Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

ΗΥ-252 – Αντικειμενοστρεφής Προγραμματισμός
Βασίλης Χριστοφίδης

Τελική Εξέταση (3 ώρες)
Ημερομηνία: 25 Φεβρουαρίου 2007

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Άσκηση 1 (20 μονάδες) Βασική Λειτουργικότητα Αντικειμένων

Σας δίνεται η παρακάτω κλάση Java `OPoint` για την αναπαράσταση σημείων σ'ένα τρισδιάστατο χώρο:

```
class OPoint{
  private double x, y, z;
  public OPoint(double xp, double yp, double zp){
    x=xp;
    y=yp;
    z=zp;
  }
  public double x(){return x;}
  public double y(){return y;}
  public double z(){return z;}
  public String toString(){
    return "("+x+","+y+","+z+")";
  }
  public boolean equals(Object o){
    if(o instanceof OPoint){
       OPoint that = (OPoint)o;
       return that.x==this.x && that.y == this.y &&
                      that.z == this.z;
    }
    else return false;
  }
  public static void main(String[] a){
    OPoint p1 = new OPoint(1,2,3);
    OPoint p2 = new OPoint(1,2,3);
    OPoint p3 = new OPoint(2,2,2);
    System.out.println(p1);
    System.out.println(p2);
    System.out.println(p3);
    System.out.println();
    System.out.println(p1==p1);
    System.out.println(p1==p2);
    System.out.println(p1==p3);
    System.out.println();
    System.out.println(p1.equals(p1));
    System.out.println(p1.equals(p2));
    System.out.println(p1.equals(p3));
    }
}
```

Κατά την εκτέλεση αυτού του προγράμματος παίρνουμε το ακόλουθο αποτέλεσμα:

```
(1.0,2.0,3.0)
(1.0,2.0,3.0)
(2.0,2.0,2.0)
true
false
false
true
true
false
```

α) **(2 μονάδες)** Εξηγήστε γιατί η λογική έκφραση `p1==p2` είναι ψευδής ενώ `p1.equals(p2)` είναι αληθής.

> **Λύση:**
> == compares references, and `p1` and `p2` are pointing at two different instances of the class `OPoint`. On the other hand, the method `p.equals(q)` inspects the content of `p` and of `q` to determine equality of content of the 2 `OPoints` involved. In this case `p1` and `p2` have the same content (state).

β) **(10 μονάδες)** Τροποποιήστε τον ορισμό της κλάσης `OPoint` (μετονομάζοντάς την σε `Point`) έτσι ώστε μόνο ένα στιγμιότυπο κάθε σημείου (point) να δημιουργείται από έναν οποιοδήποτε κώδικα-πελάτη. Θα πρέπει να περιλάβετε μια δομή δεδομένων (`HashMap`) για να αποθηκεύσετε τα στιγμιότυπα της κλάσης `Point` που έχουν ήδη δημιουργηθεί. Πιο συγκεκριμένα:

- Καταστήστε αρχικά τον κατασκευαστή της κλάσης ιδιωτικό (private), έτσι ώστε ένας κώδικας -πελάτης να μην μπορεί να τον χρησιμοποιήσει!
- Στην συνέχεια, υλοποιήστε μια μέθοδο `public static Point point(double xp, double yp, double zp)` για να δημιουργούμε αντικείμενα σημεία (points) από ένα κώδικα -πελάτη.
  Σε αυτή τη μέθοδο, κάντε πρώτα τον έλεγχο εάν το στιγμιότυπο με τις συντεταγμένες xp, yp, zp (που δίνονται σαν παράμετρος) υπάρχει ήδη στην αποθηκευτική δομή δεδομένων `createdPoints`. Σε αυτή την περίπτωση, επιστρέψτε το στιγμιότυπο που είναι ήδη αποθηκευμένο! Διαφορετικά, χρησιμοποιήστε τον κατασκευαστή της κλάσης για να δημιουργήσετε ένα καινούργιο σημείο, αποθηκεύστε το στην δομή δεδομένων `createdPoints` και επιστρέψτε το ήδη δημιουργημένο στιγμιότυπο στον κώδικα-πελάτη!

Σημείωση: Η κλάση `HashMap` παρέχει τις ακόλουθες μεθόδους:

```
public Object put(Object key, Object value)
public boolean containsKey(Object key)
public Object get(Object key)
```

> **Λύση:**
> Here is a proposal where I convert a point to a string and use its `hashCode` instead of defining a `hashCode` method for points:
> `class Point{`

```
  private double x, y, z;
  private static java.util.Map createdPoints =
                                        new java.util.HashMap();
  private Point(double xp, double yp, double zp){
    x=xp;
    y=yp;
    z=zp;
  }
  public static point point(double xp,double yp,double zp){
    Point p = new Point(xp,yp,zp);
    if(!createdPoints.containsKey(p.toString()))
       createdPoints.put(p.toString(),p);
    return (Point)createdPoints.get(p.toString());
  }
}

Assessors and equals() are the same
```

γ) **(8 μονάδες)** Τροποποιήστε την μέθοδο `main` σύμφωνα με τις αλλαγές που κάνατε στο προηγούμενο ερώτημα και γράψτε τι θα τυπωθεί κατά την εκτέλεσή της.

Λύση:

```
  public static void main(String[] a){
    Point p1 = point(1,2,3);
    Point p2 = point(1,2,3);
    Point p3 = point(2,2,2);
    System.out.println(p1);
    System.out.println(p2);
    System.out.println(p3);
    System.out.println();
    System.out.println(p1==p1);
    System.out.println(p1==p2);
    System.out.println(p1==p3);
    System.out.println();
    System.out.println(p1.equals(p1));
    System.out.println(p1.equals(p2));
    System.out.println(p1.equals(p3));
  } // 3 points
```

The output is

```
1.0,2.0,3.0
1.0,2.0,3.0
2.0,2.0,2.0
true
true
false
true
true
false // 3 points
```

Άσκηση 2 (10 μονάδες) Χειρισμός Εξαιρέσεων

Τι θα τυπωθεί κατά την εκτέλεση του παρακάτω προγράμματος Java:

```
public class ExceptionQuiz {
 // These exception classes would each have the usual 2
 // constructors inside -- one with zero parameters and one
```

```
// with a String parameter. Omitted here to save space.
public static class RedException extends RuntimeException {}
public static class BlueException extends RedException {}
public static class GreenException extends RedException {}
public static class PurpleException extends RedException {}
public static void main(String args[]) {
 String words[] = {"abcdef","xyzabc","abcdxyz","xyz","zyxabc"};
 for (int i = 0; i < words.length; i++) {
     try {
          System.out.println(filter1(words[i]));
         }
     catch (RedException e) {
          System.out.println("red");
         }
    } // end for
} // end main
public static String filter1(String s) {
 String t = filter2(s);
 String result = "";
 try {
     for (int i = 0; i < t.length(); i++) {
          char c = t.charAt(i);
          if (c == 'x') {
              System.out.println("green");
              throw new GreenException();
          }
          else if (c == 'z') {
               System.out.println("purple");
               throw new PurpleException();
             }
             else
               result += c;
      } // end for
 }
 catch (GreenException e) {
    result = "X";
 }
 return result;
} // end filter1
public static String filter2(String s) {
  if (s.length() < 4) {
     System.out.println("blue");
     throw new BlueException();
  }
  else
     return s.substring(0,4);
} // end filter2
} // end class
```

Λύση:

```
abcd
green
X
abcd
blue
red
purple
red
```

Άσκηση 3 (20 μονάδες) Χειρισμός Πλαισίου Συλλογών Αντικειμένων

Για κάθε ένα από τα παρακάτω προγράμματα Java, σας δίνονται τέσσερις επιλογές για το τι θα μπορούσε να τυπωθεί κατά την εκτέλεσή τους. Υπογραμμίστε την σωστή απάντηση και δώστε μια σύντομη εξήγηση.

