Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

HY-252 – Οντοκεντρικός Προγραμματισμός
Βασίλης Χριστοφίδης

Επαναληπτική Εξέταση (3 ώρες)
Ημερομηνία: 12 Σεπτεμβρίου 2002

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Άσκηση 1 (10 μονάδες)

1.1 Τι συμβαίνει κατά την μετάφραση του παρακάτω προγράμματος Java:

```
final class A { ...}
class B extends A { ... }
```

Λύση:
Compile error: We can't extend final classes

1.2 `Shape` είναι μια αφηρημένη (abstract) κλάση που ορίζεται ως ακολούθως:

```
abstract class Shape {
     abstract void draw();
     abstract void move(int x, int y);
     abstract void scale(double factor);
     int x,y;
     int width, height;
}
```

Τι συμβαίνει κατά την μετάφραση της παρακάτω κλάσης `Square` που εξειδικεύει την `Shape`?

```
class Square extends Shape {
     void move(int x, int y)
     {this.x = x; this.y = y;};
}
```

Λύση:
Compile error: Concrete classes extending an abstract class should implement all its abstract methods

1.3 `Switch` είναι μια διεπαφή (interface) που ορίζεται ως ακολούθως:

```
interface Switch {
     public void turnOn();
     public void turnOff();
}
```

Τι συμβαίνει κατά την μετάφραση του παρακάτω ορισμού της κλάσης `CamelSwitch` που υλοποιεί την `Switch`?

```
class CamelSwitch implements Switch {
     public void turnOn(){on = true;}
     private boolean on = false;
}
```

Λύση:
Compile error: Concrete classes implementing an interface should implement all its abstract methods

1.4 Ποιο είναι το λάθος στον παρακάτω ορισμό της κλάσης `Test`?

```
class Test {
       int x = 0;
       public static void main(String[] args){
          System.out.println(x);
       }
}
```

Λύση:
Compile error: Static methods can't access instance variables

1.5 Θα μεταφραστεί χωρίς λάθη το παρακάτω πρόγραμμα Java? Αν όχι προτείνετε μια σωστή διόρθωσή του.

```
class A {
     protected int x;
}
class B extends A {
      private int y;
}
class C extends B {
       int z;
       public C(){
            z = super.y;
       }
```

Λύση:
Compile error: Instance variable in class B should be protected

Άσκηση 2 (16 μονάδες)

Ποια είναι η έξοδος του παρακάτω προγράμματος Java? Δώστε την απάντηση σας στις κενές γραμμές που προβλέπονται στην εκφώνηση της άσκησης.

```
class A {
 public void f()    {System.out.println("  A-f");}
 public void h(A o) {System.out.print("A-h"); o.f();}
 public void m(A o) {System.out.print("A-m"); o.f();}
}
class B extends A {
 public void f()    {System.out.println("  B-f");}
 public void h(A o) {System.out.print("B-h"); o.f();}
 public void m(B o) {System.out.print("B-m"); o.f();}
}
```

```
public class Test {
 public static void main(String[] args) {
   B b1 = new B();
   B b2 = new B();
   A a1 = new A();
   A a2 = b2;
   b1.h(a1);
```

Λύση:
```
B-h      A-f
```

```
   b1.m(a1);
```

Λύση:
```
A-m      A-f
```

```
   b1.h(a2);
```

Λύση:
```
B-h      B-f
```

```
   b1.m(a2);
```

Λύση:
```
A-m      B-f
```

```
   a1.h(a2);
```

Λύση:
```
A-h      B-f
```

```
   a1.m(a2);
```

Λύση:
```
A-m      B-f
```

```
   a2.h(a2);
```

Λύση:
```
B-h      B-f
```

```
   a2.m(a2);
```

Λύση:
```
A-m      B-f
```

```
  }
}
```

Άσκηση 3 (14 μονάδες)

Σας δίνεται ο ορισμός σε Java της παρακάτω κλάσης **Employee**:

```
class Employee {
 private String name;
 private String ssn;
```

```
 private int salary;
 public Employee(String name, String ssn, int salary){
    this.name=name;
    this.ssn=ssn;   this.salary=salary;
  }
  public int earnings() {
    return salary;
  }
  public String toString() {
    return("Name:"+ name +"\n"+"SSN:"+ ssn +"\n"+
           "Earnings:"+ earnings() +"\n");
  }
}
```

3.1 (**6 μονάδες**) Υλοποιείστε μια καινούργια κλάση **Executive** η οποία εξειδικεύει την **Employee** έχοντας μια επιπλέον μεταβλητή στιγμιοτύπων **bonus** καθώς και μεθόδους στιγμιοτύπων συμπεριλαμβανομένου και ενός καινούργιου κατασκευαστή αντικειμένων. Στην **Executive**, πρέπει να (επανα-)ορίσετε τις παρακάτω μεθόδους της **Employee** έτσι ώστε:

