Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

ΗΥ-252 – Αντικειμενοστρεφής Προγραμματισμός
Βασίλης Χριστοφίδης

Επαναληπτική Εξέταση (3 ώρες)
Ημερομηνία: 24 Αυγούστου 2009

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Άσκηση 1 (10 μονάδες) Συγκρίσεις Αντικειμένων

Υλοποιήστε μια στατική μέθοδο compareVectors(Vector v1, Vector v2, Comparator comp) η οποία συγκρίνει δύο μη-ταξινομημένα διανύσματα (Vectors) v1 και v2 χρησιμοποιώντας ένα αντικείμενο comp τύπου Comparator. Η μέθοδος επιστρέφει *αληθές* όταν όλα τα αντικείμενα του v1 είναι μικρότερα ή ίσα από όλα τα αντικείμενα του v2: διαφορετικά επιστρέφει *ψευδές*.

```
/**************************************************************
* This method compares two Vectors. It does NOT assume that
* either Vector is sorted in any way.
*
* Parameters:
* v1, v2: the two Vectors
* comp: a Comparator to use when comparing objects in the Vectors
*
* Return value:
* true if all objects in v1 are less than or equal to all objects
* in v2; otherwise, returns false
**************************************************************/

public static boolean compareVectors(Vector v1, Vector v2,
                                     Comparator comp) {
```

Λύση:

```
for (int i = 0; i < v1.size(); i++) {
\\ 3 points for the basic logic of comparing all elements
\\ of v1 to all element of v2
  for (int j = 0; j < v2.size(); j++) {
  \\ 2 points for using size() and get() methods
     if (comp.compare(v1.get(i), v2.get(j)) > 0)
     \\ 5 points for using comparator (calling
     \\ comp.compare() and using the result correctly)
         return false;
  } // end for j
} // end for i
return true;
```

```
} // end compareVectors
```

Άσκηση 2 (10 μονάδες) Επαναλήπτες και Είσοδος

Οι επαναλήπτες (Iterators) στην Java χρησιμοποιούνται για να σαρώνουν τα στοιχεία μιας συλλογής αντικειμένων και βασίζονται στην ίδια ακολουθιακή προσπέλαση (sequential access) που χρησιμοποιούν οι ροές (Streams). Υλοποιήστε έναν επαναλήπτη `MyReaderIterator` ο οποίος παίρνει σαν όρισμα έναν αναγνώστη `Reader` και επιστρέφει μια προς μια τις γραμμές εισόδου (μπορείτε να παραλείψετε την υλοποίηση της μεθόδου `remove()`). Εξασφαλίστε ότι ο επαναλήπτης `MyReaderIterator` θα κλείσει κανονικά τον αναγνώστη `Reader` όταν αυτός εξαντλήσει την είσοδο.

Βοήθεια: Θυμηθείτε ότι η κλάση `BufferedReader` διαθέτει μια μέθοδο `readLine()` η οποία επιστρέφει σαν συμβολοσειρά `String` την επόμενη γραμμή από τον αναγνώστη `Reader`, η `null` εάν έχει φτάσει στο τέλος του αρχείου. Ο επαναλήπτης `MyReaderIterator` θα πρέπει να μπορεί να χρησιμοποιηθεί από το ακόλουθο πρόγραμμα Java το οποίο τυπώνει όλες τις γραμμές που περιέχουν περισσότερους από 10 χαρακτήρες.

```
public static void main(String[] args) throws IOException{
  Reader r = new FileReader("C:\\myfile.txt");
  MyReaderIterator itr = new MyReaderIterator( r );
  while ( itr.hasNext() ){
      String s = (String) itr.next();
      if ( s.length() >= 10 )
         System.out.println( s );
  }
}
```