1. (5 μονάδες)

```
SortedSet mystery = new TreeSet();
mystery.add("sky");
mystery.add("sun");
mystery.add("grass");
SortedSet space = new TreeSet();
space.add("sun");
space.add("stars");
space.add("moon");
space.add("stars");
space.removeAll(mystery);
System.out.println(space);

a. [moon, stars]
b. [stars, moon]
c. [sun]
d. [moon, stars, stars]
```

Λύση:
a.
The mystery set gets `[grass, sky, sun]`. The space set gets [moon, stars, sun]. There is only one "stars" entry in space; sets can't have duplicate elements so the fourth add to space has no effect.
The call `space.removeAll(mystery)` removes every element in mystery from space. The only element in mystery which is also in space is "sun", so this element is deleted. Since space is a sorted list, the two remaining elements are printed in alphabetic order. If you chose b rather than a, you got 2 points, since all you missed was the order.

2. (5 μονάδες)

```
SortedMap mystery = new TreeMap();
mystery.put("sky", "blue");
mystery.put("grass", "green");
mystery.put("sun", "yellow");
mystery.put("grass", "brown");
mystery.put("sky", "grey");
System.out.println(mystery.get("sky") + ", "
     + mystery.get("trees") + ", " + mystery.get("grass"));

a. yellow, , brown
b. yellow, null, green
c. grey, null, brown
d. no output – code will throw an exception
```

Λύση:

c.
`mystery gets grass->green, sky->blue, sun->yellow`
Then we set values for grass and sky again. These overwrite the old values, so mystery becomes
`grass->brown, sky->grey, sun->yellow`
This means `mystery.get("sky")` returns `"grey"` and `mystery.get("grass")` returns `"green"`. There is no entry for `"trees"`, so `mystery.get("trees")` returns null (not an empty string, and it doesn't throw an exception).

3. (5 μονάδες)

```
SortedMap mystery = new TreeMap();
mystery.put("sky", "blue");
mystery.put("grass", "green");
mystery.put("sun", "yellow");
Set entries = mystery.entrySet();
Iterator iter = entries.iterator();
Map.Entry pair = (Map.Entry) iter.next();
iter.remove();
System.out.println(mystery);

a. {grass=green, sky=blue, sun=yellow}
b. {sky=blue, sun=yellow}
c. {sky=blue, grass=green, sun=yellow}
d. {grass=green, sun=yellow}
```

Λύση:
b.
mystery gets these entries:
`grass -> green, sky -> blue, sun -> yellow`
The set entries gets a set of the above as (key, value) pairs. This set is connected with the map, so that removing an entry from the set removes the entry from the map as well. The entry set is just providing a different view of the map.
It's important to note that the map is a sorted map. That means when you iterate over it you get the entries in order by keys. So the result of the call to next will be the `grass->green` entry. This entry is deleted from the set and therefore the map, leaving just the other two. If you forgot about ordering and left the map entries in the order they were added, you would have deleted `sky->blue` instead and ended up with choice d. I gave 2 points for that.

4. (5 μονάδες)

```
SortedMap mystery = new TreeMap();
mystery.put("sky", "blue");
mystery.put("grass", "green");
mystery.put("sun", "yellow");
Set keys = new TreeSet(mystery.keySet());
Iterator iter = keys.iterator();
```

```
Object key = iter.next();
iter.remove();
System.out.println(mystery);

a. {grass=green, sky=blue, sun=yellow}
b. {sky=blue, grass=green, sun=yellow}
c. {sky=blue, sun=yellow}
d. {grass=green, sun=yellow}
```

Λύση:
a.
The mystery map, as before, gets the values
`grass -> green, sky -> blue, sun -> yellow`
Then we create a key set and immediately copy it to the set called keys. Changing the copy won't have any effect on the map, so the rest of the code has no effect.
If you missed this fine point and thought the rest of the code was actually changing the key set, you would have chosen c. I gave you 1 point for this. If you chose b, you had the right answer in the wrong order and I gave you 2 points. If you chose d, you made both mistakes and I gave you no marks.

Άσκηση 4 (20 μονάδες) Αφαιρετικοί Τύποι Δεδομένων

Η τυποποιημένη διεπαφή `SortedSet` της Java αναπαριστά ένα σύνολο του οποίου τα στοιχεία είναι διατεταγμένα (δηλ ταξινομημένα ως προς μια διάταξη). Σας δίνεται ακολούθως μια παρόμοια διεπαφή για την αναπαράσταση μόνο συνόλων ακέραιων αριθμών ταξινομημένων σε αύξουσα διάταξη:

```
/** A set of unique elements kept in ascending sorted order. */
interface IntSortedSet {
/** Add x to the set. */
void add(int x);
/** Tests whether x is in the set.
* @return whether x is in the set. */
boolean contains(int x);
/** Remove x. */
void remove(Object x);
/** Return the lowest (minimal) element in the set.
* @return the lowest element. */
int first();
}
```

α) **(5 μονάδες)** Η προδιαγραφή της μεθόδου τροποποίησης `remove(Object x)` έχει τουλάχιστον ένα σοβαρό σχεδιαστικό λάθος. Προσδιορίστε με σαφήνεια το πρόβλημα και διατυπώστε μια καλύτερη προδιαγραφή. Μπορείτε να αλλάξετε την υπογραφή της μεθόδου εάν φυσικά δώσετε την απαραίτητη δικαιολόγηση.
Σημείωση: Δεν θεωρούμε την αποτυχία παραγωγής καλής τεκμηρίωσης javadoc σαν ένα "σοβαρό" πρόβλημα.

Λύση:
The problem with the spec is that it doesn't say what happens when the element is not in the set.

```
/** Remove an element. If o is an element of the set,
* removes it. Otherwise, the set is unchanged.
*/
```

This is not too terrible, as the interface already provides a contains(int) method, so the user already can have the information whether an item will be actually removed or it was not there at all.

A more serious problem is that the parameter is an Object, not int. This allows the user to compile programs which actually pass a String parameter, or anything. Also, for the common usage (of passing correct integers), the user has to create Integer objects (assuming this is the intended usage). What happens with Double objects? Are they rounded up? Down? Is it an error? Can we also pass a Number instance, and if its intValue() method returns an existing int, then it is removed? Letting the type effectively undefined (i.e. Object) allows many types of wrong (or under-specified) invocations which could have been caught at compile-time.

β) **(5 μονάδες)** Η προδιαγραφή της μεθόδου πρόσβασης `first()` έχει τουλάχιστον ένα σοβαρό πρόβλημα. Προσδιορίστε με σαφήνεια το πρόβλημα και διατυπώστε μια καλύτερη προδιαγραφή. Μπορείτε να αλλάξετε την υπογραφή της μεθόδου εάν φυσικά δώσετε την απαραίτητη δικαιολόγηση.

Λύση:
The problem is related to the handling of empty sets in the method `first()`. One possibility is to add a "*Checks*" clause:

```
/** Return the first element in the set.
 * Checks: the set is nonempty.
 * @return the first (lowest) element. */
```

You could also add a "*Requires*" clause instead, or change the signature to throw an exception in the case where the set is empty.

Υποθέστε στην συνέχεια ότι θέλουμε να υλοποιήσουμε την διεπαφή `IntSortedSet` με μια συνδεμένη λίστα της οποίας τα στοιχεία είναι ταξινομημένα σε αύξουσα διάταξη:

```
class SortedList implements IntSortedSet {
// Invariant: The list is kept in ascending sorted order and
// contains no duplicate elements.
  int element;
  SortedList next;
...
}
```

γ) **(2 μονάδες)** Υλοποιήστε την μέθοδο `first()` της διεπαφής `SortedList`.