- Ο κατασκευαστής αντικειμένων της **Executive** πρέπει να αρχικοποιεί τρεις μεταβλητές στιγμιοτύπων (**name**, **ssn**, **salary** και **bonus**) χρησιμοποιώντας 2 συμβολοσειρές και 2 ακεραίους που δίνονται σαν παράμετροι της μεθόδου.
- Η μέθοδος **earnings** πρέπει να επιστρέφει τις αποδοχές ενός διευθυντικού στελέχους σαν το άθροισμα του **salary** και του **bonus**.

Λύση:
```
class Executive extends Employee {
 private int bonus;
 // constructor
 public Executive(String name, String ssn,
                  int salary, int bonus) {
  super(name,ssn,salary);
  this.bonus=bonus;
 }
 // earnings method
 public int earnings() {
  return(super.earnings()+bonus);
}
```

3.2 (**8 μονάδες**) Υλοποιείστε τώρα μια κλάση **SME** για την αναπαράσταση μιας Μικρής-και-Μεσαίας Εταιρείας (ΜΜΕ) που μπορεί να έχει το πολύ 100 υπαλλήλους. Η κλάση **SME** πρέπει να περιλαμβάνει τις παρακάτω μεταβλητές και μεθόδους στιγμιοτύπων:

- Την μεταβλητή **employees** τύπου πίνακα η οποία θα περιέχει τους υπαλλήλους της εταιρείας (Θυμηθείτε την χρήση αναφορών σε αντικείμενα Java).
- Τον κατασκευαστή αντικειμένων της **SME** χωρίς παραμέτρους που δημιουργεί τον πίνακα με τους υπαλλήλους της εταιρείας χωρίς να προσθέτει αρχικά κανέναν υπάλληλο. Εξηγείστε γιατί ο εξ ορισμού (default) κατασκευαστής αντικειμένων της **SME** δεν προσφέρει αυτήν την λειτουργικότητα.

- Την μέθοδο **addEmployee** η οποία προσθέτει ένα αντικείμενο **Employee** στον πίνακα **employees**.
- Την μέθοδο **payday** η οποία εκτυπώνει το **name**, **ssn** και **earnings** όλων των υπαλλήλων της εταιρείας.

Λύση:
```
class SME {
 private Employee[] employees;
 private int empNum;
 // Constructor
 public SME() {
  employees = new Employee[100];
  empNum = 0;
 }
 // addEmployee method
 public void addEmployee(Employee e) {
   employees[empNum++] = e;
 }
 // payday method
 public void payday() {
  for (int i=0; i< empNum ; i++)
      System.out.println(employees[i].toString());
 }
}
```

Άσκηση 4 (20 μονάδες)

Υλοποιείστε μια μέθοδο **insert** η οποία εισάγει ακεραίους σε ταξινομημένες λίστες **List** χρησιμοποιώντας διαδοχικές κλήσεις μεθόδων όπως στο παρακάτω παράδειγμα:

```
(new Cons(1, new Cons(4, new Empty()))).insert(2)
==> new Cons(1, new Cons(2, new Cons(4, new Empty())))

(new Cons(1, new Cons(4, new Empty()))).insert(0)
==> new Cons(0, new Cons(1, new Cons(4, new Empty())))

(new Empty()).insert(4)
 ==> new Cons(4, new Empty())
```

4.1 (**10 μονάδες**) Για την υλοποίηση της μεθόδου **insert** χρησιμοποιείστε τους παρακάτω ημιτελείς ορισμούς των κλάσεων **Empty** και **Constructed** (για συντομογραφία **Cons**):

```
abstract class List {
    abstract public List insert(int x);
}
class Empty extends List {
    public List insert(int x) {
```

Λύση:
```
return new Cons(x, this);
```

```
    }
}
class Cons extends List {
    public int first;
    public List rest;

    public Cons(int firstelement, List restelements) {
     rest = restelements;
     first = firstelement;
    }

    public List insert(int x) {
```

Λύση:
```
if (x < this.first)
  return new Cons(x, this);
else
return new Cons(this.first, this.rest.insert(x));
```

```
    }
}
```

4.2 (**10 μονάδες**) Υλοποιείστε τώρα μια μέθοδο **sort** σε ταξινομημένες λίστες ακεραίων χρησιμοποιώντας την μέθοδο **insert** του προηγούμενου ερωτήματος. Η βασική ιδέα του αλγορίθμου ταξινόμησης είναι η ακόλουθη: για να ταξινομήσουμε μια μη κενή λίστα πρώτα ταξινομούμε το υπόλοιπο μέρος της λίστας **rest** και μετά εισάγουμε το πρώτο στοιχείο της λίστας **first** στην ταξινομημένη λίστα που μόλις δημιουργήσαμε. Για την υλοποίηση της **sort** χρησιμοποιείστε τους παρακάτω ημιτελείς ορισμούς των κλάσεων **Empty** και **Cons**:

```
abstract class List {
    abstract public List insert(int x);
    abstract public List sort();
}

class Empty extends List {
    public List sort() {
```

Λύση:
```
return this;
```

```
    }
}

class Cons extends List {
    public int first;
    public List rest;

    public List sort() {
```

Λύση:
```
return (this.rest.sort()).insert(this.first);
```

```
    }
}
```

Άσκηση 5 (20 μονάδες)

Θέλουμε να υλοποιήσουμε ένα πρόγραμμα Java για τον χειρισμό ηλεκτρονικών διευθύνσεων (e-mail). Οι ηλεκτρονικές διευθύνσεις αποθηκεύονται σ'ένα απλό αρχείο κειμένου όπως δίνεται ακολούθως:

```
Dimitris Plexousakis
dp@csd.uch.gr
Vassilis Christophides
christop@csd.uch.gr
Apostolos Traganitis
tragani@ics.forth.gr
```

Ανεξάρτητα από τον τρόπο δημιουργίας του παραπάνω αρχείου ηλεκτρονικών διευθύνσεων (π.χ. χρησιμοποιώντας κάποιον text editor) θέλουμε να έχουμε την δυνατότητα ταξινόμησης (`sort`) ως προς τα ονόματα. Για παράδειγμα το αποτέλεσμα της ταξινόμησης του παραπάνω αρχείου δίνεται ακολούθως:

```
Vassilis Christophides
christop@csd.uch.gr
Dimitris Plexousakis
dp@csd.uch.gr
Apostolos Traganitis
tragani@ics.forth.gr
```

Επιπλέον θέλουμε να έχουμε την δυνατότητα ενοποίησης (`merge`) δύο αρχείων ηλεκτρονικών διευθύνσεων. Το αποτέλεσμα πρέπει να είναι ένα ταξινομημένο αρχείο ηλεκτρονικών διευθύνσεων ανεξάρτητα από το αν τα αρχικά αρχεία ήταν ταξινομημένα ή όχι.

- Γράψτε μια δημόσια (`public`) κλάση `Sorter` και τις μεθόδους `insertFromReader` και `putToWriter` οι οποίες διαβάζουν ένα αρχείο ηλεκτρονικών διευθύνσεων και γράφουν σ'ένα ταξινομημένο αρχείο ηλεκτρονικών διευθύνσεων.
- Γράψτε μια δημόσια (`public`) κλάση `MergeAndSort` και την κύρια (`main`) μέθοδο η οποία δέχεται τρία ονόματα αρχείων σαν είσοδο. Τα πρώτα δύο ορίσματα δίνουν τα ονόματα των αρχείων ηλεκτρονικών διευθύνσεων που θέλουμε να ενοποιήσουμε και το τρίτο το ενοποιημένο και ταξινομημένο αρχείο ηλεκτρονικών διευθύνσεων.

Σημείωση: Για την άσκηση θεωρείστε ότι οποιοδήποτε όνομα αρχείου δίνεται στην είσοδο είναι ένα αρχείο ηλεκτρονικών διευθύνσεων που ακολουθεί την μορφή που σας δόθηκε στην εκφώνηση: η πρώτη γραμμή περιέχει το όνομα και η δεύτερη την ηλεκτρονική διεύθυνση, και ο συνολικός αριθμός γραμμών στο αρχείο είναι άρτιος (Ακόμη και αν θεωρήσετε καλά-ορισμένα αρχεία διευθύνσεων ο μεταφραστής Java απαιτεί χειρισμό εξαιρέσεων!).

Το πρόγραμμά σας πρέπει να χρησιμοποιήσει την κλάση `TreeMap` που υλοποιεί την διεπαφή `SortedMap`. Για να προσθέσετε ένα ζεύγος όνομα-διεύθυνση πρέπει να χρησιμοποιείτε την μέθοδο `put(name, mail)`:.έτσι το όνομα (`name`) χρησιμοποιείται σαν κλειδί και η διεύθυνση (`mail`) σαν τιμή. Η ταξινόμηση γίνεται αυτόματα από την μέθοδο `put` χρησιμοποιώντας την στάνταρτ (αλφαβητική) διάταξη σε συμβολοσειρές (δηλ. δεν χρειάζεται να υλοποιήσετε δικές σας μεθόδους διάταξης). Το αποτέλεσμα της ταξινόμησης γίνεται εμφανές με την χρήση της μεθόδου επανάληψης (iterator) `iterator` η οποία σέβεται την διάταξη των κλειδιών ενός `SortedMap` όπως επιστρέφονται από την μέθοδο `entrySet`.

Λύση:

```
import java.io.*; import java.util.*;
public class Sorter {
 private SortedMap smap = new TreeMap()
 public void insertFromReader(BufferedReader r)
                                    throws IOException{
   String name = r.readLine();
   String mail = r.readLine();
   While (name != null && mail != null){
     Smap.put(name, mail);
     name = r.readLine();
     mail = r.readLine();
   }
 }
public void putToWriter(BufferedWriter w) throws
                                          IOException{
   Iterator it = smap.entrySet().iterator();
   Map.Entry entry;
   While(it.hasnext()) {
     entry = (Map.Entry) it.next();
     w.write(entry.getKey() + "\n");
     w.write(entry.getvalue() + "\n");
   }
 }
}
public class MergeAndSort {
 public static void main(String[] args){
  BufferedReader in1, in2;
  BufferedWriter out;
  Sorter sorter = new Sorter();
  try { if(args.length ==3) {
      in1 = new BufferedReader(new
                                    FileReader(args[0]));
      sorter.insertFromReader(in1);
      in1.close();
      in2 = new BufferedReader(new
                                    FileReader(args[1]));
      sorter.insertFromReader(in2);
      in2.close();
      out = new BufferedWriter(new
                                    FileWriter(args[2]));
      sorter.putToWriter(out);
      out.close();
     }
    else throw new IllegalArgumentException("The
            number of Input/Output Files is wrong");
   }
   catch (FileNotFoundEception e) {
     throw new RunTimeException("File not found:" +
                              args[0] + "or" args[1]);
     }
   catch (IOException e) {
     throw new RunTimeException("Error in writing
                      file:" + args[2] + "or reading
                      files" + args[0] + args[1]));
     }
   }}
```

Άσκηση 6 (20 μονάδες)

6.1 (**4 μονάδες**) Αναφέρετε τις δύο τεχνικές με τις οποίες μπορούμε να διατηρήσουμε (persistence) αντικείμενα στο δίσκο μετά την λήξη ενός προγράμματος Java. Ποιες είναι οι βασικές διαφορές τους?

> **Λύση:**
> **Serialization:** the serializing process is automatically handled by Java but the default behavior can be overridden. There are no methods the developer MUST implement
> **Externalization:** the developer must manual handle the serializing process and must implement the methods readExternal and writeExternal

6.2 (**4 μονάδες**) Αναφέρετε τις δύο στρατηγικές με τις οποίες μπορούμε να χειριστούμε εξαιρέσεις (exceptions) κατά την εκτέλεση ενός προγράμματος Java. Ποιες είναι οι βασικές διαφορές τους?

> **Λύση:**
> **handle now or use a try-catch-block structure:**
>
> **handle later or use a throws clause in the method declaration:**

6.3 (**4 μονάδες**) Αναφέρετε τους δύο μηχανισμούς με τους οποίους μπορούμε να ορίσουμε μεθόδους με το ίδιο όνομα σε μια κλάση ή ιεραρχία κλάσεων Java. Ποιες είναι οι βασικές διαφορές τους?

> **Λύση:**
> **Overload:** several methods are defined with the same name but different signatures (parameter/argument lists). Overloading lets you define a similar operation in different ways for similar object types
> **Override:** a subclass defines a method with the same name and signature of a method in the superclass. Overriding lets you define a similar operation in different ways for different but compatible object types

6.4 (**4 μονάδες**) Αναφέρετε τις δύο διεπαφές με τις οποίες μπορούμε να συγκρίνουμε (compare) αντικείμενα σ'ένα πρόγραμμα Java. Ποιες είναι οι βασικές διαφορές τους?

> **Λύση:**
> **`Comparator` interface**: One is to encapsulate the *knowledge* about which objects are comparable and how to compare them in an object! Such an object implements **`compare()`** method and it used for designing a design **custom ordering scheme**
> **`Comparable` interface**: The other is to make the objects we compare implement themselves a comparison operation via the **`compareTo()`** method and it used for designing a design **natural ordering schema**

6.5 (**4 μονάδες**) Αναφέρετε τους δύο τρόπους με τους οποίους μπορούμε να δημιουργούμε κλώνους (clone) αντικειμένων σε ένα πρόγραμμα Java. Ποιες είναι οι βασικές διαφορές τους?

Λύση:
In many classes, the default implementation of (protected) **clone**() will be wrong because it duplicates a reference to an object that shouldn't be shared
A shallow copy, by default:
A deep copy is often preferable: