Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

ΗΥ-252 – Αντικειμενοστρεφής Προγραμματισμός
Βασίλης Χριστοφίδης

Πρόοδος (3 ώρες)
Ημερομηνία: 2 Δεκεμβρίου 2006

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Άσκηση 1 (15 μονάδες) Συμβολοσειρές και Αρχεία

Δώστε τον κώδικα της μεθόδου `main()` μιας κλάσης `LineAndColumn` που βρίσκει την πρώτη εμφάνιση ενός χαρακτήρα σε ένα αρχείο κειμένων και εκτυπώνει τον αριθμό της γραμμής καθώς και τη θέση μέσα στη γραμμή που ο χαρακτήρας εμφανίζεται. Το πρόγραμμα θα πρέπει να παίρνει δύο παραμέτρους από την κονσόλα, όπου η πρώτη είναι το όνομα του αρχείου και η δεύτερη ο χαρακτήρας που αναζητείται. Το πρόγραμμα θα πρέπει να επιστρέφει κατάλληλα μηνύματα λάθους εάν παίρνει έναν λανθασμένο αριθμό παραμέτρων, ή εάν η δεύτερη παράμετρος είναι μια συμβολοσειρά μεγαλύτερη από έναν χαρακτήρα. Επίσης, θα πρέπει να πληροφορεί για τυχόν προβλήματα ανάγνωσης/ εγγραφής.

Για παράδειγμα, θεωρήστε ένα αρχείο κειμένου με όνομα `test.txt` που περιέχει ακριβώς τις ακόλουθες δύο γραμμές:
```
jw3Xxv44js
snnnms8@@lvdu
```

Επιπλέον, υποθέστε ότι δεν υπάρχει κανένα αρχείο κειμένου με όνομα `xxx.txt` στον τρέχοντα κατάλογο (directory). Στην συνέχεια σας δίνεται η επιθυμητή συμπεριφορά του προγράμματος που πρέπει να υλοποιήσετε στα δεδομένα του παραπάνω παραδείγματος:

- `java LineAndColumn test.txt j`
  εκτυπώνει:
  `the character j appears first in line 1 at position 0`
- `java LineAndColumn test.txt m`
  εκτυπώνει:
  `the character m appears first in line 2 at position 4`
- `java LineAndColumn test.txt a`
  εκτυπώνει:
  `the character a does not appear in the file`
- `java LineAndColumn test.txt`
  εκτυπώνει:
  `wrong number of arguments`
- `java LineAndColumn test.txt bb`
  εκτυπώνει:
  `second argument must be a single char`
- `java LineAndColumn xxx.txt j`
  εκτυπώνει:
  `input/output problem`

Σημείωση: Μπορείτε να υποθέσετε ότι το πρόγραμμα εκτελείται στον ίδιο κατάλογο όπου βρίσκεται και το αρχείο κειμένου που δίνεται σαν παράμετρος. Επίσης δεν έχει σημασία εάν το πρόγραμμά σας θεωρεί ότι η πρώτη γραμμή του αρχείου έχει αριθμό 0 ή 1 (το ίδιο ισχύει και για την θέση ενός χαρακτήρα σε μία γραμμή). Για την υλοποίηση της μεθόδου `main()` χρησιμοποιήστε τις μεθόδους `readLine()` και `getLineNumber()`της κλάσης `LineNumberReader` καθώς και τις μεθόδους `charAt(i)` και `indexOf(c)` της `String`.

Λύση:

```
import java .io .*;
public class LineAndColumn {
 public static void main (String[] args) {
  if (args.length != 2) {
      System.out.println("wrong number of arguments");
      return;
   }
  if ( args[1].length()!= 1 ) {
      System.out.println("second argument must be a single char");
      return;
  } //2 points
String file = args[0] ;
char toFind = args[1].charAt(0) ;
String line;
int indexInLine ;
boolean found = false ;
try {
  LineNumberReader input=new LineNumberReader(new FileReader(file));
  while (((line = input.readLine()) != null ) && ! found ) {
        indexInLine = line.indexOf(toFind );
        if (indexInLine != -1) {
            found = true ;
            System.out .println ("the character " + toFind +
            "appears first in line " +input.getLineNumber() +
            "at position " + indexInLine );
            }
     }
   if (!found)
      System.out println ("the character " + toFind +
                          " does not appear in the file" );
   input.close();
  } //8 points
  catch (IOException e ) {
     System.out.println ("input/output problem");
  } //4 points
 }
}
```