Λύση:

```
int first() {
  return element;
}
```

Note that technically this is wrong. If the set is empty, then this implementation will return an **undefined** result, when it should throw an exception instead (typically java.util.NoSuchElementException)

(δ) **(8 μονάδες)** Μας ενδιαφέρει τώρα η δημιουργία μιας κλάσης `UnsortedList` που επίσης υλοποιεί την διεπαφή `IntSortedSet`, όπως και η `SortedList`, χωρίς όμως να ικανοποιεί την αμετάβλητη συνθήκη του προηγούμενου ερωτήματος. Θα μπορούσαμε να υλοποιήσουμε την μέθοδο `first()` γι'αυτήν την κλάση; Γράψτε τον κώδικα της μεθόδου ή εξηγήστε γιατί αυτή δεν μπορεί να υλοποιηθεί.

Λύση:

It can be done. We just can't rely on the first element being at the beginning of the list:

```
int first() {
  int min = element;
  IntUnsortedList curr = this;
  while (curr != null) {
     if (curr.element < min) min = curr.element;
     curr = curr.next;
  }
return min;
}
```

A lot of people got confused because they thought that they were supposed to return both the first element of the list, and the lowest element. The thing to remember is that we have to look at things from the standpoint of the client of the abstraction. The client shouldn't have to know or care that the set happens to be implemented as a list; the client is supposed to be able to think about this object as a sorted set.

Therefore, the "first" element is the lowest. Whether the list is sorted or unsorted is a detail of the implementation that may affect efficiency but does not affect the answer that is supposed to be returned.

Άσκηση 5 (10 μονάδες) Βασική Λειτουργικότητα Συλλογών Αντικειμένων

Σε αυτή την άσκηση καλείστε να γράψετε μια μέθοδο `intersection(List L1, List L2)` η οποία υλοποιεί την τομή δύο ταξινομημένων λιστών L1 και L2. Ως συνήθως, η τομή περιλαμβάνει μόνο τα κοινά στοιχεία των δύο λιστών (χωρίς διπλότυπα). Ο χρόνος εκτέλεσης του αλγορίθμου θα πρέπει να είναι O(M + N), όπου M το πλήθος των στοιχείων της L1 και N το πλήθος των στοιχείων της L2. Το αποτέλεσμα θα πρέπει να αποθηκεύεται σε μια καινούργια λίστα L3 (applicative transformer) για την οποία επίσης μας ενδιαφέρει η ταξινόμηση των στοιχείων της. Υποθέστε ότι τα στοιχεία των δύο λιστών είναι του ίδιου τύπου και ότι η μέθοδος `compareTo()` έχει ήδη υλοποιηθεί.

Σημείωση: Χρησιμοποιήστε τις παρακάτω μεθόδους που υποστηρίζει η διεπαφή `List`:
`Object get(int); void add(int, Object); boolean isEmpty();`

Λύση:

```
public static List intersection(List L1, List L2) {
  List L3 = new List();
  Iterator I1 = L1.iterator();
  Iterator I2 = L2.iterator();
  Object tmp1, tmp2;
  int i=0;
  if(I1.hasNext() && I2.hasNext()) {
     tmp1 = I1.next();
     tmp2 = I2.next();
     while(true){
           if(tmp1.compareTo(tmp2) == 0) {
              if(L3.isEmpty() ||
                 L3.get(i).compareTo(tmp1) != 0)
                 L3.add(i++, tmp1);
              if(I1.hasNext() && I2.hasNext()) {
                 tmp1 = I1.next();
                 tmp2 = I2.next();
              }
              else
                 break;
           }
           else if(tmp1.compareTo(tmp2) < 0)
                   if(I1.hasNext())
                      tmp1 = I1.next();
                   else
                      break;
                 else
                   if(I2.hasNext())
                      tmp2 = I2.next();
                   else
                      break;
     }
  }
  return L3;
}
```

Άσκηση 6 (15 μονάδες) