Προγραμματισμός II (Java)

7. Συλλογές δεδομένων

Συλλογές και ενέργειες

- ArrayList
  - Αναζήτηση συγκεκριμένου στοιχείου
  - Αναζήτηση ελαχίστου/μεγίστου
  - Συνάθροιση
- Πίνακας
  - Εισαγωγή
  - Εισαγωγή με έλεγχο διπλοτύπων

Παράδειγμα

```
ArrayList<Department> allDeps = new ArrayList<Department>();
allDeps.add(new Department(101,”Informatics”,40));
allDeps.add(new Department(102,”Geography”,100));
allDeps.add(new Department(105,”Physics”,200));
allDeps.add(new Department(103,”Dietetics”,80));
allDeps.add(new Department(104,”Ecology”,80));

Department[] allDepsArray=new Department[10];
allDepsArray[0]=new Department(101,”Informatics”,40);
allDepsArray[1]=new Department(102,”Geography”,100);
allDepsArray[5]=new Department(105,”Physics”,200);
allDepsArray[3]=new Department(103,”Dietetics”,80);
allDepsArray[4]=new Department(104,”Ecology”,80);
```

Πώς διαγράφω το τμήμα Physics; →

Διαγραφή μέσω αναζήτησης (ArrayList)

- Ψάχνω να βρω σε ποια θέση υπάρχει το τμήμα Physics. Αναζήτηση

```
for (int i=0;i<allDeps.size();i++){
    if (allDeps.get(i).getName().equals("Physics")){
            allDeps.remove(i); //και διαγραφή
            break;          //σταματώ την επανάληψη
    }
}
```

Η χρήση ArrayList δεν αφήνει κενές θέσεις (null) αλλά μεταθέτει τα αντικείμενα

Διαγραφή μέσω αναζήτησης (Array)

- Ψάχνω να βρω σε ποια θέση υπάρχει το τμήμα Physics. Αναζήτηση

```
for (int i=0;i<allDepsArray.length;i++){
    if (    allDepsArray[i]!=null &&
            allDepsArray[i].getName().equals("Physics")) {
            allDepsArray[i]=null; //και διαγραφή
            break;          //σταματώ την επανάληψη
    }
}
```

Η χρήση πίνακα αφήνει κενές θέσεις (null)

Αναζήτηση μεγίστου/ελαχίστου

```
int maxnum= Integer.MIN_VALUE;
for (int i=0;i<allDeps.size();i++){
    if (allDeps.get(i).getNumStudents()>maxnum){
        maxnum=allDeps.get(i).getNumStudents();
    }
}
---------------------------------------------------------------------------
int minnum= Integer.MAX_VALUE;
for (int i=0;i<allDepsArray.length;i++){
    if (allDepsArray[i]!=null && allDepsArray[i].getNumStudents()>minnum){
        minnum=allDepsArray[i].getNumStudents();
    }
}
```

Εισαγωγή σε πίνακα

```
boolean inserted=false;
for (int i=0;i<allDepsArray.length;i++){
    if ( allDepsArray[i]==null) //αν υπάρχει κενό
            allDepsArray[i]=new Department(106,”Maths”,200);
            inserted=true;
            break;                  //σταματώ την επανάληψη
    }
}
```

Εισαγωγή σε ArrayList

```
allDeps.add(new Department(106,”Maths”,200));
```

Εισαγωγή με έλεγχο διπλοτύπων

- Η εισαγωγή με έλεγχο διπλοτύπων θα γίνει σε δύο φάσεις:
  - Αναζήτηση για το αν υπάρχει ή όχι το αντικείμενο στον πίνακα ή τη λίστα
  - Εισαγωγή – στο τέλος ή σε κενή θέση - σε περίπτωση που δεν υπάρχει
- Υπάρχουν πιο γρήγορες λύσεις;
- Η ArrayList διαθέτει μεθόδους:
  - int indexOf(Object elem): επιστρέφει τη θέση πρώτης εμφάνισης του elem στη λίστα
  - int lastIndexOf(Object elem)
  - boolean contains(Object elem) : επιστρέφει true αν η λίστα περιέχει το elem
  - Object remove(int index) : διαγράφει το στοιχείο στη θέση index της λίστας και μας το επιστρέφει

Σύγκριση: Η μέθοδος equals

- Για να δουλέψουν οι προηγούμενοι μέθοδοι για λίστες με αντικείμενα δικών μας κλάσεων πρέπει στις κλάσεις μας να έχουμε μια μέθοδο equals π.χ.

```
public boolean equals(Object o){
    Department d=(Department)o; // πιθανό να παράγει
                                                    //ClassCastException
    if (this.id==d.getId() && this.name.equals(d.getName()) &&
                    this.numStudents==d.getNumStudents())
        return true;
    else
        return false;
}
```