Άσκηση 2 (10 μονάδες) Βασική Λειτουργικότητα Αντικειμένων

Σας δίνεται ο ακόλουθος μερικός ορισμός της κλάσης `BinaryTree`:

```
public class BinaryTree {
   BinaryTree leftChild, rightChild;
   Object value;
   ...   }
```

Υλοποιήστε μια μέθοδο στιγμιοτύπων (instance method) `equals()` για να ελέγχετε εάν δύο δυαδικά δένδρα (binary trees) έχουν την ίδια δομή (δηλ. ίδιο αριθμό κόμβων ανά επίπεδο) και τις ίδιες τιμές ανά κόμβο.

```
public boolean equals(BinaryTree that) {
```

Λύση:

```
   if (that == null) return false;
   return safeEquals(this.value, that.value) &&
          safeEquals(this.leftChild, that.leftChild) &&
          safeEquals(this.rightChild, that.rightChild);
}
private boolean safeEquals(Object obj1, Object obj2) {
   if (obj1 == obj2) return true;
   if (obj1 == null || obj2 == null) return false;
   return obj1.equals(obj2);
}
```

Σημείωση: Για την υλοποίηση της βαθιάς ισότητας (deed equality) των αντικειμένων που συνθέτουν ένα δυαδικό δένδρο (που περιλαμβάνει και τον έλεγχο null τιμών) θα χρειαστείτε μια βοηθητική μέθοδο `boolean safeEquals(Object obj1, Object obj2)`.

Άσκηση 3 (16 μονάδες) Υπερφόρτωση και Υποσκελισμός Μεθόδων

Σας δίνονται οι ακόλουθες εντολές ενός προγράμματος Java:
`A a = new A("a");`
`a.name("b")`
Για κάθε έναν από τους παρακάτω ορισμούς κλάσεων, δώστε την συμβολοσειρά που θα τυπωνόταν στην έξοδο σαν επιστρεφόμενη τιμή μετά την εκτέλεση της μεθόδου `a.name("b")`.

(α) **(4 μονάδες)**
```
class C {
      String n = "c";
      C(String n) { this.n = n; }
      String name(String s) { return n+s; }
}
class A extends C {
      A(String n) { super(n); }
}
"ab" "ac" "ba" "bc" "ca" "cb"
```

Λύση: ab

(β) **(4 μονάδες)**
```
class C {
      private String n = "c";
      C() { }
      C(String n) { this.n = n; }
      String name(String s) { return n+s; }
}
class A extends C {
      String n;
      A(String n) { this.n = n; }
}
"ab" "ac" "ba" "bc" "ca" "cb"
```

Λύση: cb

(γ) **(4 μονάδες)**
```
class C {
      String n = "c";
      C() { }
      C(String n) { this.n = n; }
      String name(String s) { return n+s; }
}
class A extends C {
      String n = "a";
      A(String n) { this.n = n; }
      String name(String s) { return s+n; }
}
"ab" "ac" "ba" "bc" "ca" "cb"
```

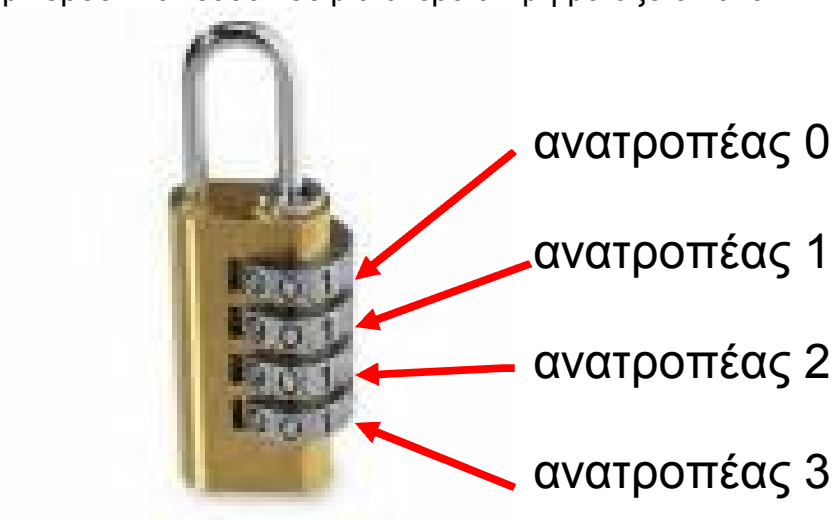
Λύση: ba

(δ) **(4 μονάδες)**
```
class C {
      String n = "c";
      C() { }
      C(String n) { this.n = n; }
      String name(String s) { return n+s; }
}
class A extends C {
      String n = "a";
      A(String n) { this.n = n; }
      String name(String s) { return super.name(n); }
}
"ab" "ac" "ba" "bc" "ca" "cb"
```

