Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

ΗΥ-252 – Οντοκεντρικός Προγραμματισμός
Βασίλης Χριστοφίδης

Επαναληπτική Εξέταση (3 ώρες)
Ημερομηνία: 29 Αυγούστου 2001

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Ασκηση 1 (10 μονάδες)

Θεωρήστε τους παρακάτω ορισμούς διεπαφών και κλάσεων Java

```
interface Point {
   ...
}

class CartPoint implements Point {
   ...
   CartPoint(double x, double y) { ... }
   ...
}

class PolarPoint implements Point {
   ...
   PolarPoint(double r, double a) { ... }
   ...
}
```

Για κάθε μία απο τις παρακάτω ακολουθίες εντολών γράψτε αν είναι:

- **ill-typed:** αν αποτυγχάνει ο ελεγκτής τύπων της Java,
- **runtime error:** αν επιτυγχάνει ο ελεγκτής τύπων αλλά κάποια down-cast λειτουγία αποτυγχάνει κατά την διάρκεια της εκτέλεσης
- **OK:** αν επιτυγχάνει ο ελεγκτής τύπων και εκτελείται χωρίς λάθη.

1. (2 μονάδες) `Point p = new CartPoint(3.0, 4.0);`

Λύση: OK

2. (2 μονάδες)
```
Point p1 = new CartPoint(3.0, 4.0);
CartPoint p2 = p1;
```

Λύση: ill-typed

3. (2 μονάδες) `PolarPoint p1 = new CartPoint(3.0, 4.0);`

Λύση: ill-typed

4. (2 μονάδες) `Point p1 = new CartPoint(3.0, 4.0);`
`CartPoint p2 = (CartPoint)p1;`

Λύση: OK

5. (2 μονάδες) `Point p1 = new CartPoint(3.0, 4.0);`
`PolarPoint p2 = (PolarPoint)p1;`

Λύση: runtime error

Ασκηση 2 (10 μονάδες)

Υποθέστε ότι η Java class A επεκτείνει (extends) την class B. Και οι δύο κλάσεις έχουν κατασκευαστές αντικειμένων (constructors) που δεν παίρνουν παραμέτρους.

(a) (2 μονάδες) Συμπληρώστε με "A" η με "B" τα παρακάτω κενά:

1. Η Class B είναι πατέρας (parent) της class A .
2. Η Class A ειναι παραγώμενη (derived) απο την class B .

(b) (3 μονάδες) Υποθέστε ότι και οι δύο κλάσεις έχουν μια μέθοδο `toString()` ορισμένη ως ακολούθως:

```
/** toString() of A */
public String toString()
{
return super.toString()+(new String("Class
A\n"));
}
/** toString() of B */
public String toString()
{
    return new String("Class B\n");
}
```

Μόνο μία απο τις παρακάτω ακολουθίες εντολών είναι σωστή. Ποια είναι η σωστή ακολουθία? Εξηγήστε με συντομία την απάντησή σας.

Ακολουθία Εντολών 1	Ακολουθίες Εντολών 2

`A varA = new A();`
`System.out.print(varA);`
`varA = new B();`
`System.out.print(varA);`
	`B varB = new A();`
	`System.out.print(varB);`
	`varB = new B();`
	`System.out.print(varB);`

Λύση: Segment 2 is valid because A *is a* B, but not the other way around. Therefore, the child class can appear on the right-hand-side of an assignment operation to the parent class.
In line 3 of Segment 1, a variable of type A is pointed to a new object B, and this shall cause a compilation error.

(c) (3 μονάδες) Γράψτε την έξοδο της σωστής ακολουθίας εντολών στο ερώτημα (b).

Λύση: The program will output the followings:
```
Class B
Class A
Class B
```

(d) (2 μονάδες) Δώστε με συντομία την χρησιμότητα του αναγνωριστικού δικαιώματος πρόσβασης (visibility modifier) "`protected`".

Λύση: The protected modifier makes class *methods* and *variables* visible to the child classes, but not to the classes outside the "family".

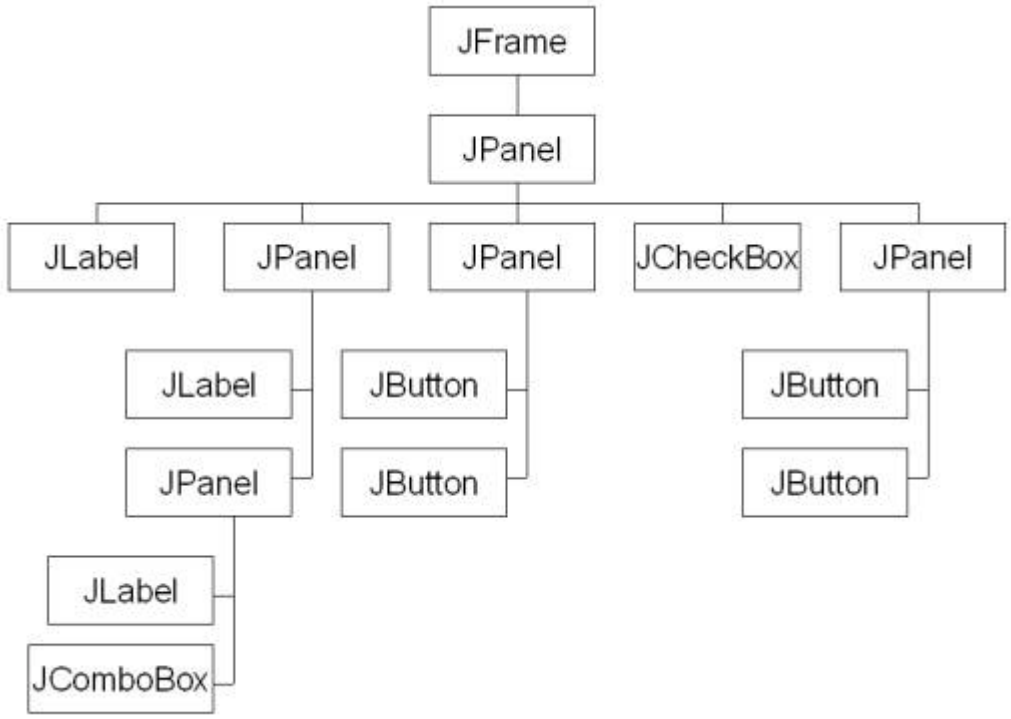
Ασκηση 3 (8 μονάδες)

(a) (2 μονάδες) Ποια από τις παρακάτω δομές δεδομένων καταλληλότερη για την *διανομή γεγονότων* (*event dispatching)* στο Java Swing*: Δυναμικά Συνδεδεμένη Λίστα (*Dynamic Linked List), Ουρά (Queue) ή Στοίβα (Stack)?

Λύση: Queue.

(b) (6 μονάδες) Το Java Swing Box layout μας επιτρέπει να τακτοποιούμε τις διάφορες συνιστώσες του Swing GUI είτε σε κατακόρυφους είτε σε οριζόντιους σχηματισμούς (configurations). Υποθέτοντας οτι το Box layout είναι ο μόνος διαθέσιμος διαχειριστής παρουσίασης (*layout manager*) σχεδιάστε το δέντρο συνιστωσών του *dialog box* του διπλανού σχήματος. Αναφερθείτε σε κάθε συνιστώσα στο σχήμα σας με το αντίστοιχο όνομα της κλάσης Swing, π.χ. *JFrame, JPanel, JRadioButton, JLabel, Jbutton*, *JCheckBox* και *JComboBox.* Η συνιστώσα στην κορυφή της ιεραρχίας σας πρέπει να είναι η *JFrame.*

Λύση:



Ασκηση 4 (12 μονάδες)

Υποθέστε μια κλάση Τρίλιζα (TicTacToe) που υλοποιεί το αντίστοιχο παιχνίδι και στην οποία έχουμε ορίσει την παρακάτω μέθοδο "makeMove":

```
// This method sets the board value on coordinate
// (x,y) Return true if successful, false if failed
public boolean makeMove(int x, int y, int val)
{// Check occupacy of the board
    if (val!=PLAYER_O && val!=PLAYER_X)
  return false;
 // Check the board array and see if it is out of
 // bound The array "board" has size (board_size x
 // board_size)
    if (x<0 || x>=board_size)
  return false;
    if (y<0 || y>=board_size)
  return false;
    if (board[y][x]==PLAYER_N)
 {board[y][x] = val;
  return true; }
    else
  return false;
}
```

Η δεύτερη και η τρίτη εντολή "**if**" χρησιμοποιούνται για να ελέξουν τα επιτρεπτά όρια του δισδιάστατου πίνακα "**board**" ώστε να αποφύγουμε την εμφάνιση μιας εξαίρεσης (exception) του τύπου **ArrayIndexOutOfBoundsException** κατά την εκτέλεση της μεθόδου **makeMove** με λάθος παραμέτρους.

(a) (2 μονάδες) Με δεδομένο ότι η εκτέλεση της παραπάνω μεθόδου μπορεί να διουργήσει μια Java exception, εξηγήστε γιατί η δήλωση "throws" δεν είναι υποχρεωτική στην υπογραφή (signature) της μεθόδου?

Λύση: The exception **ArrayIndexOutOfBound** is *not* a *checked* exception. An unchecked exception in Java cannot be detected during compilation time, and they are the descendents of the Java class **RuntimeException**.

(b) (6 μονάδες) Ξαναγράψτε το σώμα της παραπάνω μεθόδου χρησιμοποιώντας την εντολή ***try...catch...finally***. Η χρήση της ***finally*** είναι προαιρετική.

Λύση:

```
public boolean makeMove(int x, int y, int val)
{
    if (val!=PLAYER_O && val!=PLAYER_X)
  return false;
    try
    {
if (board[y][x]==PLAYER_N)
  {
board[y][x] = val;
return true;
  }
else
return false;
    }
    catch (ArrayIndexOutOfBoundsException e)
    {
return false;
    }
}
```

c) (4 μονάδες) Γράψτε με συντομία τα πλεονεκτήματα και τα μειονεκτήματα των δύο υλοποιήσεων της μεθόδου "**makeMove**" δηλ. με ή χωρίς την εντολή ***try...catch...finally.***

Λύση: Exception handling allow us to (a) Make normal logic of an action clear (b) Decide whether to handle or defer handling an error (c) Handle errors in library code on a custom basis. However, we should use exceptions for exceptional situations only: (a) exception handling is a less structured mechanism than the other language control structures (b) exception handling are usually less efficient than other control mechanisms (c) when used for dealing with situations other than errors, it could lead to a less understandable code

Ασκηση 5 (20 μονάδες)

(α) (7 μονάδες) Γράψτε μια Κλάση **Employee** η οποία έχει
τις παρακάτω *μεταβλητές στιγμιοτύπων* (*instance variables*)

- **name** - a String containing the name of the employee
- **salary** - a float containing the employee's annual salary
- **hiredate** - a String in the format ("MMDDYYYY") containing the date the employee was hired

και τις παρακάτω *μεθόδους*

- **getName** - returns the employee's name
- **getSalary** - returns the employee's salary
- **setSalary** - sets the employee's salary
- **getHireDate** - returns the employee's hire date
- **toString** - returns a string containing state of the employee object
- a **constructor** that initializes the attributes

Λύση:

```
class Employee
{
 private String name;
 private float salary;
 private String hireDate;

 public Employee(String n, float s, String h)
 {name = n; salary = s; hireDate = h;}

 public String getName(){return name;}

 public float getSalary(){return salary;}

 public String getHireDate(){return hireDate;}

 public void setSalary(float amt) {salary=amt;}

 public String toString()
 {return name + "-" + salary + "-" + hireDate;}
}
```

(b) (7 μονάδες) Γράψτε μια Κλάση **Manager** η οποία έχει επιπλέον των *μεταβλητών στιγμιοτύπων* και *μεθόδων* της κλάσης **Employee** τις παρακάτω:

μεταβλητές στιγμιοτύπων

- **secretaryName** - a string containing the name of the managers secretary

και *μεθόδους*

- **getSecretaryName** - gets the secretary's name
- **setSecretaryName** - sets the secretary's name

Σημείωση: Μην ξεχάσετε να τροποποιήσετε κατάλληλα την μέθοδο **toString** ώστε να τυπώνει και το όνομα της γραμματέας του διευθυντή. Επίσης τροποποιήστε την μέθοδο **getName** ώστε η συμβολοσειρά "**Boss**" να προστείθεται μπροστά απο το όνομα του διευθυντή κάθε φορά που εκτελείται η μέθοδος.

Λύση:

```
class Manager extends Employee
{
 String secretaryName;

 public Manager(String n, float s, String h,
String sn)
{super(n, s, h); secretaryName = sn;}

 public String getSecretaryName(){return
secretaryName;}

 public String getName()
{return "Boss " + super.getName();}

 public void setSecretaryName(String newName)
{secretaryName = newName;}

 public String toString()
 {return super.toString() + "-" + secretaryName;}
}
```

(c) (6 μονάδες) Γράψτε μια κλάση **DreamJob** η οποία υλοποιεί τις παρακάτω λειτουργίες

- **creates** an array of 5 Employee objects
- **populates** the array with the following data (you can use initials for the names)
- **prints** out a list of employee names
- **prints** a blank line
- **prints** a list containing all employee information

Position	Name	Salary	Date Hired	Secretary
Manager	Jon Fouss	$1,000,000	07101999	Alyssa Milano
Employee	Jennifer Aniston	$200,000	01292000	
Employee	Yasmine Bleeth	$250,000	03112000	
Employee	Kathy Ireland	$175,000	06262000	
Employee	Alyssa Milano	$300,000	08011999	

Λύση:

```
public class DreamJob
{
 public static void main(String[] args)
 {
  Employee[] x = new Employee[5];
  x[0] = new Manager("JF", 1000000f, "07101999",
  "AM");
  x[1] = new Employee("JA", 200000f, "01292000");
  x[2] = new Employee("YB", 250000f, "03112000");
  x[3] = new Employee("KI", 175000f, "06262000");
  x[4] = new Employee("AM", 300000f, "08011999");
  for (int i=0; i<x.length; i++)
    System.out.println(x[i].getName());
  System.out.println();
  for (int i=0; i<x.length; i++)
    System.out.println(x[i]);
 }
}
```

Ασκηση 6 (25 μονάδες)

Θεωρήστε τους παρακάτω 4 ΑΤΔ (ADTs), ορισμένους σαν διεπαφές Java:

```
public interface StoreOnlyA {
  public void putIn(Object newThing);
  public int size();
  boolean isEmpty();
}
public interface StoreOnlyB {
  public void putIn(Object newThing) throws
   FullStoreException;
  public int size();
  boolean isEmpty();
}
public interface GetA {
  public Object getNewest();
    // Remove and return object most recently putIn
  public Object seeNewest();
    // Return, but don't remove, object most
    // recently putIn
}
```

```
public interface GetB {
  public Object getOldest();
   // Remove and return object least recently putIn
  public Object seeOldest();
   // Return, but don't remove, object least
   // recently putIn
}
```

a) [10 μονάδες] Υποθέστε τώρα ότι θέλουμε να υλοποιήσουμε ένα ΑΤΔ Στοίβα (Stack ADT) χρησιμοποιώντας πίνακες Java σαν την υποκείμενη δομή δεδομένων:

1. [2 μονάδες] Στην παρακάτω δήλωση της κλάσης:
   `public class Stack implements ----------`
   ποιες απο τις παραπάνω διεπαφές πρέπει να χριμοποιηθούν στην θέση των κενών?

Λύση: `StoreOnlyB` and `GetA`.

2. [3 μονάδες] Γράψτε τις *μεταβλητές στιγμιοτύπων* (*instance variables*) που απαιτούνται γι'αυτήν την κλάση:

Λύση:
```
private Object[] data;
private int top;
```

3. [5 μονάδες] Γράψτε τον απαιτούμενο κώδικα για το σώμα της μεθόδου `putIn()` της κλάσης `Stack`:

Λύση:
```
public void putIn(Object obj) throws
   FullStoreException {
     if (top == data.length - 1)
 throw new FullStoreException("Stack
    full, object not putIn");
 data[++top] = obj;
     }
```

b) [15 μονάδες] Υποθέστε τώρα ότι θέλουμε να υλοποιήσουμε ένα ΑΤΔ Ουρά (Queue ADT) χρησιμοποιώντας μια Μονά Συνδεδεμένη Λίστα (Singly Linked List) σαν την υποκείμενη δομή δεδομένων:

1. [2 μονάδες] Στην παρακάτω δήλωση της κλάσης:
   `public class Queue implements ----------`
   ποιες απο τις παραπάνω διεπαφές πρέπει να χριμοποιηθούν στην θέση των κενών?

Λύση: `StoreOnlyA` and `GetB`.

2. [3 μονάδες] Γράψτε τις *μεταβλητές στιγμιοτύπων* (*instance variables*) που απαιτούνται γι'αυτή την κλάση:

```
Λύση: private Node head;
     // not strictly necessary, but good for
     // efficiency
      private Node tail;
     // not strictly necessary, but good for
     // efficiency
      private int size;
```

3. [5 μονάδες] Γράψτε τον απαιτούμενο κώδικα για το σώμα της μεθόδου **putIn()** της κλάσης **Queue**:

```
Λύση: public void putIn(Object obj) {
      Node newNode = new SomeNode(obj, null);
      if (size == 0)
 head = newNode;
      else
 tail.setNext(newNode);
      size++;
      }
```

4. [5 μονάδες] Σκιαγραφήστε τον ορισμό μιας διεπαφής **Node** στον βαθμό που την χρησιμοποιήσατε στο υποερώτημα b.3.: Θεωρήστε επίσης μια κλάση που υλοποιεί την διεπαφή **Node**

```
Λύση: public interface Node {
      // sets "next" reference
      public void setNext(Node next);
      // gets "next" reference
      public Node getNext();
      ...
      }
      public class SomeNode implements Node {
SomeNode(Object obj, Node next) {
// stores obj reference and sets next
// reference:
...
}
```

Ασκηση 7 (25 μονάδες)

Η υλοποίηση ενός σχετικά απλού ΑΤΔ (ADT) χρησιμοποιώντας έναν πιό εκφαστικό ΑΤΔ είναι μια συνηθισμένη προγραμματιστική πρακτική. Π.χ. ο ΑΤΔ Στοίβα (Stack) μπορεί να υλοποιηθεί χρησιμοποιώντας τον ΑΤΔ Ουρά Διπλών Ακρων (Doubly-Ended Queue η Deque). Σ'αυτή την

άσκηση θεωρούμε την αντίθετη περίπτωση, δηλ. την υλοποίηση ενός εκφαστικού ΑΤΔ χρησιμοποιώντας έναν απλούστερο ΑΤΔ

a) [5 μονάδες] Συμπληρώστε τις παρακάτω διεπαφές Java για τον ΑΤΔ **Deque**. Γράψτε σύντομα σχόλια που επεξηγούν κάθε μέθοδο.

```
public interface Deque {
public int size();
public boolean isEmpty();
public Object first();
public void insertFirst(Object o);
public Object removeFirst();
}
```

```
Λύση: // Methods first() and removeFirst() can
// throw a DequeEmptyException
// return element at end
public Object last() throws
                              DequeEmptyException;
// insert object at end of deque
public void insertLast(Object o);
// delete item at end of deque;
// return item stored there
public Object removeLast() throws
                              DequeEmptyException;
```

b) [5 μονάδες] Συμπληρώστε τις παρακάτω διεπαφές Java για τον ΑΤΔ **RankedSeq** (**ΑκολουθίαμεΔείκτες**). Γράψτε σύντομα σχόλια που επεξηγούν κάθε μέθοδο.

```
public interface RankedSeq {
public int size();
public boolean isEmpty();
public Object first();
public void insertElemAtRank(int r, Object o);
}
```

```
Λύση: // Method insertElemAtRank() can throw a
// BoundaryViolationException
// return element at rank
public Object elemAtRank(int r) throws
                        BoundaryViolationException;
// replace object at r with o;
// return object previously at r
public void replaceElemAtRank(int r, Object o);
// remove item at rank r;
// return object previously at r
public Object removeElemAtRank(int r) throws
                        BoundaryViolationException;
```

c) [15 μονάδες] Υποθέστε τώρα οτι έχουμε στην διαθεσή μας την παρακάτω κλάση Java:

```
// This class implements all the methods of the
// Deque interface For convenience, it also provides
// the usual methods for stacks and queues, since
// those are easily implemented using a deque.
public class OurDeque implements Deque, Stack, Queue
{ ..... }
```

Γράψτε τον κώδικα της κλάσης Java που υλοποιεί τον ΑΔΤ **Ακολουθία μεΔείκτες** (Ranked Sequence ADT) χρησιμοποιώντας ένα η περισσότερα στιγμιότυπα της κλάσης **OurDeque**. Η κλάση σας θα πρέπει να περιλαμβάνει τουλάχιστον την μέθοδο **insertElemAtRank(r,o).**

Λύση: The following class identifies the top element of the **OurDeque**-stack with rank zero:

```
public class MyRankedSeq implements RankedSeq {
private OurDeque data, tmp;
private int size;
  public void insertElemAtRank(int r, Object o)
   throws BoundaryViolationException {
     try {
     if (r < 0 || r > data.size())
         throw new
    BoundaryViolationException("Invalid rank");
     for (int i = 0; i < r; i++)
        // remove top r elements,
        tmp.push(data.pop());
     // insert the new one
     data.push(o);
     for (int i = 0; i < r; i++)
        // put back top r elements
        data.push(tmp.pop());
     }
     catch (StackEmptyException e) {
     // should never occur:
     throw new Error("Internal error");
     }
  }
}
```