Ταξινόμηση

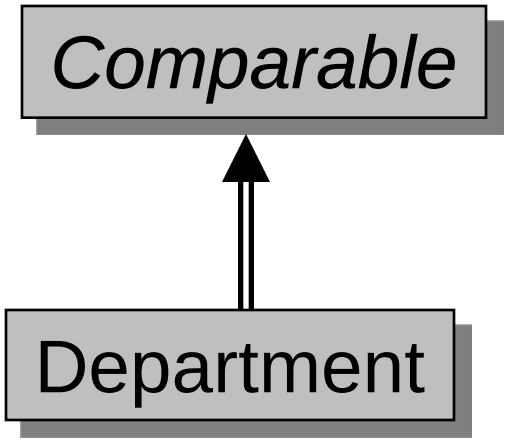
- Με ποιο τρόπο μπορώ να ταξινομήσω τα στοιχεία ενός πίνακα ή μιας λίστας; BubbleSort:

```
public void bubbleSort(int[] unsortedArray, int length) {
        int temp, counter, index;
       for(counter=0; counter<length-1; counter++) {
            for(index=0; index<length-1-counter; index++) {
                if(unsortedArray[index] > unsortedArray[index+1]) {
                    temp = unsortedArray[index];
                    unsortedArray[index] = unsortedArray[index+1];
                    unsortedArray[index+1] = temp;
                }
            }
        }
    }
```

Διάταξη

- Η διάταξη στους ακεραίους είναι δεδομένη
- Τι γίνεται όμως με τις δικές μας κλάσεις;
- Πώς μπορούμε να ορίσουμε διάταξη στα αντικείμενά τους;

```
public interface Comparable
{
   int compareTo(Object o);
}
```



```
public class Department implements Comparator{
    …
     public int compareTo(Object o){
         Department d=(Department)o; // πιθανό να παράγει
                                                    //ClassCastException
         if (this.numStudents>d.getNumStudents())
           return 1;
         else if (this.numStudents<d.getNumStudents())
           return -1;
         else
           return 0;
     }
}
```

Ταξινόμηση

Collections.sort(allDeps);

Ταξινομεί τα τμήματα με βάση τον αριθμό σπουδαστών που έχουν

Χρησιμοποιεί την QuickSort

Δομή δεδομένων σε Java

- Μπορώ να υλοποιήσω μια δομή δεδομένων σε Java; π.χ. Queue

```
public class DepQueue{
private int N;        // number of elements on queue
private Node first; // beginning of queue
Node last;            // end of queue
private class Node {
    private Department dep;
    private Node next;
}
```

```
public DepQueue() {
    first = null;
    last = null;
}
public void enqueue(Department dep) {
    Node x = new Node();
    x.dep = dep;
    if (isEmpty()) { first = x; last = x; }
    else { last.next = x; last = x; }
    N++;
}
public Item dequeue() {
    if (isEmpty())
        throw new RuntimeException("Queue is empty");
    Department dep = first.dep;
    first = first.next;
    N--;
    return dep;
}
```

```
public boolean isEmpty() { return first == null; }
public int length() { return N; }
public int size() { return N; }
```

Πλαίσιο συλλογών

Collections Framework

Πλαίσιο γενικότερα

- Ως πλαίσιο ορίζεται το σύνολο των τάξεων που καθορίζει το σχεδιασμό λύσης για μια οικογένεια προβλημάτων
- Περιλαμβάνει κάποιες βασικές τάξεις αλλά μπορεί να επεκταθεί
- Καθορίζει την αρχιτεκτονική μιας εφαρμογής: τις τάξεις, τα αντικείμενα, τις μεταξύ τους συνεργασίες και τον έλεγχο των νηματικών διεργασιών
- Ταυτόχρονα προσφέρει και έτοιμες υλοποιήσεις που μπορούμε να χρησιμοποιήσουμε

Πλαίσιο διαχείρισης συλλογών

- Το πρόβλημα είναι η διαχείριση συλλογών αντικειμένων
- Το πλαίσιο καθορίζει τις τάξεις για τους διάφορους τύπους συλλογών και τις μεθόδους που απαιτεί η διαχείριση μιας συλλογής
- Γενικότερα μια ***συλλογή*** (***collection***) είναι ένα αντικείμενο που περιλαμβάνει περισσότερα αντικείμενα
- Σε μια ***συλλογή*** θέλουμε να ***αποθηκεύουμε***, να ***ανακτούμε*** και να ***διαχειριζόμαστε*** αντικείμενα.

Συγκεκριμένα

- Να προσθέτουμε ένα αντικείμενο
- Να βγάζουμε ένα αντικείμενο
- Να εντοπίζουμε ένα αντικείμενο
  - Αν υπάρχει συλλογή
  - Πόσες φορές
  - Αν κάποιο ισοδύναμό του υπάρχει στη συλλογή
- Να δίνουμε κάποια τιμή κλειδί και να ανακτούμε το αντικείμενο
- Να διασχίζουμε όλη τη συλλογή
- Πώς υλοποιούνται όλα αυτά;

Java Collections Framework

- Το πλαίσιο συλλογών της Java περιλαμβάνει βασικές συλλογές αντικειμένων (object containers)
  - ***Διεπαφές***: αφηρημένοι τύποι που αντιστοιχούν σε τύπους συλλογών (interfaces και abstract classes)
  - ***Υλοποιήσεις***: των προηγούμενων διεπαφών
  - ***Αλγορίθμους***: μεθόδους που κάνουν συγκεκριμένες λειτουργίες όπως αναζήτηση, ταξινόμηση κλπ σε αντικείμενα της συλλογής

Διεπαφές και υλοποιήσεις

- Βασικές διεπαφές: Collection, Set, List, Map, SortedSet, SortedMap
- Λειτουργικές διεπαφές: Comparator, Iterator
- Λειτουργικές τάξεις: Collections, Arrays
- Τύποι συλλογών:

	Με σειρά	Χωρίς σειρά
Επιτρέπει διπλότυπα	List	Multiset (Bag)
Δεν επιτρέπει	Hashtable	Set