Λύση:

```
class MyReaderIterator implements Iterator{
  private String currentLine;
  private BufferedReader r;
  MyReaderIterator(Reader r) {
     this.r = new BufferedReader(r);
     try{
         currentLine = this.r.readLine();
     }catch(IOException e){
         r = null;}
  }
  public boolean hasNext(){
     return currentLine != null;
  }
  public Object next(){
     Object ret = currentLine;
     try{
        currentLine = r.readLine();
        if (currentLine == null) r.close();
     }catch(IOException e){
       r = null; }
  return ret;
  }
}
```

Άσκηση 3 (11 μονάδες) Κληρονομικότητα Κλάσεων και Εξαιρέσεις

Δώστε τον κώδικα των κλάσεων SetException, DuplicateException, και NotFoundException έτσι ώστε το παρακάτω πρόγραμμα Test να παράγει στην έξοδο:

Output

```
Cannot add duplicate object
Cannot remove non-existing object
```

```
class Test {
 public static void main (String[ ] args) {
  try {
      throw new DuplicateException( );
      }
  catch (SetException e) { e.printMessage( ); }
  try {
      throw new NotFoundException( );
      }
  catch (SetException e) { e.printMessage( ); }
 }
}
```

Λύση:

```
class SetException extends Exception { //1 point
 public SetException(String s) {
   super(s);
 }// 2 points
 public void printMessage( ) {
  System.out.println (getMessage( ));
 }// 2 points
} // 5 points
class DuplicateException extends SetException {
 public DuplicateException( ) {
  super("Cannot add duplicate object");
 }
} // 3 points
class NotFoundException extends SetException {
 public NotFoundException( ) {
  super("Cannot remove non-existing object");
 } // 3 points
}
```

Άσκηση 4 (10 μονάδες) Κατανόηση Αντικειμενοστρεφούς Κώδικα

Βρείτε τουλάχιστον 10 λάθη (υπάρχουν περισσότερα) στον κώδικα της παρακάτω κλάσης. Τα λάθη είναι συντακτικά (compiler errors) και λογικά (bugs). Για κάθε λάθος δώστε την γραμμή εμφάνισής του και εξηγείστε σύντομα πως θα μπορούσατε να το διορθώσετε.

```
1 import java.io.*;
2
3 /* A class that counts the letter frequency in a text input
4 * stream. */
5
6 public class Count {
7
8    /* Counts the letter frequency in a character
```

```
9      * input stream. */
10    public static count(String fileName) {
11    /* initialize the frequency counts */
12      int[] alpha = new int[];
13      for(int i=0 ; i <= alpha.length ; i++ )
14          alpha[i] = 0
15
16     /* Keep track of line numbers for error reporting */
17      try {
18          LineNumberReader in2 = new LineNumberReader(
19                                 new FileReader( fileName ) ) ;
20
21     /* Read through to end of file. */
22      while((ch = in2.read()) != -1)
23              char c2 = (char) ch;
24
25     /* Update character counts. */
26      if( Character.isLetter(c2))
27         alpha[c2-'a']++;
28     } catch( Exception e ) {
29      System.err.println("Could not count file.");
30     } catch( IOException e ) {
31     /* Handle input error. */
32    System.err.println("Errorreading line"in2.getLineNumber());
33
34      System.err.println(e.getMessage());
35     } finally {
36     /* Make sure the stream is closed. */
37      try {in2.close();} catch(Exception e) { }
38    }
39
40    /* Print out results. */
41     for(int i=0 ; i < alpha.length;)
42         System.out.println("'"+(char)('a'+i)+"': "+alpha[i]) ;
43  }
44
45    /* Takes a list of files to process. */
46    public static void main(String[] args) {
47      for( int i=0 ; i < args.length ; i++ )
48          count( args[i] ) ;
49    }
50  }
```

Λύση:

Line 10: count does not have a return type.
Line 12: The initializer for alpha does not specify the length of the array.
Line 13: The for loop will exceed the array bounds when i == alpha.length.
Line 14: No semicolon after alpha[i] = 0.
Line 22: There is no declaration of the variable ch.
Line 22-27: There should be curly braces around the while loop.
Line 27: c2 may be upper or lower case. It should be converted to lower case for comparison with 'a'.
Line 30: The catch block for `Exception` will match every possible exception. The catch block for `IOException` will never execute.

