Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

ΗΥ-252 – Οντοκεντρικός Προγραμματισμός
Βασίλης Χριστοφίδης

Επαναληπτική Εξέταση (3 ώρες)
Ημερομηνία: 30 Αυγούστου 2003

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Άσκηση 1 (5 μονάδες)

Θεωρήστε τους παρακάτω ατελείς ορισμούς κλάσεων Java:

```
class Student { ... }
class Undergraduate extends Student { ... }
class Graduate extends Student { ... }
```

Τι συμβαίνει κατά την μετάφραση κάθε γραμμής του παρακάτω προγράμματος Java? Σε περίπτωση προβλήματος εξηγήστε την αιτία που το προκάλεσε.

```
/**1**/Student student1, student2;
/**2**/Graduate student3;
/**3**/student1 = new Undergraduate();
/**4**/student2 = new Graduate();
/**5**/student3 = student2;
/**6**/student3 = (Graduate) student2;
/**7**/student3 = (Graduate) student1;
```

Λύση:
1 OK
2 OK
3 OK
4 OK
5 compilation error
6 explicit cast, ok
7 compilation ok run-time exception

Άσκηση 2 (21 μονάδες)

Η ακόλουθη διεπαφή Java ορίζει το συμβόλαιο ενός ΑΤΔ για την αποθήκευση και ανάκτηση αντικειμένων (Objects) όπως μια δέσμη (deck) από τραπουλόχαρτα:

```
/** Interface describing the operations for a deck. */
 public interface Deck {

 /**
  * Adds the given element to the bottom of this deck.
  * @param element object to be added
  */
  public void add(Object element);

 /**
  * Inserts the given element at a random position
  * in this deck.
  * @param element object to be inserted
  */
   public void insert(Object element);

 /**
  * Removes the object on the top of this deck,
  * and returns that object.
  */
  public Object removeTop() throws RuntimeException;

 /** Returns the number of objects in this deck.
  */
  public int size();

 /** Returns whether or not this deck is empty.
  */
  public boolean isEmpty();
}
```

Γράψτε μια κλάση Java με όνομα **VectorDeck** η οποία υλοποιεί τον ΑΤΔ **Deck**. Για την υλοποίηση των μεθόδων της διεπαφής χρησιμοποιείστε ένα αντικείμενο τύπου **Vector**. Τεκμηριώστε τον κώδικά σας χρησιμοποιώντας κατάλληλα σχόλια Javadoc για όλες τις μεταβλητές και μεθόδους στιγμιότύπων που χρησιμοποιείτε.

Σημείωση: Για την υλοποίηση της μεθόδου **add()** θα χρειαστείτε τον παρακάτω κώδικα ο οποίος παράγει έναν τυχαίο ακέραιο αριθμό από το διάστημα [0, *n*):

```
(int)(Math.random() * n);
```

Λύση:

```
import java.util.*;
 /**
 * Implementation of the Deck interface using a Vector.
 * @author Vassilis Christophides
 */
public class VectorDeck implements Deck {

  /** Vector used to store elements. */
  protected Vector elements = new Vector();

  /** Constructs a new VectorDeck. */
  public VectorDeck() {}

  /**
   * Adds the given element to the bottom of this deck.
   * @param element object to be added
   */
```

```
    public void add(Object element) {
      elements.add(element);
    }

    /**
     * Inserts the given element at a random position
     * in this deck.
     * @param element object to be inserted
     */
     public void insert(Object element) {
      elements.add((int)(Math.random() * size(),
                          element));
     }

   /**
    * Removes the object on the top of this deck,
    * and returns that object.
    */
    public Object removeTop() throws RuntimeException {
          return elements.remove(0);
    }

   /** Returns the number of objects in this deck. */
    public int size() {
          return elements.size();
    }

   /** Returns whether or not this deck is empty. */
    public boolean isEmpty() {
          return elements.isEmpty();
    }
}
```

There is no need to grow and shrink the Vector, as that is what the class does for you automatically during add and remove operations. It is inappropriate to extend the Vector class because that exposes methods that could be used to circumvent the idea of the Deck as described in the interface. (For example, it allows removal of elements other than from the top of the deck.)
Note that the remove method used in removeTop will throw a runtime exception if the deck is empty. If you explicitly threw an exception in the proper case, that's fine too.

Άσκηση 3 (20 μονάδες)

Θεωρήστε τον παρακάτω ατελή ορισμό της κλάσης Java με όνομα **Student**, τον οποίον πρέπει να συμπληρώσετε :

```
// Student.java: student grade record
public class Student implements Comparable{
 private String name; //In the form "LastName, FirstName"
 private int totalHours;//total number of hours completed
 private int totalGradePoints; // GPA = totalGradePoints
                                       / totalHours
 private double GPA; // grade point average

 // uses name, total hours taken, and total gradepoints
 // as parameters
 public Student(String n, int initHours, int
                           initGradePoints) {
```

```
    // Fill in for Question 1.
  }
  // same as above, but replaces total gradepoints by
  // grade point average
  public Student(String n, int initHours, double initGPA){
       // Fill in for Question 2.
  }
  // return the GPA
  public double getGPA() {
     return GPA;
  }
  // assume the students takes a single course and gets a
  //grade
  public void updateRecord(int hours, int grade) {
   totalHours += hours; totalGradePoints += hours * grade;
   GPA = (double)totalGradePoints / totalHours;
  }
  public String toString() {
     return "Name: " + name + ", GPA: " + GPA +
            ", Total Hours: " + totalHours;
  }
  // compare names in alphabetic order.
  public int compareTo(Object t) {
       // Fill in for Question 4.
    }
}
```

1. **(3 μονάδες)** Συμπληρώσετε τον κώδικα για τον πρώτο κατασκευαστή αντικειμένων (constructor). Υπολογίστε τον μέσο όρο του βαθμού: **GPA = totalgradepoints / totalHours**.

Λύση:
```
name = n;
totalGradePoints = initGradePoints;
totalHours = initHours;
GPA = (double)totalGradePoints / totalHours;
```

2. **(3 μονάδες)** Συμπληρώσετε τον κώδικα για τον δεύτερο κατασκευαστή αντικειμένων (constructor). Υπολογίστε τον συνολικό βαθμό: **totalgradepoints = number of hours * GPA**.

Λύση:
```
name = n;
totalHours = initHours;
totalGradePoints = (int)(initGPA * totalHours);
GPA = initGPA;
```

3. **(1 μονάδα)** Στο παραπάνω πρόγραμμα σας δίνονται δύο κατασκευαστές αντικειμένων. Πως η εικονική μηχανή (VM) της Java καταλαβαίνει ποιον από τους δύο κατασκευαστές πρέπει να χρησιμοποιήσει?

Λύση:
By the constructor's arguments.

4. **(2 μονάδες)** Συμπληρώσετε τον κώδικα της μεθόδου `compareTo()` η οποία υλοποιεί την στάνταρ διεπαφή `Comparable`. Η μέθοδος συγκρίνει τα ονόματα των φοιτητών σε αλφαβητική σειρά.

Λύση:
```
return name.compareTo(((Student)t).name);
```

5. **(5 μονάδες)** Γράψτε έναν κώδικα σε μια ξεχωριστή κλάση ή μια μέθοδο `main()` ο οποίος δημιουργεί 4 αντικείμενα της κλάσης `Student` με τα παρακάτω δεδομένα :
   a. "Φοιτητής, Πρωτοετής", 20, 3.4
   b. "Προχωρημένος, Java", 35, 92
   c. "Πρωτάρης, Java", 25, 3.2
   d. "Φοιτητής, Τελειόφοιτος", 40, 145

Εκτυπώστε στο τέλος τα δεδομένα των αντικειμένων που δημιουργήσατε.

Λύση:
```
// Students.java: test the Student class
public class Students {
  public static void main (String[] args) {
  Student s1 = new Student("Φοιτητής,Πρωτοετής",
20,3.4);
  Student s2 = new Student("Προχωρημένος,Java",
35,92);
  Student s3 = new Student("Πρωτάρης,Java",25,3.2);
  Student s4 = new Student("Φοιτητής,Τελειόφοιτος",
40,145);
  System.out.println(s1 + "\n" + s2 + "\n" + s3 +
                     "\n" + s4);
 }
}
```

6. **(3 μονάδες)** Δείξτε πώς μπορείτε να καλέσετε την μέθοδο `getGPA()` για τον φοιτητή Java Πρωτάρης. Τι θα τυπωθεί στην έξοδο του προγράμματος μετά από την εκτέλεση αυτής της εντολής?

Λύση:
```
  System.out.println("GPA Java Πρωτάρης: " +
                     s3.getGPA());

GPA Java Πρωτάρης: 2.6285714285714286
```

7. **(3 μονάδες)** Δείξτε πώς μπορείτε να συγκρίνετε τον τελευταίο με τους δύο πρώτους φοιτητές από την παραπάνω λίστα χρησιμοποιώντας την μέθοδο `compareTo()`. Τι θα τυπωθεί στην έξοδο του προγράμματος μετά από την εκτέλεση αυτών των εντολών?

Λύση:
```
  System.out.println(s4.compareTo(s1));
  System.out.println(s4.compareTo(s2));

1
1
```

Άσκηση 4 (22 μονάδες)

Θεωρήστε το παρακάτω πρόγραμμα Java:

```
 1  interface Inter {
 2      int number();
 3  }
 4  abstract class Abs {
 5      static int foo = 12;
 6      int number() { return 5; }
 7      abstract int ace();
 8  }
 9  final class Sub extends Super {
10      Sub(int bar) { foo = bar; }
11      public int number() { return 10; }
12      int ace() { return 13; }
13      int dubble(int i) { return 2 * i; }
14  }
15  public class Super extends Abs implements Inter {
16      public int number() { return 11; }
17      public static void main(String args[]) {
18          Super s1 = new Super();
19          Super s2 = new Sub(16);
20          System.out.println(      );
21      }
22      int twice(int x) { return 2 * x; }
23      public int thrice(int x) { return 3 * x; }
24      int ace() { return 1; }
25      String dubble(String s) { return s + s; }
26  }
```

1. (**2 μονάδες**) Η κλάση `Super` προσφέρει μια σωστή υλοποίηση της διεπαφής `Inter`? Δώστε μια σύντομη εξήγηση.

> **Λύση:**
> Yes, because it defines `number()`

2. (**2 μονάδες**) Η παράληψη της δήλωσης της μεθόδου `number()` στην γραμμή 11 δημιουργεί ένα λάθος μετάφρασης . Γιατί?

> **Λύση:**
> Because `number()` in `Sub` overrides `number()` in `Super()`, and you cannot make an overridden function less public.

3. (**1 μονάδα**) Η μέθοδος στην γραμμή _____ υποσκελίζει (overrides) την μέθοδο που ορίζεται στην γραμμή _____. Συμπληρώστε τα κενά.

> **Λύση:**
> 24 and 12, *or* 11 and 16, *or* 16 and 6 (any one pair)

4. (**1 μονάδα**) Η μέθοδος στην γραμμή _____ υπερφορτώνει (overloads) την μέθοδο που ορίζεται στην γραμμή _____. Συμπληρώστε τα κενά.

Λύση:
13 and 25 (or 25 and 13)

5. (**1 μονάδα**) Για να εκτυπώσετε την μεταβλητή **foo** με την αρχική της τιμή 12 χρειαζόμαστε μια εντολή εκτύπωσης μεταξύ των γραμμών _____ και _____. Δώστε μια σύντομη εξήγηση.

Λύση:
17 and 19
must be in a method, and must be done before an instance of Sub is created in line 19

6. (**2 μονάδες**) Ποια από τις παραπάνω αφηρημένες ή συγκεκριμένες κλάσεις έχουν μόνο τους αρχικούς κατασκευαστές αντικειμένων?

Λύση:
Super (only)

7. (**1 μονάδα**) Ποιο είναι το αποτέλεσμα της δήλωσης της κλάσης **Sub** σαν final στην γραμμή 9?

Λύση:
Prevents any other class from extending this one

8. (**12 μονάδες**) Τι θα τυπωθεί στην γραμμή 20 για τις ακόλουθες παραμέτρους της **println()**?
   a. `s1.ace()`
   b. `s2.ace()`
   c. `s1.foo`
   d. `s2.foo`
   e. `s1.twice(3)`
   f. `s2.twice(3)`
   g. `s1.dubble(6)`
   h. `s2.dubble(7)`
   i. `s1.number()`
   j. `s2.number()`
   k. `s1.dubble("8")`
   l. `s2.dubble("9")`

Λύση:

```
s1.ace()  1
s2.ace()  13
s1.foo  16
s2.foo  16
s1.twice(3)  6
s2.twice(3)  6
s1.dubble(6) error
s2.dubble(7) error
s1.number()  11
s2.number()  10
s1.dubble("8")  88
s2.dubble("9")  99
```

Άσκηση 5 (22 μονάδες)

WordJumble
char[] letters
+ WordJumble(String s) + char getLetter(int i) + void jumble() – void swap(int i,int j) + String toString()

Γράψτε μια κλάση Java με όνομα **WordJumble** (ανακατωμένες λέξεις) η οποία υλοποιεί την διεπαφή που σας δίνεται γραφικά στην άσκηση σε συμβολισμό UML. Τα σύμβολα πριν από τις μεταβλητές στιγμιότύπων και τις μεθόδους της κάθε κλάσης υποδεικνύουν τα αντίστοιχα δικαιώματα εξουσιοδοτημένης πρόσβασης: συν (+) για **public**, πλην (-) για **private** και δίεση (#) για **protected**.

Ο κατασκευαστής αντικειμένων (constructor) της κλάσης παίρνει σαν είσοδο μια συμβολοσειρά (**String**) και αποθηκεύει με την σειρά τους χαρακτήρες του σε ένα καινούργιο πίνακα (**array**) από χαρακτήρες (**char**) που αναφέρεται απο την μεταβλητή στιγμιότύπων **letters**. Η εκ των προτέρων συνθήκη (precondition) για τον κατασκευαστή αντικειμένων της **WordJumble** είναι ότι η συμβολοσειρά εισόδου δεν πρέπει να είναι κενή (**null**), και αν αυτή η συνθήκη παραβιάζεται πρέπει να δημιουργείται μια **RuntimeException** με κατάλληλο μήνυμα λάθους.

Η μέθοδος **getLetter(i)** πρέπει να επιστρέφει τον χαρακτήρα που βρίσκεται στην θέση **i**. Η εκ των προτέρων συνθήκη (precondition) για την μέθοδο είναι ότι ο δείκτης εισόδου πρέπει να είναι έγκυρος σε σχέση με τα περιεχόμενα του πίνακα χαρακτήρων, και αν αυτή η συνθήκη παραβιάζεται πρέπει να δημιουργείται μια **RuntimeException** με κατάλληλο μήνυμα λάθους.

Η μέθοδος **jumble()** πρέπει να υλοποιεί τον παρακάτω αλγόριθμο:

```
for each index i in the array named letters:
     choose an index j in the range [0, letters.length)
          swap the values in letters at indices i and j
```

Δεν χρειάζεται να εξασφαλίσετε ότι οι δείκτες *i* και *j* είναι διαφορετικοί.

Η μέθοδος **swap(i,j)** πρέπει να αντιμεταθέτει τους χαρακτήρες της μεταβλητής **letters** που βρίσκονται στις θέσεις *i* και *j*. Δεν υπάρχουν εκ των προτέρων συνθήκες για την μέθοδο **swap.**

Η μέθοδος **toString()** πρέπει να επιστρέφει μια συμβολοσειρά (String) η οποία περιέχει όλους τούς χαρακτήρες του πίνακα **letters** και με την ίδια σειρά.

Σημείωση: Ο βασικός τύπος `char` αναπαριστά έναν χαρακτήρα: letter, number, punctuation mark, &c. Χειριστείτε τους χαρακτήρες σαν ακεραίους (int). Για την υλοποίηση της άσκησης θα χρειαστείτε τις μεθόδους της κλάσης `String toCharArray()` και `length()`. Για την υλοποίηση της μεθόδου `jumble()` θα χρειαστείτε τον παρακάτω κώδικα ο οποίος παράγει έναν τυχαίο ακέραιο αριθμό από το διάστημα [0, *n*):

```
(int)(Math.random() * n);
```

Τέλος μην παραλείψετε να τεκμηριώσετε τον κώδικά σας χρησιμοποιώντας κατάλληλα σχόλια Javadoc για όλες τις μεταβλητές και μεθόδους στιγμιότύπων που χρησιμοποιείτε.

Λύση:

```
/**
 * Represents text that can be jumbled.
 * @author Vassilis Christophides
 */
public class WordJumble {

 /** Letters in this. */
  protected char[] letters;

 /**
  * Constructs a new word jumble.
  * @param s String containing the text for the word
  * jumble
  */
  public WordJumble(String s) {
   if (s == null) {
     throw new RuntimeException("String is null");
    }
   letters = s.toCharArray();
  }
 /**
  * Returns the letter in this at the given index.
  * @param i index of letter
  * @example (new WordJumble("Fox")).getLetter(-1)
  * is an error
  * @example (new WordJumble("Fox")).getLetter(1)
  * returns 'o'
  * @example (new WordJumble("Fox")).getLetter(3) is
  * an error
  */
  public char getLetter(int i) {
   if ((i < 0) || (i >= letters.length)) {
      throw new RuntimeException("Index out of
      bounds");
    }
    return letters[i];
   }
  /**
    * Jumbles the letters in this.
    * @example (new WordJumble("Grape")).jumble()
    * might result in letters in the order "aeGpr"
    */
   public void jumble() {
```

```
      for (int i = 0; i < letters.length; i++) {
        swap(i, (int)(Math.random()*letters.length));
       }
    }
   /**
    * Swaps the letters in this at the given indices.
    * @param i first index of letters to swap
    * @param j other index of letters to swap
    * @example (new WordJumble("Box")).swap(0, 2)
    * results in letters in the order "xoB"
    * @example (new WordJumble("Box")).swap(1, 1)
    * results in letters in the order "Box"
    */
    private void swap(int i, int j) {
      char temp = letters[i];
      letters[i] = letters[j];
      letters[j] = temp;
    }
   /** Returns a String version of this. */
    public String toString() {
      return new String(letters);
    }
}
```

Άσκηση 6 (10 μονάδες)

Συμπληρώστε με σύντομες απαντήσεις τα κενά στις παρακάτω προτάσεις:

1. Ένα αντικείμενο Java μπορεί να είναι στιγμιότυπο μόνο μιας κλάσης αλλά μπορούμε να το χειριστούμε σαν τιμή ενός ή περισσοτέρων **τύπων** .

2. Οι τρεις βασικές αφαιρέσεις (abstractions) δεδομένων με τις οποίες μπορούμε να ορίζουμε νέους τύπους αντικειμένων Java είναι: **class** , **interface** , και **abstract class** .

3. Ο βασικός τύπος `byte` στην Java αναπαριστά έναν ακέραιο στο διάστημα [-128, 128). Η κλάση Java με το όνομα `Byte` ορίζεται με μια μεταβλητή στιγμιότύπων τύπου `byte`. Ο αριθμός των αντικειμένων Byte που μπορούμε να δημιουργήσουμε σε ένα πρόγραμμα Java καθορίζεται μόνο οπό την **μνήμη της μηχανής που τρέχει το πρόγραμμα**.

4. Μια κλάση Java μπορεί να οριστεί ότι **implement** μία ή περισσότερες διεπαφές, αλλά μια κλάση μπορεί να **extend** μόνο μια άλλη κλάση.

5. Υποσημειώστε ποια από τα ακόλουθα μέρη του ορισμού μιας κλάσης συνεισφέρουν στην δήλωση της δημόσιας διεπαφής της κλάσης:
   a. public data members
   b. protected data members
   c. public constructors
   d. private methods