Λύση: ca

Άσκηση 4 (20 μονάδες) Σχεδίαση και Υλοποίηση κλάσεων

Σχεδιάστε και υλοποιήστε μια κλάση Java με όνομα Lock η οποία αναπαριστά μια κλειδαριά με συνδυασμό 4 αριθμών. Η κλειδαριά έχει τέσσερις ανατροπείς (tumblers) που μπορούν να τεθούν σε μια ακέραια τιμή μεταξύ 0 και 9.



Κάθε κλειδαριά έχει τα παρακάτω δεδομένα:

- μια τρέχουσα ρύθμιση αριθμών (current setting) για κάθε έναν από τους 4 ανατροπείς (από 0 έως 9).
- έναν απαιτούμενο συνδυασμό αριθμών (required setting) για τους 4 ανατροπείς ώστε η κλειδαριά να ανοίγει.

Κάθε κλειδαριά έχει τις παρακάτω μεθόδους:

- έναν κατασκευαστή `Lock(int[] required)` που παίρνει σαν παράμετρο τον συνδυασμό αριθμών που απαιτείται για τους 4 ανατροπείς ώστε η κλειδαριά να ανοίγει.
- μια μέθοδο `alterTumbler(int tumbler, int newSetting)` που αλλάζει την τιμή ενός ανατροπέα. Η πρώτη παράμετρος προσδιορίζει ποιος ανατροπέας πρέπει να αλλαχτεί (0-3) ενώ η δεύτερη προσδιορίζει ποια τιμή πρέπει να θέσουμε στον ανατροπέα (0-9).
- μια μέθοδο `canOpen()` η οποία επιστρέφει true εάν η κλειδαριά μπορεί να ανοίξει. Αυτό συμβαίνει μόνο εάν η τρέχουσα ρύθμιση αριθμών για κάθε έναν από τους 4 ανατροπείς είναι ίδια με τον απαιτούμενο συνδυασμό της κλειδαριάς.
- μια μέθοδο `sameSettings(Lock otherLock)` η οποία επιστρέφει true εάν το τρέχον αντικείμενο που την εκτελεί έχει τον ίδιο απαιτούμενο συνδυασμό αριθμών με αυτόν του αντικειμένου που περνάμε σαν παράμετρο.

Σημείωση: Για κάθε μία από τις παραπάνω μεθόδους που δημιουργείτε, δώστε τις κατάλληλες προσυνθήκες (preconditions) με την μορφή σχολίων Java. Εάν επιθυμείτε, μπορείτε να προσθέσετε επιπλέον μεθόδους.

Λύση:

```
public class Lock {
      public static final int NUM_TUMBLERS = 4;
      private int[] mySettings;
      private int[] myRequired;
      public Lock(int[] required) {
            //pre: required != null, required.length = NUM_TUMBLERS,
            // 0 <= required[i] <= 9
            mySettings = new int[NUM_TUMBLERS];
            myRequired = new int[NUM_TUMBLERS];
            if (required.length>3)
             throw new IllegalArgumentException ("Lock has 4
                                                        tumblers");
            for(int i = 0; i < NUM_TUMBLERS; i++)
                if (required[i]<0 || required[i]>9)
                      throw new IllegalArgumentException("Tumbler
```

```
                                                    invalid value ") ;
                myRequired[i] = required[i];
        }
        public void alterTumbler(int tumbler, int newSetting) {
                // pre: 0 <= tumbler < NUM_TUMBLERS, 0 <= newSetting <= 9
                if (tumbler<0 || tumbler > 3 || newSetting <0 ||
                                                        newSetting >9 )
                        throw new IllegalArgumentException(" Invalid tumbler
                                                    or tumbler value");
                mySettings[tumbler] = newSetting;
        }
        public boolean canOpen(){
                // pre: none
                boolean correct = true;
                int i = 0;
                while( correct && i < NUM_TUMBLERS)
                {       correct = mySettings[i] == myRequired[i];
                        i++;
                }
                return correct;
        }
        public boolean sameSettings(Lock otherLock) {
                //pre: otherLock != null
                boolean same = true;
                int i = 0;
                while( same && i < NUM_TUMBLERS;
                {       same = mySettings[i] == otherLock.mySettings[i];
                        i++;
                }
                return same;
        }
}
```