Line 32: There is no concatenation operator (`+') before `in2.getLineNumber()`.
Lines 33 and 37: in2 is local to the try block, it isn't defined in the catch and finally blocks.
Line 41: There is no increment in the `for` loop. The loop will never exit.
Line 46: string should be capitalized: `String`.

Άσκηση 5 (20 μονάδες) Σχεδίαση Βασισμένη σε Συμβόλαια

Θυμηθείτε ότι ένα προγραμματιστικό συμβόλαιο είναι μια περιγραφή που συνοδεύει μια μέθοδο και δηλώνει τις ιδιότητες της μεθόδου όταν αυτή εκτελεστεί. Τυπικά περιλαμβάνει προ- και μετα-συνθήκες καθώς και αμετάβλητες συνθήκες που χαρακτηρίζουν όλα τα στιγμιότυπα μιας κλάσης. Αυτές οι συνθήκες επιτρέπουν ουσιαστικά στον προγραμματιστή να κάνει ρητές όλες τις παραδοχές που διέπουν την σχεδίαση μιας συνιστώσας λογισμού όπως μια κλάση. Για παράδειγμα, σας δίνεται η ακόλουθη κλάση `C` που περιλαμβάνει δύο αμετάβλητες συνθήκες `I` και `J`.

```
class C {
  int max;
  int a []; // Invariant I: for all 0 < j < max: a[j] < max
  int b []; // Invariant J : for all 0 < i < max*max: b[i] < max
  /* Contract P */
  int aa (int i) {
     return ((i + 89) * 42 % (max -1)); }
  /* Contract Q */
  int bb (int i) {
  return (i * a[aa(i)] *3145 % (a[i]*max-1)); }
  /* Contract R */
  int lookup (int i) {
  return (a[b[bb(i)]]);}
}
```

α) **(15 μονάδες)** Σας ζητείται να αναγνωρίσετε τα συμβόλαια `P`, `Q`, και `R` των 3 μεθόδων της κλάσης και να αποδείξετε ότι η μέθοδος `lookup()` δεν θα τερματίσει πρόωρα εγείροντας μια εξαίρεση `ArrayIndexOutOfBoundsException` δηλ. ότι ικανοποιείται το συμβόλαιο `R` από τα συμβόλαια `P` και `Q` και τις αμετάβλητες συνθήκες `I` και `J`.

Λύση:
1. Give a contract for P.
`Pre condition: true. Post condition: 0<aa(i)<max.`
2. Give a contract for Q. Argue that it is satisfied.
Pre `condition: i<max. Post condition: 0<bb(i)<max.`
Immediate, by 1.
3. Argue that the contract for R given as
`Pre condition: i<max. Post condition: true.`
entails that method lookup can never crash with an array index out of bounds exception. Argue that the contract is satisfied.

It is pretty clear that this contract entails the claim that lookup doesn't crash. If it were to crash, the post condition couldn't be true. That the code is ok follows directly from 1. and 2.

β) **(5 μονάδες)** Σας δίνεται η ακόλουθη κλάση `D` που περιλαμβάνει τα συμβόλαια των μεθόδων `forward()` και `backward()`. Τα δύο συμβόλαια δηλώνουν ουσιαστικά ότι οι δείκτες θέσης των πινάκων διατηρούνται εντός των προβλεπόμενων ορίων. Σας ζητείται να βρείτε τις αμετάβλητες συνθήκες `I` και `J` που εξασφαλίζουν την ικανοποίηση αυτών των συμβολαίων.

```
class D {
  int max;
  int a []; // Invariant I
  int b []; // Invariant J
  /* Contract P: Pre condition: true. Post condition: true. */
  int forward (int i) {return (a[b[i]]);}
  /* Contract Q: Pre condition: true. Post condition: true. */
  int backward (int i) { return (b[a[i]]);}
}
```

Λύση:
For all i such that `0<=i<max` the following holds: `a[i]<max` and `b[i]<max`.