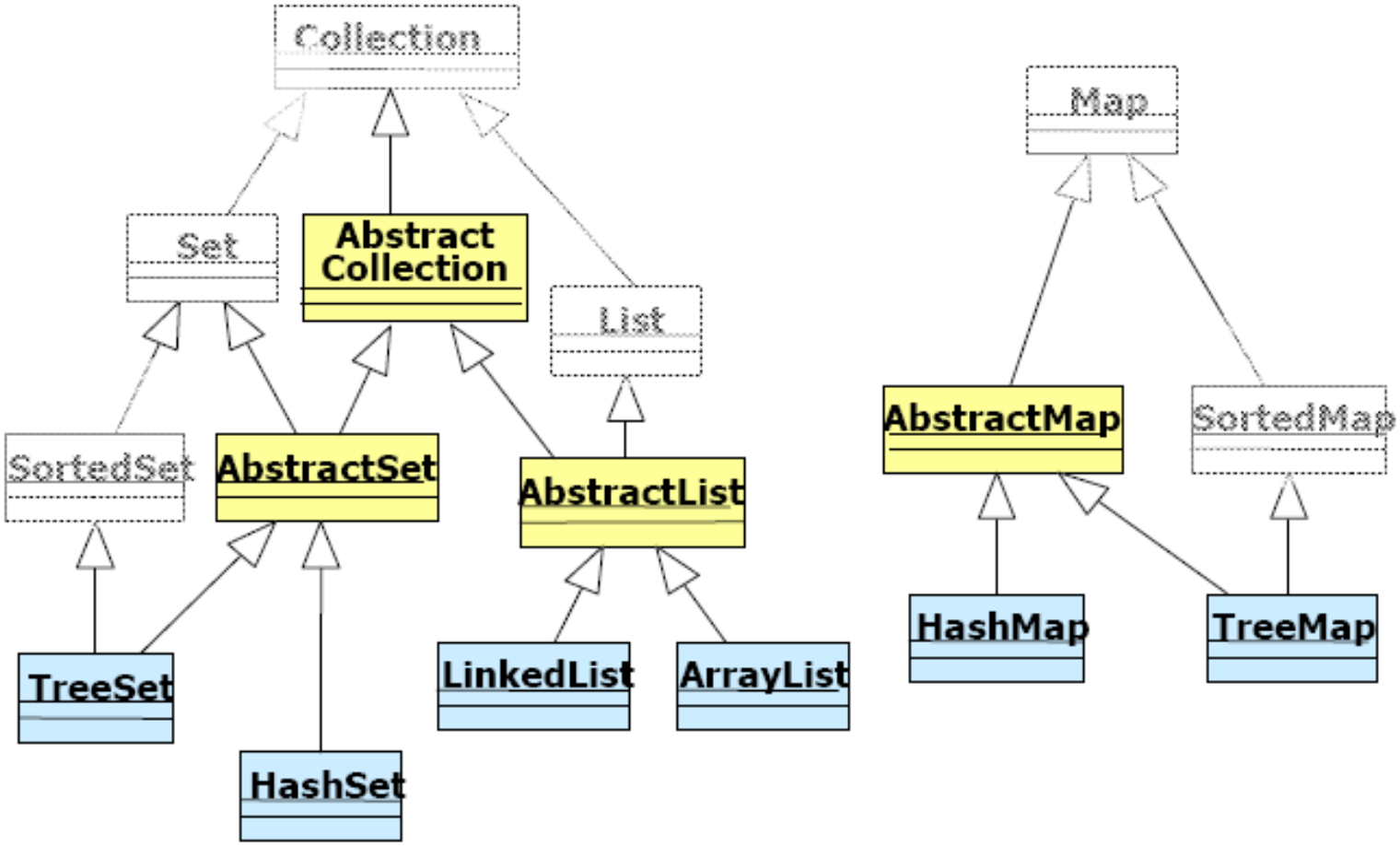
Βασικές διεπαφές

- Το Collection είναι η βασικότερη συλλογή, επιτρέπει διπλότυπα, δεν έχει σειρά
  - Σε μια συλλογή μπορούμε να προσθέσουμε αφαιρέσουμε στοιχεία να τη διασχίσουμε και να αναζητήσουμε στοιχεία.
- Το Set δεν επιτρέπει διπλότυπα
  - Είναι απόγονος του Collection και μπορεί να επεκταθεί για να κάνει ταξινόμηση
- Το List επιτρέπει διπλότυπα αλλά διατηρεί σειρά

Συλλογές με ευρετήρια

- Το Map επιτρέπει την εισαγωγή στοιχείων με ταυτόχρονη εισαγωγή ενός κλειδιού.
  - Το ζεύγος key-value συνδέει το σύνολο των κλειδιών με τη συλλογή των τιμών.
  - Αυτό σημαίνει ότι τα κλειδιά δεν επιτρέπουν διπλοεγγραφές ενώ οι τιμές επιτρέπουν
  - Η διάσχιση του map γίνεται με διάσχιση του key-set και ανάκτηση κάθε value.

Βασικές υλοποιήσεις



Set και List

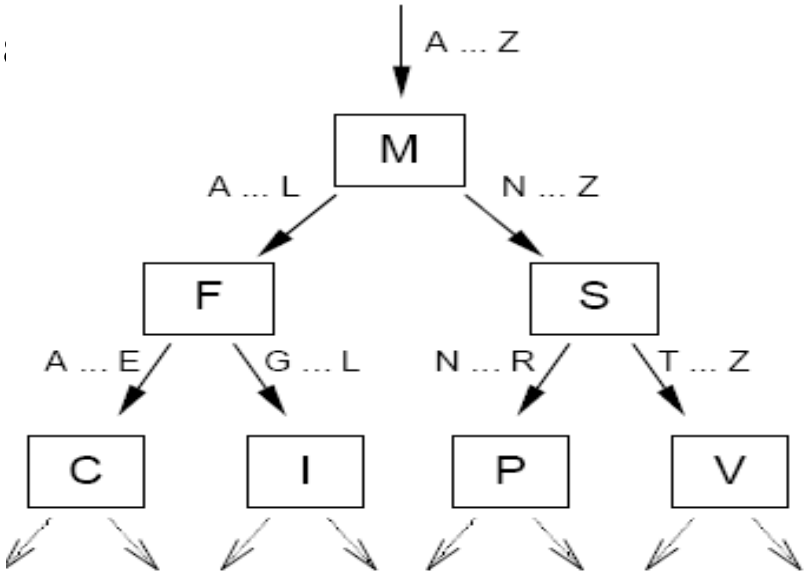
- Ένα Set
  - δεν περιέχει διπλότυπα: π.χ. e1.equals(e2)
  - περιέχει το πολύ ένα στοιχείο null
  - Έχει ένα Iterator για τη διάσχιση
  - π.χ. Τα γράμματα ‘A’ ως ‘Z’, το {} κλπ.
- Μια List
  - Τα στοιχεία της είναι σε διάταξη
  - Επιτρέπει διπλότυπα
  - Και συνεπώς πολλά null αντικείμενα
  - Έχει μεθόδους get και set με δείκτη θέσης

ArrayList και LinkedList

- Μια ArrayList
  - Περικλείει στην ουσία ένα array
  - Υλοποιεί τα πάντα μέσω array γι’ αυτό και μια εισαγωγή στην αρχή σημαίνει μετακίνηση όλων των περιεχομένων της λίστας
  - Προσφέρει iterator για τη διάσχιση και μεθόδους get και set με δείκτη θέσης
- Μια LinkedList
  - Είναι μια διπλά συνδεδεμένη λίστα
  - Προσφέρει υλοποιήσεις remove και insert πιο γρήγορες από την ArrayList
  - Έχει πιο αργή διάσχιση (τα στοιχεία δεν είναι σε συνεχόμενες θέσεις μνήμης) και υλοποίηση της get.

Δέντρα

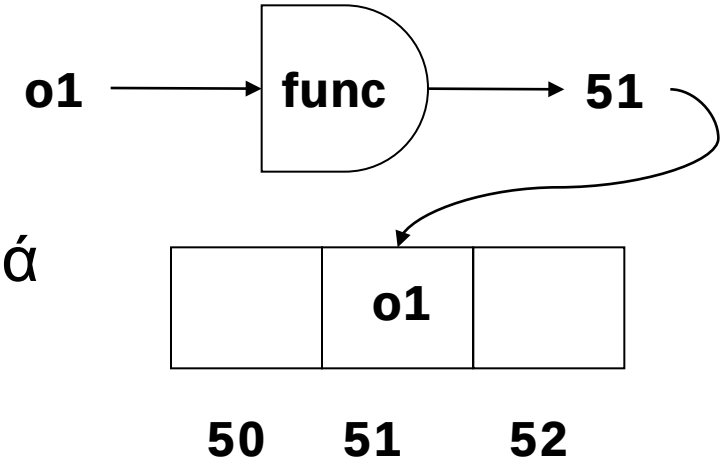
- Τα Trees προσφέρουν οργάνωση αντικειμένων σε δύο διαστάσεις
  - TreeSet: Δυαδικά δέντρα αναζήτησης - δυαδικά δέντρα με σειρά στα στοιχεία τους
  - TreeMap: υλοποίηση του Map μ
    TreeSet
- Γρήγορη διάσχιση και αναζήτηση
- Αργά remove/insert γιατί προξενούν αναδιοργάνωση του δέντρου. Πάντως πιο γρήγορα από το ArrayList.



Κατακερματισμος

- Hash: Στον κατακερματισμό ένα κλειδί μετατρέπεται με χρήση κατάλληλης συνάρτησης (hash function) σε μια θέση πίνακα (ευρετήριο)
  - HashSet: Τα στοιχεία του set μπαίνουν σε θέσεις του πίνακα με χρήση μιας συνάρτησης: θέση=func(στοιχείο)
  - HashMap: υλοποίηση του Map με τα κλειδιά να αποθηκεύονται σε HashSet

- Αργή διάσχιση (απαιτεί συνεχείς κλήσεις της func)
- Γρήγορα get/set γιατί καλεί μια φορά τη func και εντοπίζει τη θέση
- Τι γίνεται αν μια θέση είναι γεμάτη;



Διεπαφή Collection

- Μέθοδοι διαχείρισης:
  - size(), isEmpty(), add(Object), remove(Object), contains(Object), iterator()
- Μέθοδοι μαζικής διαχείρισης
  - containsAll(Collection), addAll(Collection), removeAll(Collection), retainAll(Collecton), clear():void
- Μέθοδοι μεταφοράς σε πίνακα
  - toArray():Object[], toArray(Type[]):Type[]

Διεπαφή Set

- Έχει τις ίδιες μεθόδους με το Collection
- Δεν επιτρέπει διπλότυπα
- Δύο Set είναι ίσα αν περιέχουν τα ίδια στοιχεία
  - s1.**containsAll**(s2): true αν s2 υποσύνολο του s1
  - s1.**addAll**(s2): s1 γίνεται η ένωση των s1 και s2
  - s1.**retainAll**(s2): s1 γίνεται η τομή των s1 και s2

Διεπαφή List

- Συλλογή με σειρά (ordered collection ή και ακολουθία) και επιτρέπει διπλότυπα
- Επιπλέον μέθοδοι:
- πρόσβασης σε θέση
  - get(int), set(int,Object), add(int,Object), remove(int index), addAll(int, Collection)
- αναζήτησης
  - indexOf(Object), lastIndexOf(Object)
- διάσχισης
  - listIterator():ListIterator;
  - listIterator(int):ListIterator;
- Υποσυνόλου
  - subList(int, int):List;

Διεπαφή Map

- Αντιστοιχεί κλειδιά σε τιμές (key-value)
- Δεν εγγυάται σειρά αλλά μπορεί να εμφανιστεί σε υλοποιήσεις
- Μέθοδοι διαχείρισης
  - put(Object, Object),get(Object), remove(Object), containsKey(Object), containsValue(Object), size(), isEmpty()
- Μέθοδοι μαζικής διαχείρισης
  - putAll(Map t), clear()
- Μέθοδοι ανάκτησης
  - keySet():Set, values():Collection, entrySet():Set
- Μέθοδοι χειρισμού εγγραφής (interface Map.Entry)
  - getKey(), getValue(), setValue(Object);

Διεπαφή Iterator

- Χρησιμοποιείται για τη διάσχιση των συλλογών
- Επιτρέπει
  - Τον έλεγχο για την ύπαρξη επόμενου: hasNext():boolean
  - Τη μετάβαση στο επόμενο στοιχείο: next():Object
  - Τη διαγραφή του τρέχοντος στοιχείου: remove():void
- Χρησιμοποιούνται για να αντικαταστήσουν τα for:

```
Iterator i = collection.iterator();
while(i.hasNext()) {
Object value = i.next();
....
}
```

Διεπαφή ListIterator

- Κληρονομεί την Iterator
- Προσφέρει μεθόδους:
  - Διάσχισης της λίστας σε κάθε κατεύθυνση
  - Τροποποίησης της λίστας κατά τη διάσχιση
- hasPrevious(), previous(), nextIndex(), previousIndex(), set(Object), add(Object)
- Παράδειγμα:

```
ListIterator i = list.getListIterator();
while ( i.hasNext() )
{ if ( object.compareTo(i.next()) < 0 )
  { break;} }
i.add(object);
```

Αφηρημένες τάξεις

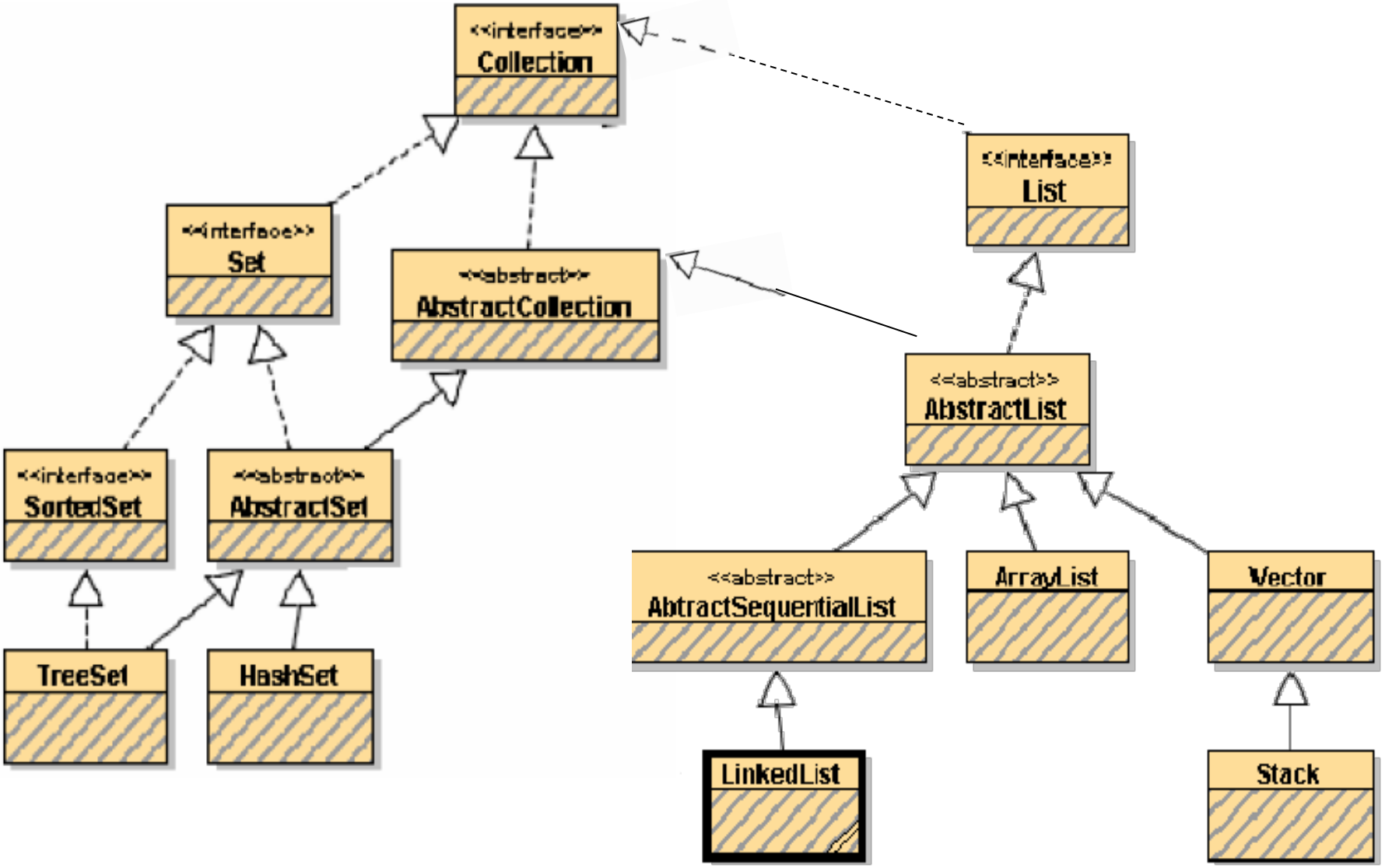
- AbstractCollection
  - Υλοποιεί τις: containsAll(Collection c), contains(Object o), removeAll(Collection c), remove(Object o), retainAll(Collection c) και
  - Ορίζει τις add(Object obj), iterator();
- AbstractSet
  - Υλοποιεί τις: equals(Object) και hashCode()
  - Ορίζει τις: size() και iterator()
- AbstractList
  - Υλοποιεί τις: indexOf(Object) και lastIndexOf(Object)
  - Ορίζει τις: add() και listIterator()

Διεπαφές – υλοποιήσεις και παρελθόν

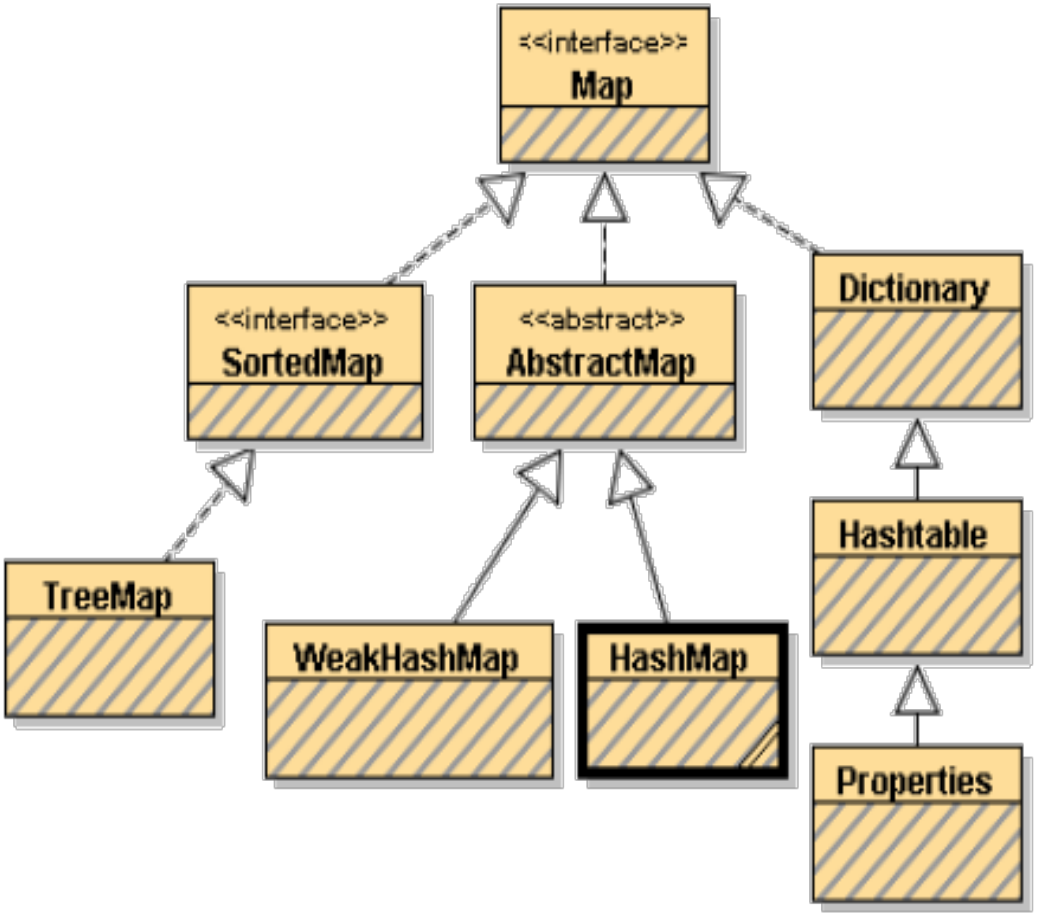
Διεπαφή	Υλοποιήσεις				Παλιότερες υλοποιήσεις
Set	HashSet		TreeSet		
List		ArrayList		LinkedList	Vector Stack
Map	HashMap		TreeMap		HashTable Properties

Μια τάξη υλοποίησης μπορεί να μην υλοποιεί μια συγκεκριμένη μέθοδο του interface (κάποιες μέθοδοι είναι προεραιτικές)
Στην περίπτωση αυτή παράγει **UnsupportedOperationException**

Υλοποιήσεις



Υλοποιήσεις



- WeakHashMap είναι μια υλοποίηση του Map που καλεί τον garbage collector όταν ένα κλειδί δε χρησιμοπείται πλέον

Δικές μας υλοποιήσεις των Collections

- Πρέπει να ορίζουν σωστά τις
  - **equals**()
  - **hashCode**()
  - Και **compareTo**() ή **compare()**
- Στα Map να μη χρησιμοποιούνται πολυμορφικά αντικείμενα
- Στα Map η hashCode() πρέπει:
  - να επιστρέφει πάντα ίδια τιμή για το ίδιο αντικείμενο
  - να επιστρέφει ίδια τιμή για ίσα αντικείμενα
  - μπορεί να επιστρέφει ίδια τιμή για άνισα αντικείμεναh

Ταξινόμηση και Σύγκριση

- Η διεπαφή Comparable πρέπει
  - Να υλοποιείται από όλα τα στοιχεία του SortedSet
  - Να υλοποιείται από όλα τα κλειδιά του SortedMap
  - Να έχει τη μέθοδο: int compareTo(Object o)
  - Να ορίζει τη φυσική σειρά των στοιχείων
- Η διεπαφή Comparator
  - Συγκρίνει δύο αντικείμενα
  - Ορίζει δικό της τρόπο ταξινόμησης
  - Έχει τη μέθοδο: int compare(Object o1,Object o2)
  - Έχει τη μέθοδο: boolean equals(Object o)

Η τάξη Collections

- Περιέχει ένα σύνολο στατικών μεθόδων για το χειρισμό συλλογών (κυρίως Lists).
- Ενσωματώνει:
  - Μεθόδους που υλοποιούν αλγορίθμους (sort(List), binarySearch(List, Object), reverse(List), shuffle(List), fill(List, Object), copy(List dest, List src), min(Collection), max(Collection))
  - Μεθόδους μετατροπής του τύπου μιας συλλογής (toArray())
  - Μεθόδους για δημιουργία συγχρονισμένων συλλογών και συλλογών μόνο για ανάγνωση

Ταξινόμηση

- Παράδειγμα:

```
public static void main (String args[]){
   List l= Arrays.asList( args );
   Collections.sort( l );
}
```

- Η υλοποίηση της sort μπορεί να γίνει με ένα προσωρινό array

```
public static void sort(List list, Comparator c){
   Object a[] = list.toArray();
   Arrays.sort(a, c); //Αλγόριθμος merge sort
   ListIterator i =list.listIterator();
   for (int j=0;j<a.length; j++) {i.next(); i.set(a[j]);}
}
```

Μια άλλη κατηγοριοποίηση συλλογών

- Τροποποιήσιμες / Μη τροποποιήσιμες: Ανάλογα με το αν υποστηρίζουν μεθόδους τροποποίησης: add(), remove(), clear()
- Σταθερές/ασταθείς (mutable): Ανάλογα με το αν επιτρέπουν αλλαγές κατά τη διάσχιση: iterator(), size(), contains()
- Σταθερού/μεταβλητού μεγέθους: Ανάλογα με το αν διατηρούν το μέγεθός τους ανεξάρτητα με την προσθήκη στοιχείων

Πώς επιλέγουμε τύπο συλλογής

- Πώς θέλουμε να εντοπίζουμε αντικείμενα;
  - Με κλειδί ➔ **Map**
  - Με θέση ➔ **ArrayList** (ή **array**)
- Ποια σειρά μας ενδιαφέρει στη διάσχιση;
  - Τα στοιχεία να είναι ταξινομημένα➔ **TreeSet**
  - Να διατηρείται η σειρά εισαγωγής➔ **List**
- Τι θέλουμε να γίνεται γρήγορα;
  - Προσθήκη και αφαίρεση στοιχείων➔ **LinkedList**
  - Αναζήτηση➔ **Set**

Tree ή Hash

- Αν ο hashCode είναι συνεπής της equals και δεν ενδιαφέρει η σειρά διάσχισης
  - Χρησιμοποιούμε hash υλοποίηση
  - Αλλιώς tree
- Αν έχουμε και τις δύο υλοποιήσεις (Hash και Tree) η Hash είναι συνήθως πιο γρήγορη
- Για Tree υλοποιήσεις ελέγχουμε αν:
  - Έχει υλοποιηθεί η διεπαφή Comparable ή αν έχει οριστεί ο Comparator (στα αντικείμενα του Set, ή στα κλειδιά του Map)