomMove()
{
  if (dim == SIZE){
    dim -= rand()%2;
  }else if (dim == -SIZE){
    dim += rand()%2;
  }else{
    dim += (rand()%3)-1;
  }
}
```

Java

```
class Dimension
{
……
  public void RandomMove(){
    if (dim == CarGame.SIZE){
        dim -= r.nextInt(2);
    }else if (dim == -CarGame.SIZE){
        dim += r.nextInt(2);
    }else {
        dim += r.nextInt(3) - 1;
    }
  }
……
}

public  class CarGame
{
  public static int SIZE = 2;
  ……
}
```

Java v.s. C++

```
#include <iostream>
#include <cmath>
#include <cstdlib>
#include <ctime>

using namespace std;

const int SIZE = 2;

……

void Dimension::RandomMove()
{
  if (dim == SIZE){
    dim -= rand()%2;
  }else if (dim == -SIZE){
    dim += rand()%2;
  }else{
    dim += (rand()%3)-1;
  }
}
```

```
class Dimension
{
……
  public void RandomMove()
  {
    if (dim == CarGame.SIZE){
        dim -= r.nextInt(2);
    }else if (dim == -CarGame.SIZE){
        dim += r.nextInt(2);
    }else {
        dim += r.nextInt(3) - 1;
    }
  }
……
}


public  class CarGame
{
  public static int SIZE = 2;
  ……
}
```

Στη Java μπορούμε να ορίσουμε σταθερές ορίζοντας τις ως στατικές μεταβλητές μιας κλάσης (στο παράδειγμα μας, στατικές μεταβλητές της βασικής κλάσης). Μετά έχουμε πρόσβαση σε αυτές χωρίς να χρειάζεται να έχουμε αντικείμενο της κλάσης

Java v.s. C++

C++

```
class Position
{
private:
   int numOfDims;
   Dimension *dims;
public:
   Position(int);
   ……
   ……
};

Position::Position(int d)
{
   numOfDims = d;
   dims = new
Dimension[numOfDims];
}
….
```

Java

```
class Position
{
   private int numOfDims;
   private Dimension [] dims;

   public Position(int d)
   {
      numOfDims = d;
      dims = new Dimension[d];
      for (int i = 0; i < d; i ++){
         dims[i] = new Dimension();
      }
   }
……
}
```

Java v.s. C++

```
class Position
{
private:
  int numOfDims;
  Dimension *dims;
public:
  Position(int);
  ……
  ……
};

Position::Position(int d)
{
  numOfDims = d;
  dims = new Dimension[numOfDims];
}
….
```

```
class Position
{
    private int numOfDims;
    private Dimension [] dims;

    public Position(int d)
    {
        numOfDims = d;
        dims = new Dimension[d];
        for (int i = 0; i < d; i ++){
            dims[i] = new Dimension();
        }
    }
……

}
```

Στην Java όλες οι μεταβλητές είναι αναφορές σε αντικείμενα. Οι αναφορές αρχικοποιούνται σε null. Ο πίνακας dims είναι ένας πίνακας με αναφορές. Με το πρώτο new δίνουμε χώρο για τον πίνακα. Με το δεύτερο new δεσμεύουμε μνήμη για το κάθε στοιχείο του πίνακα.

Στην C++ ο πίνακας dims είναι ένας πίνακας από αντικείμενα. Όταν δεσμεύσουμε μνήμη για τον πίνακα τα αντικείμενα δημιουργούνται με τον default constructor

Java v.s. C++

C++

```
void Position::RandomMove()
{
  for (int i = 0; i < numOfDims; i ++){
    dims[i].RandomMove();
  }
}
```

Java

```
public void RandomMove()
{
  for (Dimension d: dims){
    d.RandomMove();
  }
}
```

Στην Java μπορούμε να διατρέξουμε ένα πίνακα χρησιμοποιώντας αυτό το συντακτικό, το οποίο υλοποιεί την foreach εντολή.

Το foreach συντακτικό μπορεί να χρησιμοποιηθεί για κάθε αντικείμενο τύπου Collection (το οποίο περιλαμβάνει αντικείμενα κλάσεως List, Set, Array, αλλά όχι Map)

Java v.s. C++

- Παράδειγμα: Δημιουυργείστε μια κλάση point για ένα δισδιάστατο σημείο, η οποία να έχει την εξής λειτουργικότητα.
  - Μπορούμε να θέσουμε (set) τιμή στο point
  - Μπορούμε να τυπώσουμε (print) την τιμή του point
  - Μπορούμε να αντιγράψουμε (clone) την τιμή ενός point σε ένα άλλο

```
#include <iostream>

using namespace std;

class point{
private:
  int x,y;
public:
  point();
  void set(int,int);
  void clone(point);
  void print();
};

point::point(){
  x = y = 0;
}

void point::set(int i, int j){
  x = i; y = j;
}

void point::clone(point other){
  other.set(x,y);
}

void point::print(){
  cout << x << " " << y << endl;
}
```

```
int main(){
  point p1, p2;
  p1.set(2,3);
  p1.clone(p2);
  p1.print();
  p2.print();
}
```

Ποια είναι η έξοδος του προγράμματος?

2 3
0 0
Το πέρασμα της παραμέτρου στην clone είναι δια τιμής. Η τιμή του p2 δεν μεταβάλλεται γιατί δημιουργείται τοπικό αντίγραφο

```
class point{
    int x = 0;
    int y = 0;

    public void set(int i, int j){
        x =i; y = j;
    }

    public void clone(point other){
        other.set(x,y);
    }

    public void print(){
        System.out.println(x+" "+y);
    }
}

public class main{
    public static void main(String [] args){
        point p1;
        point p2;
        p1.set(2,3);
        p1.clone(p2);
        p1.print(); p2.print();
    }
}
```

Ποια είναι η έξοδος του προγράμματος?

Το πρόγραμμα δεν κάνει compile επειδή p1, και p2 δεν έχουν αρχικοποιηθεί

```
class point{
    int x = 0;
    int y = 0;

    public void set(int i, int j){
        x =i; y = j;
    }

    public void clone(point other){
        other.set(x,y);
    }

    public void print(){
        System.out.println(x+" "+y);
    }
}

public class main{
    public static void main(String [] args){
        point p1 = new point();
        point p2 = new point();
        p1.set(2,3);
        p1.clone(p2);
        p1.print(); p2.print();
    }
}
```

Ποια είναι η έξοδος του προγράμματος?

2 3
2 3
Οι μεταβλητές p1 και p2 είναι αναφορές. Το πέρασμα παραμέτρων στην clone είναι δια αναφοράς

```
#include <iostream>

using namespace std;

class point{
private:
  int x,y;
public:
  point();
  void set(int,int);
  void clone(& point);
  void print();
};

point::point(){
  x = y = 0;
}

void point::set(int i, int j){
  x = i; y = j;
}

void point::clone(point &other){
  other.set(x,y);
}

void point::print(){
  cout << x << " " << y << endl;
}
```

```
int main(){
  point p1, p2;
  p1.set(2,3);
  p1.clone(p2);
  p1.print();
  p2.print();
}
```

Ποια είναι η έξοδος του προγράμματος?

2 3
2 3
Το πέρασμα παραμέτρων στην clone είναι πλέον δια αναφοράς

Java Containers

- Οι βασικοί containers στη Java είναι οι πίνακες και τα List, Set, και Map.
- Τα List, Set, και Map, είναι Interfaces.
  - Αφηρημένες κλάσεις που απλά ορίζουν τις συναρτήσεις που θα πρέπει να έχει η συγκεκριμένη υλοποίηση.
  - Υπάρχουν πολλές διαφορετικές υλοποιήσεις ανάλογα με την δομή η οποία χρησιμοποιείται
    - ArrayList, LinkedList, …
    - HashSet, SortedSet, TreeSet, …
    - HashMap, TreeMap, …
  - Τα List, Set, υλοποιούν το Interface Collection.

ΚΩΔΙΚΑΣ ΣΕ ΠΟΛΛΑ ΑΡΧΕΙΑ

ΑΝΑΚΕΦΑΛΑΙΩΣΗ ΑΠΟ ΤΗ ΔΙΑΛΕΞΗ 10

Διαχείριση μεγάλων προγραμμάτων

- Σε μεγάλα projects όπου έχουμε μεγάλη ποσότητα κώδικα και πολλαπλές κλάσεις ο κώδικας διασπάται σε πολλαπλά αρχεία.
  - Η κάθε κλάση τοποθετείται σε ξεχωριστό αρχείο.
- Για κάθε κλάση δημιουργούμε 2 αρχεία:
  - Ένα αρχείο επικεφαλίδα (.h file) που περιέχει τον ορισμό της κλάσης.
  - Ένα αρχείο κώδικα (.cpp file) που περίεχει την υλοποίηση των μεθόδων.
  - Το .cpp αρχείο κάνει include το .h αρχείο.
- Αν η κλάση A θέλει να δημιουργήσει αντικείμενα μιας άλλης κλάσης B τότε κάνουμε include το B.h αρχείο στο A.h αρχείο.

Πολλαπλοί ορισμοί

- Αν έχουμε πολλά header αρχεία που το ένα κάνει include το άλλο, τότε υπάρχει κίνδυνος σε ένα αρχείο να δηλώσουμε πολλαπλές φορές την ίδια κλάση.
  - Στην περίπτωση αυτή θα πάρουμε λάθος από τον compiler.
- Για να το αποφύγουμε τοποθετούμε μια εντολή μέσα στο .h αρχείο της κλάσης η οποία λέει στον preprocessor να μην ορίσει δύο φορές την ίδια κλάση

Εντολή προεπεξεργαστή

```
#if !defined(__CLASSNAME__)
#define __CLASSNAME__

Class ClassName
{
        …
        …
        …
}

#endif
```

Με αυτό τον τρόπο εξασφαλίζουμε ότι η κλάση θα οριστεί μόνο μία φορά. Τότε θα οριστεί και η μεταβλητή **__CLASSNAME__** οπότε την επόμενη φορά δεν θα μπούμε μέσα στο if

Το αρχείο Person.h

```
using namespace std;

#if !defined(__PERSON__)
#define __PERSON__

class Person {
private:
  int id;
  string fname;
  string lname;
public:
  Person();
  Person(int);
  void SetDetails(int id, string fn, string ln);
  bool operator == (const Person &) const;
  bool operator < (const Person &) const;
  int getId() const;
  void PrintDetails() const;
};

#endif
```

Το αρχείο Person.cpp

```
#include <iostream>
#include "Person.h"

Person::Person() {}

Person::Person(int i) { id = i; }

void Person::SetDetails(int i, string f, string l)
{ id = i;  fname = f;  lname = l; }

bool Person::operator == (const Person &p2)
{ return (id == p2.id); }

bool Person::operator < (const Person &p2)
{ return (id < p2.id); }

int Person::getId() const
{ return id;}

void Person::PrintDetails() const
{
  cout << "id: " << id
       <<" first name:" << fname << " last name:" << lname << endl;
}
```

Το αρχείο main.cpp

```
#include <iostream>
#include <set>
#include "Person.h"

int main(){
  set<Person> S;
  Person P[3];

  int id;
  string fname, lname;
  for(int i =0 ; i< 3; i ++){
    cin >>id >> fname >> lname;
    P[i].SetDetails(id, fname, lname);
    S.insert(P[i]);
  }

  cin >> id;
  Person sp(id);
  set<Person>::iterator iter = S.find(sp);
  if (iter == S.end()){
    cout << "id not found\n";
  }else{ iter->PrintDetails(); }
}
```

makefiles

- Τα makefiles ρυθμίζουν τον τρόπο με τον οποίο συνδέουμε πολλά αρχεία μεταξύ τους
- Όταν μεταφράζουμε τον πηγαίο (source) κώδικα, ο μεταγλωττιστής παράγει ένα αρχείο με τελικό (object) κώδικα.
- Ο linker συνδέει τα αρχεία τελικού κώδικα με τον τελικό κώδικα από τις βιβλιοθήκες και παράγει το εκτελέσιμο.

File: makefile

```
# Person.o DEPENDS on Person.h and Person.cpp
# We create only the .o[bject] file and do not
# link it to an executable
# main DEPENDS on the Person.o file and main.cpp
# We compile it and we produce the executable
# main.exe

main: Person.o main.cpp
    g++ Person.o -o main.exe main.cpp
Person.o: Person.h Person.cpp
    g++ -O -c Person.cpp
```

Οι γραμμές αυτές θα πρέπει να ξεκινάνε οπωσδήποτε με tab !

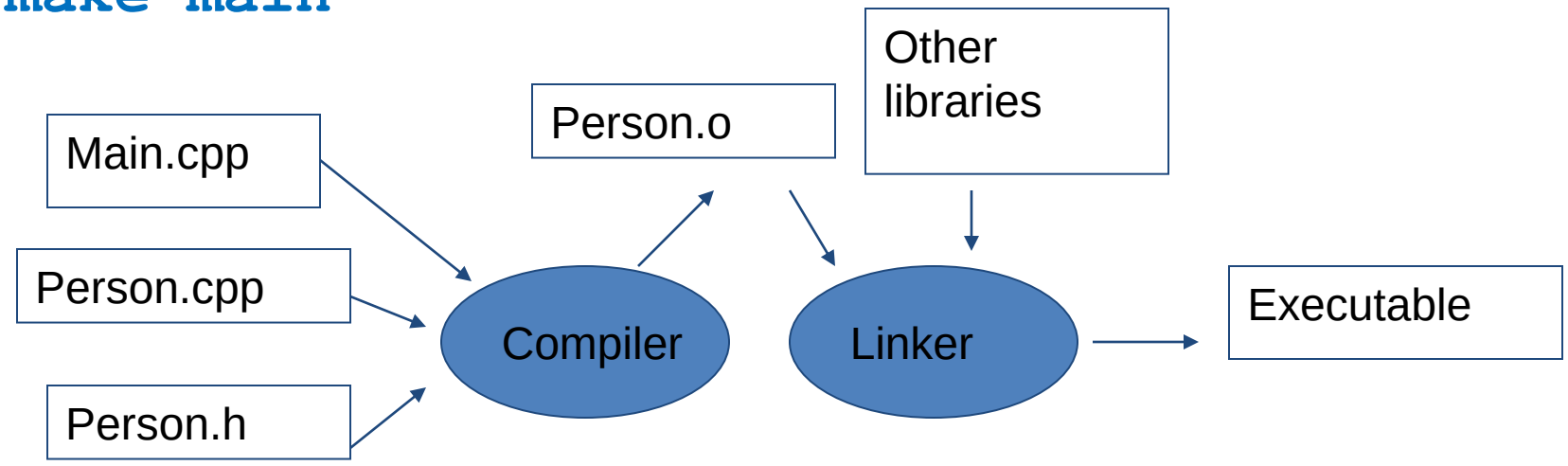
makefiles

- Τα περιεχόμενα του τρέχοντος directory είναι:

```
Person.h, Person.cpp, main.cpp, makefile
```

- Η μεταγλώττιση γίνεται ως εξής:

```
make main
```



Εναλλακτικά

- Ένας πιο απλός τρόπος για να κάνετε compile κατ' ευθείαν το πρόγραμμά σας:

```
g++ -o main.exe main.cpp Person.cpp
```

Ένα μεγαλύτερο παράδειγμα.

- Θυμηθείτε το παράδειγμα όπου έχουμε μία κλάση Array που έχει στοιχεία αντικείμενα μιας αφηρημένης κλάσης Element, και έχουμε δυο τύπους από Element: IntElement, και PointElement.
- Θα δούμε πώς θα σπάσουμε τον κώδικα σε πολλαπλά αρχεία.

Abstract Class Element

```
class Element
{
public:
   virtual bool operator < (Element &other) = 0;
   virtual void Print() = 0;
};
```

Class Array

```
class Array
{
private:
   Element **A;
   int size;
public:
   Array(int s);
   Element *& operator [](int);
   void Print();
   void Sort();
};
```

```
Array::Array(int s)
{
  size = s;
  A = new Element*[size];
}

Element *& Array::operator [] (int i)
{
  return A[i];
}

void Array::Print()
{
  for (int i = 0; i < 10; i ++){
    A[i]->Print();
  }
}

void Array::Sort()
{
  for (int i = 0; i < 10; i ++){
    for (int j = i+1; j < 10; j ++){
      if (*A[i] < *A[j]){
         Element *temp = A[i];
         A[i] = A[j];
         A[j] = temp;
      }
    }
  }
}
```

IntElement

```
class IntElement: public Element
{
private:
   int val;
public:
   IntElement(int);
   int GetVal();
   bool operator < (Element &);
   void Print();
};
```

```
IntElement::IntElement(int i)
{
  val = i;
}

int IntElement::GetVal()
{
  return val;
}


bool IntElement::operator <(Element &cOther)
{
  IntElement &other = dynamic_cast<IntElement &>(cOther);
  if (val < other.GetVal()){
    return true;
  }
  return false;
}

void IntElement::Print()
{
  cout << val << endl;
}
```

PointElement

```
class PointElement:public Element
{
private:
  point val;
public:
  PointElement(int,int);
  point GetVal();
  bool operator < (Element &);
  void Print();
};
```

```
PointElement::PointElement(int x,int y)
{
  val.x = x;
  val.y = y;
}

point PointElement::GetVal()
{
  return val;
}


bool PointElement::operator < (Element &cOther)
{
  PointElement &other = dynamic_cast<PointElement &>(cOther);
  if (val.x < other.GetVal().x){
    return true;
  }else if (val.x == other.GetVal().x){
    if (val.y < other.GetVal().y ){
      return true;
    }else {
      return false;
    }
  }else {
    return false;
  }
}

void PointElement::Print()
{
  cout << val.x << " " << val.y << endl;
}
```

main

```
int main()
{
  srand(time(NULL));
  Array iA(10);
  Array pA(10);

  for (int i = 0; i < 10; i ++){
    iA[i] = new IntElement(rand()%10);
    pA[i] = new PointElement(rand()%10, rand()%10);
  }

  iA.Print(); pA.Print();
  iA.Sort(); pA.Sort();
  iA.Print(); pA.Print();
}
```

Utils.h

Ένα αρχείο με διάφορες βιβλιοθήκες που χρειάζονται οι περισσότερες κλάσεις

```
#include <iostream>
#include <cstring>
#include <stdio.h>
#include <cmath>
#include <ctime>
#include <cstdlib>

using namespace std;
```

Element.h

```
#if !defined(__ELEMENT__)
#define __ELEMENT__

class Element
{
public:
  virtual bool operator < (Element &other) = 0;
  virtual void Print() = 0;
};

#endif
```

Για την κλάση Element δεν χρειάζεται να ορίσουμε .cpp αρχείο.

Array.h

```
#if !defined(__ARRAY__)
#define __ARRAY__

#include "Utils.h"
#include "Element.h"

class Array
{
private:
  Element **A;
  int size;
public:
  Array(int s);
  Element *& operator [](int);
  void Sort();
  void Print();
};

#endif
```

Array.cpp

```
#include "Array.h"

Array::Array(int s)
{
  size = s;
  A = new Element*[size];
}

Element *& Array::operator [] (int i)
{ return A[i]; }

void Array::Sort()
{
  for (int i = 0; i < 10; i ++)
    for (int j = i+1; j < 10; j ++)
      if (*A[i] < *A[j]){
        Element *temp = A[i]; A[i] = A[j]; A[j] = temp;
      }
}

void Array::Print()
{
  for (int i = 0; i < 10; i ++) { A[i]->Print();}
}
```

IntElement.h

```
#if !defined(__INT_ELEMENT__)
#define __INT_ELEMENT__

#include "Utils.h"
#include "Element.h"

class IntElement: public Element
{
private:
   int val;
public:
   IntElement(int);
   int GetVal();
   bool operator < (Element &);
   void Print();
};

#endif
```

IntElement.cpp

```
#include "IntElement.h"

IntElement::IntElement(int i)
{ val = i; }

int IntElement::GetVal()
{ return val; }


bool IntElement::operator <(Element &cOther)
{
  IntElement &other = dynamic_cast<IntElement &>(cOther);
  if (val < other.GetVal()){
    return true;
  }
  return false;
}

void IntElement::Print()
{ cout << val << endl; }
```

PointElement.h

```
#if !defined(__POINT_ELEMENT__)
#define __POINT_ELEMENT__

#include "Utils.h"
#include "Element.h"

struct point
{
  int x;
  int y;
};

class PointElement: public Element
{
private:
  point val;
public:
  PointElement(int,int);
  point GetVal();
  bool operator < (Element &);
  void Print();
};

#endif
```

PointElement.cpp

```
#include "PointElement.h"

PointElement::PointElement(int x,int y)
{ val.x = x; val.y = y;}

point PointElement::GetVal()
{ return val; }


bool PointElement::operator <(Element &cOther)
{
  PointElement &other = dynamic_cast<PointElement &>(cOther);
  if (val.x < other.GetVal().x){ return true; }
  if (val.x == other.GetVal().x){
    if (val.y < other.GetVal().y ){
      return true;
    }else { return false; }
  }else { return false; }
}

void PointElement::Print()
{
  cout << val.x << " " << val.y << endl;
}
```

main.cpp

```
#include "Utils.h"
#include "Array.h"
#include "IntElement.h"
#include "PointElement.h"

int main()
{
  srand(time(NULL));
  Array iA(10);
  Array pA(10);

  for (int i = 0; i < 10; i ++){
    iA[i] = new IntElement(rand()%10);
    pA[i] = new PointElement(rand()%10, rand()%10);
  }

  iA.Print(); pA.Print();
  iA.Sort(); pA.Sort();
  iA.Print(); pA.Print();
}
```

makefile

```
main: main.cpp Array.o IntElement.o PointElement.o Utils.h
	g++ -o main.exe Array.o IntElement.o PointElement.o main.cpp
Array.o: Array.h Array.cpp Element.h Utils.h
	g++ -c Array.cpp
IntElement.o: IntElement.h IntElement.cpp Element.h Utils.h
	g++ -c IntElement.cpp
PointElement.o: PointElement.h PointElement.cpp Element.h Utils.h
	g++ -c PointElement.cpp
```

ΣΥΝΘΕΣΗ
ΣΥΝΑΘΡΟΙΣΗ

Συνδυασμός κλάσεων/αντικειμένων

- Οι κλάσεις μας επιτρέπουν να ορίζουμε νέους τύπους δεδομένων. Αυτούς τους νέους τύπους θέλουμε να μπορούμε να τους χρησιμοποιούμε και στον ορισμό άλλων κλάσεων και αντικειμένων.
  - Ένα Car θα έχει ένα Position
  - Ένα Department θα έχει ένα σύνολο από Employees
  - Ένας Player μπορεί να κρατάει ένα Weapon
  - Ένα House έχει Doors και Windows

Συνδυασμός κλάσεων/αντικειμένων

- Ο συνδυασμό κλάσεων και αντικειμένων γίνεται συνήθως με δείκτες.
  - Πολλά διαφορετικά αντικείμενα μπορεί να έχουν δείκτη στο ίδιο αντικείμενο.
- Όταν σχεδιάζουμε τον κώδικα είναι σημαντικό να έχουμε σκεφτεί πως θα γίνει ο συνδυασμός των αντικειμένων. Θα πρέπει να ξέρουμε ποιος έχει την ιδιοκτησία του αντικειμένου.
  - Ποιος δημιουργεί το αντικείμενο
  - Ποιος καταστρέφει το αντικείμενο όταν δεν χρειάζεται πλέον (αποφεύγουμε memory leaks)
  - Ποιος φροντίζει ώστε όλοι οι δείκτες να είναι ενημερωμένοι (αποφεύγουμε πρόσβαση σε μνήμη που έχει αποδεσμευτεί).

Σύνθεση, Συνάθροιση

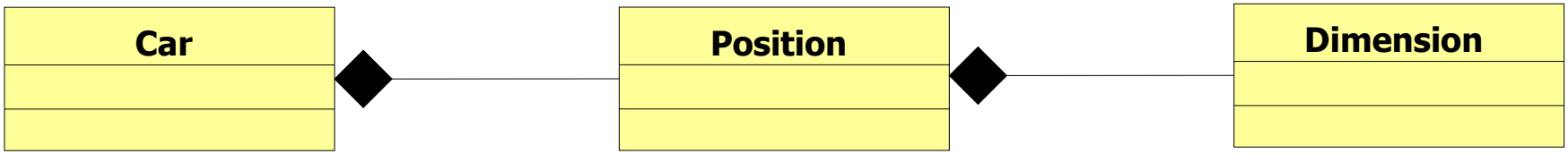
- Η σύνθεση και η συνάθροιση είναι έννοιες που μας βοηθάνε στο να ελέγχουμε καλύτερα πως διαχειριζόμαστε και συνδυάζουμε αντικείμενα.
- Σύνθεση: Μία κλάση Α έχει σχέση σύνθεσης με μία κλάση Β, όταν η κλάση Α δημιουργεί και καταστρέφει τα αντικείμενα της κλάσης Β.
- Συνάθροιση: Μία κλάση Α έχει σχέση συνάθροισης με μία κλάση Β, όταν η κλάση Α χρησιμοποιεί υπάρχοντα αντικείμενα της κλάσης Β.

Σύνθεση, Συνάθροιση

- Η σχέση σύνθεσης ή συνάθροισης είναι σχεδιαστική επιλογή του προγραμματιστή
  - Δεν υπάρχει απόλυτη αλήθεια στο ποια σχέση είναι η «σωστή». Υπάρχει όμως καλός και κακός σχεδιασμός
  - Πολλές φορές και οι δύο επιλογές είναι πιθανές και εξαρτάται από την συγκεκριμένη εφαρμογή
    - Για την υλοποίηση ενός παιχνιδιού ένα House έχει σχέση σύνθεσης με Door και Window
    - Για την υλοποίηση ενός προγράμματος διαχείρισης κατασκευαστικών υλικών ένα House έχει σχέση συνάθροισης με Door και Window

Παράδειγμα σύνθεσης

- Στο παιχνίδι με το όχημα που κινείται σε ένα πολυδιάστατο χώρο, οι σχέσεις μεταξύ των κλάσεων είναι σχέσεις σύνθεσης.



- Θα μπορούσαμε να κάνουμε το ίδιο πρόγραμμα με συνάθροιση?

Παράδειγμα συνάθροισης

- Υπάρχει ένα σύνολο από Positions και ο κάθε Car έχει ένα δείκτη στο Position στο οποίο βρίσκεται εκείνη τη στιγμή.

- Μια τέτοια υλοποίηση μπορεί να βολεύει περισσότερο στην περίπτωση που το Position έχει και αυτό συνολικές ιδιότητες που δεν εξαρτώνται από το Car.
  - Π.χ. υπάρχει ένα Weapon

Συνάθροιση

- Έχουμε ένα πρόγραμμα διαχείρισης ενός πανεπιστημίου.
  - Υπάρχει μια κλάση Professor που κρατάει πληροφορίες για τους καθηγητές
    - Για κάθε καθηγητή κρατάμε και ένα vector των μαθημάτων που διδάσκει
  - Υπάρχει μια κλάση Student που κρατάει πληροφορίες για τους φοιτητές
    - Για κάθε φοιτητή κρατάμε και ένα vector των μαθημάτων που παίρνει
  - Υπάρχει μια κλάση Course που κρατάει πληροφορίες για το μάθημα
    - Για κάθε μάθημα κρατάμε ένα δεικτη στον Professor που διδάσκει το μάθημα, και ένα vector με τους φοιτητές που παίρνουν το μάθημα.
  - Μία κλάση Department κρατάει μια λίστα με τα μαθήματα που δίνονται το συγκεκριμένο εξάμηνο.
- Τι σχέσεις έχουμε μεταξύ των διαφορετικών κλάσεων?
- Τι γίνεται όταν στο τέλος του εξαμήνου τελειώσει το μάθημα?

Σύνθεση/Συνάθροιση v.s. Κληρονομικότητα

- Οι έννοιες της σύνθεσης και της συνάθροισης συχνά χρησιμοποιούνται για να ξεχωρίσουμε από την περίπτωση της κληρονομικότητας.
  - Και πάλι είναι μια σχεδιαστική επιλογή αν θα χρησιμοποιηθεί κληρονομικότητα ή συνθεση/συνάθροιση.
- Στο βιβλίο της Java σύνθεση και συνάθροιση είναι ένα πράγμα και ξεχωρίζονται από την κληρονομικότητα.
  - Εν μέρει γιατί στην Java το πρόβλημα της αποδέσμευσης της μνήμης δεν είναι τόσο μεγάλο.
  - Σε σχεδιαστικό επίπεδο όμως παραμένουν διαφορετικά.

ΑΠΟΧΑΙΡΕΤΙΣΜΟΣ

Εξετάσεις

- Στο τέλος αυτής της βδομάδας θα είναι διαθέσιμες οι τυπωμένες οι διαφάνειες του μαθήματος μέχρι τα Χριστούγεννα.
- Για τις εξετάσεις, διαβάστε προσεκτικά τις σημειώσεις και ξαναδείτε τις ασκήσεις
  - Περισσότερη έμφαση στις έννοιες και όχι τόσο το συντακτικό.
- Ευχαριστώ για τη συμμετοχή και το feedback.
- Καλή επιτυχία!

Εξόρυξη δεδομένων

- Μάθημα επιλογής του επόμενου εξαμήνου
- Συνδυάζει προγραμματισμό, σχεδίαση αλγορίθμων, μαθηματικά
- Θα μάθετε:
  - τι σχέση έχουν οι μπύρες με τις πάνες
  - πως δουλεύει ο αλγόριθμος PageRank του Google
  - πως φτιάξετε ένα πρόγραμμα που ξεχωρίζει τα spam emails