Άσκηση 6 (15 μονάδες) Υλοποιήσεις Πλαισίου Συλλογών Αντικειμένων

Μας ενδιαφέρει η υλοποίηση ενός δισδιάστατου αραιού (sparse) πίνακα πραγματικών διπλής ακρίβειας (doubles) χρησιμοποιώντας απεικονίσεις (maps) αντί για συνδεδεμένες λίστες (linked lists), προσφέροντας κατάλληλες μεθόδους ανάκλησης (fetch) και αποθήκευσης (store).
α) **(3 μονάδες)** Τι θα χρησιμοποιήσετε σαν κλειδί (key) της παραπάνω απεικόνισης; Εξηγήστε την λογική της επιλογής σας.

Λύση:
Best: A String such as: `String key= row + "," + column;`
//best because `hashCode()` is already defined for `Strings`
OK: An object containing row and column (must define `equals()`, `hashCode()`)
Wrong: An arithmetic combination of row and column (cannot guarantee uniqueness)
Very wrong: anything that depends on the value of the `double`

β) **(2 μονάδες)** Τι θα χρησιμοποιήσετε σαν τιμή (value) της παραπάνω απεικόνισης;

Λύση:
The `double` wrapped in a `Double`.

γ) **(2 μονάδες)** Αναφέρετε τουλάχιστον ένα μειονέκτημα αυτής της υλοποίησης.

Λύση:
No easy way to step through values in a row or column

δ) **(8 μονάδες)** Υλοποιήστε τις μεθόδους `store()` και `fetch()` της κλάσης `SparseArray`.

```
import java.util.HashMap;
public class SparseArray {
  HashMap map = new HashMap(500);
  public void store(int row, int column, double value) {
```

Λύση:
```
map.put(row + "," + column, new Double(value));
```

```
  }

  public double fetch(int row, int column) {
```

Λύση:
```
Object obj = map.get(row + "," + column);
if (obj == null) return 0.0;
else return ((Double) obj).doubleValue();
```

```
  }
}
```

Άσκηση 7 (30 μονάδες) Σχεδίαση και Υλοποίηση Κλάσεων Αντικειμένων

Οι συμβολοσειρές είναι αντικείμενα της κλάσης `java.lang.String` των οποίων το περιεχόμενο δεν μπορεί να αλλάζει (immutable). Αν και αυτή είναι μια εν γένει χρήσιμη λειτουργικότητα, ορισμένες όμως φορές, όπως για παράδειγμα στους editors, είναι βολικό να έχουμε επίσης τη δυνατότητα αλλαγής του περιεχομένου μιας συμβολοσειράς. Σας δίνεται η παρακάτω διεπαφή `MutableString` η οποία ορίζει τέσσερεις μεθόδους για μεταβαλλόμενες συμβολοσειρές. Επιπλέον σας δίνεται η κλάση `Test` που περιέχει μια κύρια μέθοδο για τον έλεγχο μιας συγκεκριμένης υλοποίησης `MutableStringArray` αυτής της διεπαφής η οποία αναπαριστά μια συμβολοσειρά σαν έναν πίνακα από χαρακτήρες.

```
interface MutableString {
/* index of first character is 0.
* The last character has index size()-1
* return the number of characters in this mutable string
*/
int size();

// return the char at the given index
char charAt(int index);

// replace the substring given by from and to with replacement.
void replace(int from, int to, MutableString replacement);
}
// return a mutable string that is a copy of the charaters at
```

```
// index from to index to. Both index from and to are included.
MutableString subString(int from, int to);


class MutableStringArray implements MutableString {
  private char[] characters;
  MutableStringArray() { characters = new char[0]; }
  MutableStringArray(String initial){
    characters = new char[initial.length()];
    for (int i = 0; i<initial.length();i++)
         characters[i] = initial.charAt(i);
    }
  //other methods inherited from Object – see questions below
}

class Test {
  public static void main(String[] a){
    MutableString ms = new MutableStringArray("Kasper");
    System.out.println(ms);
    System.out.println(ms.subString(1,4));
    ms.replace(2,3,new MutableStringArray("zzbb"));
    System.out.println(ms);
    ms.replace(2,5,new MutableStringArray(""));
    System.out.println(ms);
    ms.replace(2,1,new MutableStringArray("sp"));
    System.out.println(ms);
  }
}
```

Η εκτέλεση της μεθόδου της κλάσης `Test` δίνει το παρακάτω αποτέλεσμα:

```
Kasper
aspe
Kazzbber
Kaer
Kasper
```

α) **(2 μονάδες)** Υλοποιήστε τις μεθόδους `size()` και `charAt(int)` της κλάσης `MutableStringArray`

Λύση:

```
public int size(){return characters.length;}
public char charAt(int index){ return characters[index]; }
```

β) **(5 μονάδες)** Δώστε τις προ συνθήκες για τα ορίσματα `from` και `to` της μεθόδου `replace()`; Σας συνιστάται να κατανοήσετε γιατί είναι σωστή η τελευταία κλήση της κύριας μεθόδου της `Test`, ακόμα και εάν το `to` είναι μικρότερο από το `from`.

Λύση:

The pre-condition for from and to in the replace method is:
The substring starting at index from upto and including index to is is replaced by replacement. `0<=from<=length` and `-1<=to<length` and `to>=from-1`.

What is worth to notice is that the call `replace(2,1,"sp")` replaces the empty substring starting at 2 with "sp".

γ) **(5 μονάδες)** Η μέθοδος `replace()` δέχεται επίσης σαν όρισμα μια συμβολοσειρά `MutableString`. Είναι ασφαλές να υποθέσουμε στην υλοποίηση της ότι το όρισμα αυτό θα είναι όντως στιγμιότυπο της `MutableStringArray`; Εξηγήστε σύντομα τον συλλογισμό σας.
Η τιμή επιστροφής της μεθόδου `subString()` δηλώνεται σαν μια συμβολοσειρά `MutableString`. Είναι ασφαλές να επιστρέψουμε στην υλοποίηση της ένα στιγμιότυπο της `MutableStringArray`; Εξηγήστε σύντομα τον συλλογισμό σας.

Λύση:
No, it is not safe to assume that the argument is really a `MutableStringArray` as we could have made a different implementation of the `MutableString` and passed that as argument. On the other hand, it is safe to return a `MutableStringArray` when a `MutableString` is assumed.

δ) **(10 μονάδες)** Υλοποιήστε την μέθοδο `substring(int from, int to)` της κλάσης `MutableStringArray`.

Λύση:
```
public MutableString subString(int from, int to){
  if (from <0 ) from = 0;
  if (to >= size() ) to = size()-1;
  MutableStringArray sub = new MutableStringArray();
  sub.characters = new char[to-from+1];
  int subindex=0;
  for (int i = from; i<=to; i++){
    sub.characters[subindex] = characters[i];
    subindex++; }
  return sub;
}
```

ε) **(8 μονάδες)** Υλοποιήστε την μέθοδο `equals(Object other)` της κλάσης `MutableStringArray`.

Λύση:
```
public boolean equals(Object other){
  if (other == null) return false;
  if (! (other instanceof MutableString)) return false;
  MutableString obj = (MutableString)other;
  if (obj == this) return true;
  if (obj.size() != this.size()) return false;
  for (int i = 0; i<size();i++)
  if (obj.charAt(i) != this.charAt(i)) return false;
  return true;
}
```