Point breakdown:
2 - instance var(s) for current settings.
2 - instance var(s) for required settings.
the constructor and methods are worth 4 points each
1 point: for stated precondition (no need to check precondition)
1 point: attempt to carry out purpose of method / constructor
2 points: correctly carry out purpose of method / constructor
Bonus + 3 Exception handling

Άσκηση 5 (16 μονάδες) Ορθότητα Αντικειμενοστρεφούς Κώδικα

Στους παρακάτω ορισμούς κλάσεων Java υπάρχουν 4 τουλάχιστον λογικά λάθη. Η μέθοδος `main()` της κλάσης `Example` ενώ δεν πρέπει να τυπώνει τίποτα, τυπώνει την συμβολοσειρά `"(0,0) is same"`.

```
/**
 * Point in plane with (0,0) as upper left.
 * x increases to right, y increases down.
 */
1. public class Point {
2.  /** x,y point in plane */
3.  int x;
4.  int y;

  /** Equality Check */
5.  public boolean equals (Object p) {
6.    return (this.x - x == this.y - y);
7.  }

  /** Construct Point */
8.  public Point (int a, int b) {
9.    y = b;
10.   x = a;
11. }
12.}
```

```
/**
 * represents a point whose x value
 * is the same as its y value.
 */
13.public class SpecialPoint extends
Point {
/** Common value for both x and y */
14. int z;

  /** Constructor */
15. public SpecialPoint (int z) {
16.  super (z,z);
17. }

  /** Represent as String */
18. public String toString () {
19.  return "(" + z + "," + z + ")";
20. }
21.}
```

```
/**
 * Test code
 */
22. public class Example {
23.  public static void main (String [] args) {
24.    Point p = new SpecialPoint(5);
25.    Point q = new Point (3,3);
26.    if (p.equals(q)) {
27.      System.out.println (p + " is same");
28.    }
29.  }
30. }
```

Για κάθε ένα από τα λάθη που εντοπίζετε στον παραπάνω κώδικα
(α) Γράψτε την γραμμή στην οποία εμφανίζεται.
(β) Δώστε μια σύντομη περιγραφή του λάθους.
(γ) Υποδείξτε μια διόρθωση του λάθους.
Σημείωση: Μπορείτε να περιγράψετε τα λάθη με οποιαδήποτε σειρά (δηλ όχι αναγκαστικά με την σειρά εμφάνισής τους στο πρόγραμμα). Για την διόρθωσή τους μπορείτε να προτείνετε την προσθήκη καινούργιων γραμμών κώδικα ή την αφαίρεση γραμμών από τον υπάρχοντα κώδικα.

Λύση:
1. In SpecialPoint constructor (line 15), the instance variable for z is not set. Either set it, or move the toString() method into Point and replace z,z with x,y
2. In Point.equals() (line 6), the code makes multiple faults
   2a) note that the reference 'this.x' is the same as 'x', so the lonely 'x' should instead be (Point) p.x
   2b) note that the reference 'this.y' is the same as 'y', so the lonely 'y' should instead be (Point) p.y
   2c) the '==' check is invalid (this will return true if both points are on the XY diagonal. Replace with '(this.x == (Point) p.x) && (this.y == (Point) p.y)'

Άσκηση 6 (15 μονάδες) Κληρονομικότητα Κλάσεων

Δημιουργήστε τρεις κλάσεις Java που αναπαριστούν διαφορετικά είδη ζωυφίων (bugs). Τα ζωύφια ανήκουν σε δύο κατηγορίες: έντομα (insects) και αράχνες (spiders). Όλα τα έντομα έχουν 6 πόδια (legs) ενώ όλες οι αράχνες έχουν 8 πόδια. Επιπλέον, όλα τα είδη ζωυφίων έχουν διαφορετικά χρώματα τα οποία και θα πρέπει να αναπαραστήσετε με συμβολοσειρές. Για τα έντομα, θα πρέπει επίσης να καταγράψετε εάν το έντομο είναι μυρμήγκι (ant) ενώ για τις αράχνες, θα πρέπει να καταγράψετε εάν η αράχνη είναι δηλητηριώδης (poisonous). Η παρακάτω κλάση `UseBug` χρησιμοποιεί την κλάση `Bug` και τις υποκλάσεις της, τις οποίες καλείστε να υλοποιήστε ώστε η μέθοδος `main()` να εκτελείται κανονικά.

```
public class UseBug {
  public static void main(String args[]) {
   Bug critters[] = new Bug[2];
   Insect bug1 = new Insect();
   bug1.isAnt = true;
   Spider bug2 = new Spider();
   bug2.isPoison = true;
   critters[0] = bug1;
   critters[1] = bug2;
   int totalLegs = 0;
   for (int i = 0; i < critters.length; i++) {
        totalLegs += critters[i].numLegs();
        critters[i].color = "black";
  //critters[i].isAnt or critters[i].isPoison would be illegal here
   } // end for
   System.out.println(totalLegs); // will print 14
  } // end main
} // end class UseBug
```

Σημείωση: Δεν χρειάζεται να ασχοληθείτε με προβλήματα ενθυλάκωσης (encapsulation) και απόκρυψης των δεδομένων (information hiding). Μπορείτε να υποθέσετε ότι κανένας άλλος κώδικας πελάτης δεν θα θελήσει να δημιουργήσει στιγμιότυπα της κλάσης Bug εκτός από την μέθοδο main() της UseBug. Επίσης δεν είναι απαραίτητο να παρέχετε άλλες μεθόδους ή μεταβλητές στιγμιοτύπων εκτός από αυτές που απαιτούνται από την UseBug.

Λύση:

```
public abstract class Bug {
      public abstract int numLegs();
      public String color;
} // end class Bug
public class Insect extends Bug {
      public int numLegs() {
        return 6;
      }
      public boolean isAnt;
} // end class Insect
public class Spider extends Bug {
      public int numLegs() {
        return 8;
      }
      public boolean isPoison;
} // end class Spider

// alternative solution where Bug doesn't really have to be abstract
public class Bug {
      int legs; // may be public or private
      public int numLegs() {
        return legs;
      }
      String color;
} // end class Bug
public class Insect extends Bug {
      public boolean isAnt;
      public Insect() {
        legs = 6;
      }
} // end class Insect
public class Spider extends Bug {
      public boolean isPoison;
      public Spider() {
        legs = 8;
      }
} // end class Spider
We can also create a constructor for Bug
      public Bug(int numLegs) {
            legs = numLegs; }
and call it from the subclasses constructors
      public Insect() {
            super(6); }
      public Spider() {
            super(8);
      }
```

Άσκηση 7 (15 μονάδες) Θεωρία Αντικειμενοστρεφούς Κώδικα

(α) **(5 μονάδες)** Τι είναι η συμβατότητα (compatibility) τύπων (κατά την αντικατάσταση εκφράσεων ενός κώδικα Java), και πώς ο προγραμματισμός που βασίζεται στους γενικότερους τύπους ενισχύει την ικανότητα επαναχρησιμοποίησης κώδικα; Δώστε μια σύντομη εξήγηση.

Λύση:
Type compatibility refers to the ability to use an instance of a subclass anywhere an instance of a superclass is expected, such as in assignment or in parameter passing. It helps promote code reusability because functions and other code can be written to use the more generic superclass, but that same code can be reused to work with all subclasses.

(β) **(5 μονάδες)** Υπάρχει κάποιος λόγος για να δηλώσουμε σαν αφηρημένες (abstract) μεθόδους μιας κλάσης που είναι ιδιωτικές (private); Δώστε μια σύντομη εξήγηση.

Λύση:
No. "Abstract" means that late binding will be performed – this is only relevant if the method is overridden in a subclass. Since private member functions can't be accessed by the subclass, they probably aren't going to be overridden.

(γ) **(5 μονάδες)** Έχει νόημα η δήλωση μιας τελικής αφηρημένης (final abstract) μεθόδου; Δώστε μια σύντομη εξήγηση.

Λύση:
No. A final method cannot be overridden, and an abstract method has no implementation. Thus a final abstract method could never be implemented and is useless.