Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

ΗΥ-252 – Αντικειμενοστρεφής Προγραμματισμός
Βασίλης Χριστοφίδης

Επαναληπτική Εξέταση (3 ώρες)
Ημερομηνία: 31 Αυγούστου 2007

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Άσκηση 1 (10 μονάδες) Υπερφόρτωση και Υποσκελισμός Μεθόδων

Σας δίνεται ο παρακάτω κώδικας των κλάσεων Java, `ClassA`, `ClassB`, `ClassC` και `TestClass`. Τι θα γραφτεί στην στάνταρ έξοδο κατά την εκτέλεση της μεθόδου `main()` της `TestClass`; Υποθέστε ότι όλες οι κλάσεις μεταφράζονται και εκτελούνται χωρίς λάθη.

```
public abstract class ClassA {
      private int num1;
      private int num2;
      public ClassA (int n1, int n2) {
            num1 = n1;
            num2 = n2;
      }
      public abstract int sumNums();
      public abstract int subNums();
      public abstract int subNums(char c);
      public int timesNums() {
            return num1 * num2;
      }
      public int getNum1 () { return num1; }
      public int getNum2 () { return num2; }
} // end ClassA

public class ClassB extends ClassA {
      private int num1;
      public ClassB (int n1, int n2, int n3) {
            super(n1, n2);
            num1 = n3;
      }
      public ClassB (int n) {
            super(n, n + 1);
            num1 = n;
      }
      public int sumNums () {
            return num1 + getNum1() + getNum2();
      }
      public int timesNums() {
            return super.timesNums() * num1;
      }
      public int subNums () {
            return num1 - super.getNum2() - super.getNum1();
      }
```

```
      public int subNums (char dir) {
            if (dir == 'f')
                  return super.getNum1()-super.getNum2()-num1;
            else
                  return subNums();
      }
      public int getNum1 () { return num1; }
} // end ClassB

public class ClassC extends ClassB {
      private int num1;
      public ClassC (int n1, int n2, int n3, int n4) {
            super(n1, n2, n3);
            num1 = n4;
      }
      public ClassC (int n) {
            super(n);
            num1 = n;
      }
      public int timesNums() {
            return super.timesNums() * num1;
      }
} // end ClassC

public class TestClass {
 public static void main (String[] args) {
      ClassA b = new ClassB(2, 3, 4);
      System.out.println(b.getNum1());
```

Λύση: 4

```
      System.out.println(b.sumNums());
```

Λύση: 11

```
      System.out.println(b.timesNums());
```

Λύση: 24

```
      System.out.println(b.subNums('f'));
```

Λύση: -5

```
      System.out.println(b.subNums());
```

Λύση: -1

```
      ClassA c = new ClassC(3);
      System.out.println(c.timesNums());
```

Λύση: 108

```
      System.out.println(c.sumNums());
```

Λύση: 10

```
      System.out.println(c.getNum1());
```

Λύση: 3

```
      System.out.println(c.getNum2());
```

Λύση: 4

```
      ClassA cc = new ClassC(5, 6, 7, 8);
      System.out.println(cc.sumNums());
```

Λύση: 20

```
 } // end main
} // end TestClass
```

Άσκηση 2 (25 μονάδες) Κληρονομικότητα Κλάσεων και Διεπαφές

Σας δίνεται η παρακάτω διεπαφή Shape σαν μέρος ενός γεωμετρικού πακέτου (package) Java:

```
/**A two-dimensional shape on the plane, with Cartesian
coordinates*/
interface Shape {
// These four methods give coordinates that bound the shape in a
// rectangle.
float leftX();
float rightX();
float bottomY();
float topY();
/** The area (εμβαδόν) of the shape. */
float area();
}
```

Σε αυτό το πακέτο, θέλουμε να υλοποιήσουμε διάφορα γεωμετρικά σχήματα. Για παράδειγμα, ο παρακάτω κώδικας περιλαμβάνει τις κλάσεις για τον χειρισμό παραλληλόγραμμων (Rectangle) και κύκλων (Circle).

```
class Rectangle implements Shape {
      private float x0, x1, y0, y1;
      public Rectangle(float x0, float x1, float y0, float y1) {
          this.x0 = x0; this.x1 = x1; this.y0 = y0; this.y1 = y1;
      }
      public float leftX() { return x0; }
      public float rightX() { return x1; }
      public float bottomY() { return y0; }
      public float topY() { return y1; }
      public float area() { return (x1 - x0) * (y1 - y0); }
}
class Circle implements Shape {
      private float center_x, center_y, radius;
      public Circle(float cx, float cy, float r) {
          center_x = cx; center_y = cy; r = radius;
      }
      public float leftX() { return center_x - radius; }
      public float rightX() { return center_x + radius; }
      public float bottomY() { return center_y - radius; }
      public float topY() { ... }
      public float area() { return Math.PI*radius*radius; }
}
```

α) **(3 μονάδες)** Υλοποιήστε την μέθοδο Circle.topY(). Είναι ένας κατασκευαστής (creator), παρατηρητής (observer), ή μεταλλάκτης (mutator)?

> Λύση:
> public float topY() { return center_y + radius; }
> Observer.

β) **(10 μονάδες)** Υλοποιήστε μια κλάση Square (τετράγωνο) η οποία κληρονομεί την κλάση Rectangle αλλά έχει την ακόλουθη μέθοδο κατασκευής Square(float center_x, float center_y, float size).

Λύση:
```
class Square extends Rectangle {
     Square(float cx, float cy, float size) {
          super(cx - size/2, cx + size/2,
                cy - size/2, cy + size/2);
     }
}
```

γ) **(12 μονάδες)** Υλοποιήστε μια κλάση `Annulus` (δακτύλιος) η οποία κληρονομεί την κλάση `Circle` αλλά έχει την ακόλουθη μέθοδο κατασκευής `Annulus(float cx,float cy,float outer_radius,float inner_radius)` Θυμηθείτε ότι ένας δακτύλιος είναι η περιοχή που βρίσκεται ανάμεσα σε δύο ομόκεντρους (concentric) κύκλους. Ποιες μεθόδους, εάν υπάρχουν, της κλάσης `Circle` θα πρέπει να υποσκελίσουμε (overridden)?

Λύση:
We need to override `area()`.
```
class Annulus extends Circle {
     float inner_radius;
     Annulus(float cx, float cy, float outer_radius,
             float inner_radius) {
             super(cx, cy, outer_radius);
             this.inner_radius = inner_radius;
     }
     float area() {
             return super.area() –
                    PI*inner_radius*inner_radius;
     }
}
```

Άσκηση 3 (10 μονάδες) Βασική Λειτουργικότητα Αντικειμένων

Υλοποιήστε την μέθοδο `equals()` της κλάσης `String`. Μπορείτε να χρησιμοποιήσετε τις ακόλουθες μεθόδους της `String`:

- `char charAt(int index)` επιστρέφει τον χαρακτήρα στην συγκεκριμένη θέση
- `int length()` επιστρέφει το μήκος της συμβολοσειράς
- Ο δείκτης θέσης αριθμείται από 0.

Η υλοποίησή σας θα πρέπει να επιστρέφει μια τιμή ακόμα και όταν το όρισμα δεν είναι τύπου `String`.

Λύση:
```
public boolean equals(Object obj) {
     if (!(obj instanceof String)) return false;
     String s = (String)obj;
     if (length() != s.length()) return false;
     for (int i = 0; i < length(); i++) {
         if (charAt(i) != s.charAt(i)) return false;
     }
     return true;
}
```

Άσκηση 4 (10 μονάδες) Αφαιρετικοί Τύποι Δεδομένων (ΑΤΔ)

Η τιμή ενός εμπορεύματος (item) προς πώληση δίνεται σε ευρώ *priceEuros* και σε σέντς *priceCents*. Για την πληρωμή του σε μετρητά θα πρέπει να δώσετε *payEuros* ευρώ και *payCents* σέντς. Γράψτε το συμβόλαιο (δηλ. τις εκ των προτέρων και εκ των υστέρων συνθήκες) της μεθόδου `getChange()` που υπολογίζει τα ρέστα που, ενδεχομένως, θα πρέπει να σας επιστραφούν. Το σωστό υπόλοιπο θα πρέπει να υπολογίζεται σαν τιμή επιστροφής της `getChange()` με τύπο αντικείμενα της κλάσης `Change` (με μεταβλητές στιγμιοτύπων *changeEuros* και *changeCents*). Στην προδιαγραφή της μεθόδου θα πρέπει να συμπεριλάβετε και την περιγραφή των ορισμάτων που χρησιμοποιούνται. Διατυπώστε με σαφήνεια τη σχέση μεταξύ των ορισμάτων εισόδου και εξόδου της μεθόδου.

Λύση:
Purpose: to compute the change received by paying priceEuros and priceCents for an item that costs priceEuros and priceCents.
`Change getChange(int priceEuros, int priceCents, int payEuros, int payCents)`
Pre-conditions:100*`priceEuros`+ `priceCents` < 100*`payEuros`+ `payCents`
All are arguments >= 0, `priceCents` < 100.
Post-conditions:
100 * `changeEuros` + `changeCents` + 100 * `priceEuros` + `priceCents` = 100 * `payEuros` + `payCents`, 0 <= `changeCents` < 100
Parameters: priceEuros and priceCents describe the price of the item, `payEuros` and `payCents` describe the amount paid `changeEuros` and `changeCents` describe the amount of change received

Άσκηση 5 (10 μονάδες) Υλοποιήσεις Πλαισίου Συλλογών Αντικειμένων

Σας δίνεται η ακόλουθη διεπαφή Java `Iterator`:

```
Public interface Iterator {
 boolean hasmore(); //For testing the traversing of a collection
 void next(); //Advance to next element of collection
 Object current(); //Returns the current element of collection
 Void reset(); //Restart at the beginning of collection
 Iterator spawn(); //Create new iterator with the same current
                   //element
```

Γράψτε τον κώδικα μιας κλάσης `myIterator` η οποία υλοποιεί την διεπαφή `Iterator` για πίνακες τύπου `Object[]`. Ο κατασκευαστής της κλάσης παίρνει σαν όρισμα μια αναφορά στον πίνακα, τα στοιχεία του οποίου θέλουμε να σαρώσουμε.

Λύση:
```
class myIterator implements Iterator { //1
  Object [] theArray;
  Int cursor;
  Iterator (Object[] oA) {
    theArray = oA;
    cursor = 0;
```

```
  }//2
  public Boolean hasmore() {
    return (cursor < theArray.length);
  }//1
  public void next() {
    cursor++
  }//1
  public Object current() {
    return theArray[cursor];
  }//1
  public void reset() {
    cursor=0;
  }//1
  Public Iterator spawn() {
    myIterator it = new myIterator(theArray);
    It.cursor = cursor;
    Return it;
  }//3
```

Άσκηση 6 (15 μονάδες) Συγκρίσεις Αντικειμένων

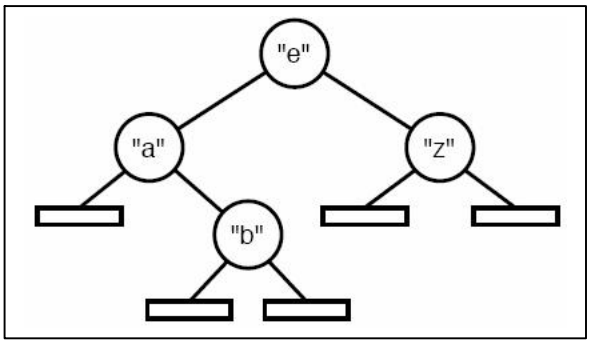
Σας δίνεται η παρακάτω κλάση SearchTree για την αναπαράσταση ενός δυαδικού δέντρου αναζήτησης. Θυμηθείτε ότι το αριστερό (δεξί) υποδέντρο έχει τιμές μικρότερες (μεγαλύτερες) του κλειδιού κάθε κόμβου.

```
1 public class SearchTree {
2   private SearchTree left, right;
3   private Comparable key;
4   private SearchTree(SearchTree left, Comparable key,
5                                          SearchTree right) {
6     this.key = key;
7     this.left = left;
8     this.right = right;
9   }
10  public static SearchTree newTree(SearchTree left,
11                          Comparable key, SearchTree right) {
12    if (left != null && left.max().compareTo(key) >= 0 ||
13        right != null && right.min().compareTo(key) <= 0 )
14        throw new IllegalArgumentException(
15                                  "ill-formed search tree");
16    return new SearchTree(left, key, right);
17  }
18  public boolean lookup(Comparable key) {
19    int c = key.compareTo(this.key);
20    if (c < 0)
21       return left != null ? left.lookup(key) : false;
22    if (c > 0)
23       return right != null ? right.lookup(key) : false;
24    return true;
25  }
26  public void insert(Comparable key) {
27    int c = key.compareTo(this.key);
28    if (c < 0) {
29       if (left == null)
30          left = new SearchTree(null, key, null);
31       else
32         left.insert(key);
33    }
```

```
34    else if (c > 0) {
35       if (right == null)
36         right = new SearchTree(null, key, null);
37       else
38         right.insert(key);
39      }
40  }
41  public SearchTree delete(Comparable key) {
42    int c = key.compareTo(this.key);
43    if (c < 0) {
44       if (left != null)
45           left = left.delete(key);
46       return this;
47       }
48    if (c > 0) {
49       if (right != null)
50           right = right.delete(key);
51       return this;
52      }
53    if (left == null) return right;
54    if (right == null) return left;
55    left = left.raiseMax(this);
56    return this;
57  }
58  private SearchTree raiseMax(SearchTree target) {
59    if (right != null) {
60        right = right.raiseMax(target);
61    return this;
62    }
63    target.key = this.key;
64    return left;
65  }
66  public Comparable min() {
67    if (left != null)
68        return left.min();
69    return key;
70  }
71  public Comparable max() {
72    if (right != null)
73        return right.max();
74    return key;
75  }
76}
```

α) **(5 μονάδες)** Η παρακάτω μέθοδος `getElements(List list)` της κλάσης `SearchTree` αποσκοπεί στο να προσθέτει σε αύξουσα διάταξη όλα τα στοιχεία ενός υπο-δέντρου σε μια (non-null) λίστα που δίνεται σαν όρισμα. Για παράδειγμα, για το δυαδικό δέντρο του ακόλουθου παραδείγματος, η λίστα θα περιέχει τα στοιχεία "a" "b" "e" "z" (με αυτή την σειρά).
Σημείωση: Χρησιμοποιήστε την μέθοδο `add()` της διεπαφής `List`.

```
public void getElements(List list)
{
      right.getElements(list);
      list.add(key);
      left.getElements(list);
}
```



Δυστυχώς υπάρχουν 3 τουλάχιστον λάθη (λογικά ή εκτέλεσης). Εντοπίστε τα λάθη και διορθώστε αντίστοιχα την υλοποίηση της μεθόδου.

Λύση:

```
public void getElements(List list) {
 if (left != null) left.getElements(list);
 list.add(key);
 if (right != null) right.getElements(list);
}
```

+1 if each element is added to list exactly once
+1 if no null pointers exceptions would occur
+2 if elements are added in order
+1 if Binary Search Tree used correctly
-1 if original tree is changed, a specific data type is assumed for key (other than `Comparable`), method header is changed or wrong syntax

β) **(10 μονάδες)** Συμπληρώστε την παρακάτω υλοποίηση της μεθόδου `range(Comparable low, Comparable high, List list)` της κλάσης `SearchTree`, η οποία προσθέτει στην λίστα που δίνεται σαν όρισμα όλα τα κλειδιά του δέντρου που είναι μεγαλύτερα ή ίσα από το `low` και μικρότερα ή ίσα από το `high`:

```
public void range(Comparable low, Comparable high, List list) {
   // (1) test if key is greater than or equal to low and less or
   // equal to high
```

Λύση:

```
if (key.compareTo(low)>= 0&&key.compareTo(high)<= 0) {
```

```
  // (2) what to do in this case
```

Λύση:

```
  list.add(key);
  if (left !=null) left.range(low,high, list);
  if (right != null) right.range(low, high, list);
}
```

```
  // (3) what to do if key less than low or greater than high
```

Λύση:

```
else {
   if (key.compareTo(low) < 0 && right != null)
       right.range(low, high, list);
    else if (left != null)
       left.range(low, high, list);
}
```

+1 for using `compareTo()` in if statements (not <= or >=)
+1 for correct comparisons in if statements
+2 for adding the key to the list at the right place
+3 for correct recursion in if-case (partial credit given)
+3 for correct recursion in else-case (partial credit given)

```
}
```

Άσκηση 7 (25 μονάδες) Σχεδίαση και Υλοποίηση Κλάσεων Αντικειμένων

Υλοποιήστε μια κλάση Car η οποία χρησιμοποιείται από την κλάση UsesCar. Από τον παρακάτω κώδικα της μεθόδου main(), θα πρέπει να βρείτε ποιες μεταβλητές και μεθόδους στιγμιότυπων της κλάσης Car είναι χρήσιμες και ποια η λειτουργικότητά τους. Θα πρέπει επίσης να χρησιμοποιήσετε την κατάλληλη απόκρυψη πληροφοριών (δηλώσεις "private" and "public"). Δεν πρέπει να συμπεριλάβετε μεθόδους που δεν είναι απαραίτητες για τη σωστή εκτέλεση της μεθόδου main() της κλάσης UsesCar. Για ευκολία οι μέθοδοί σας δεν είναι απαραίτητο να κάνουν οποιοδήποτε είδος ελέγχου λαθών.

```
public class UsesCar {
 public static void main(String args[]) {
  Car redCompact = new Car("red", 2);
  System.out.println(redCompact); //should print "red, 2 doors"
  Car greenVan = new Car("green");
  System.out.println(greenVan); //should print "green, 4 doors"
  System.out.println(greenVan.getColour());//should print "green"
  // The colourChangeCount method should return the number of
  // times that any car has changed its colour (i.e. the total
  // for all cars created in this program).
  System.out.println(Car.colourChangeCount()); //should print 0
  greenVan.setColour("yellow");
  System.out.println(greenVan.getColour());//should print"yellow"
  System.out.println(greenVan); //should print "yellow, 4 doors"
  System.out.println(Car.colourChangeCount());//should print 1
  redCompact.setColour("black");
  redCompact.setColour("red");
  System.out.println(Car.colourChangeCount());//should print 3
  Car yellowVan = new Car("yellow", 4);
  if (yellowVan.equals(greenVan))//if statement should print"yes"
      System.out.println("yes");
  else
     System.out.println("no");
  } // end main
} // end class UsesCar
```

Λύση:

```
public class Car { // 1 point for class header
 private int numDoors;//1 point for declaration
 private String colour;//1 point for declaration
 private static int numColourChanges = 0; //1 point for
 //declaration, must be static
 public Car(String col,int doors){//2 point for constructor
  colour = col; //or setColour(col); numColourChanges--;
  numDoors = doors;
 } // end constructor
 public Car(String col){//3 points for constructor,
                        // including the default of 4 doors
  this(col, 4); //or setColour(col); numDoors = 4;
                //numColourChanges--;
 } // end constructor
 public String toString() { // 3 points
  return colour + ", " + numDoors + " doors";
 } // end toString
```

```
 public void setColour(String col) {//2 point for basic
 // method (not counting incrementing counter)
  colour = col;
  numColourChanges++;
 } // end setColour
 public String getColour() { // 1 point
  return colour;
 } // end getColour
 //4 points total for this method. A good equals method
 //takes an Object as a parameter, so that the method
 //overrides the equals method in the Object class. If your
 //equals method takes a Car as a parameter, the example
 //will work fine but some kinds of general code won't work
 //properly. I didn't really stress this in class so no
 //deduction if you used a Car as a parameter and one bonus
 //point if you used an Object properly.
 public boolean equals(Object other) {
  if (other instanceof Car) {
    Car otherCar = (Car) other;
    //Accessing otherCar.numDoors in the following
    // statement is legal, even though numDoors is private.
    // Many people wrote a getNumDoors method to avoid
    // this. Legal but unnecessary.
    return (otherCar.numDoors == numDoors &&
            colour.equals(otherCar.colour));
    // A distressing number of people tested
    // otherCar.numDoors.equals(numDoors). You can't call
    // equals (or any other method) for an int. Use "=="
    // for primitive values and equals for objects.
  }
  else {
    return false;
  } // end if
 } //end equals
 //1 point for declaring this method properly, including
 // "static". 2 more points total for the logic:
 // counter incremented in setColour and returned here.
 public static int colourChangeCount() {
  return numColourChanges;
 } // end colourChangeCount
} // end class Car
// 2 points for information hiding: variables private (or
//protected) and required methods public
```

Άσκηση 8 (20 μονάδες) Πολυπλοκότητα Κώδικα

Σας δίνονται οι τρεις παρακάτω μέθοδοι που υπολογίζουν το x^n για κάποιο $n \geq 0$ ($x^0 = 1$):

α) **(5 μονάδες)** Επαναληπτική υλοποίηση

```
public static int power1(int x, int n) {
      int val = 1;
      for (int i=0; i<n; i++)
            val *= x;
      return val;
}
```

β) **(5 μονάδες)** Αναδρομική υλοποίηση που βασίζεται στην φόρμουλα $x^n = x * x^{n-1}$ εάν n > 0

```
public static int power2(int x, int n) {
      if (n <= 0) return 1;
      else return x * power2(x, n-1);
}
```

γ) **(10 μονάδες)** Αναδρομική υλοποίηση που βασίζεται στην φόρμουλα $x^n = (x^{n/2})^2$ εάν n > 0 και n είναι ζυγός ενώ $x^n = x * (x^{n/2})^2$ εάν n > 0 και n είναι περιττός.

```
public static int power3(int x, int n) {
      if (n <=0) return 1;
   // This line necessary only if you use power(power(x, n/2), 2);
      if (n==2) return x * x;
      int tmp = power3(x, n/2);
      if (n%2 == 0) return tmp * tmp;
      else return x * tmp * tmp;
}
```

Ποια είναι η πολυπλοκότητα (complexity) του χρόνου εκτέλεσης (στην χειρότερη περίπτωση) των παραπάνω μεθόδων; Δώστε μια σύντομη εξήγηση.

Λύση:
power1 is O(n), since the loop executes n times and everything else is O(1).
power2 is O(n), since there are n recursive calls and everything else is O(1).
power3 is O(log n), since the problem size is reduced in half at each iteration.
T(n) = O(1) + T(n/2) = O(1) + O(1) + T(n/4) …
Eventually, the problem ends when the problem size is small enough, which is after log n steps.
Using iteration, we get:
T(n) = k + T(n/2^k)
T(n) = log n + T(1)
T(n) is O (log n)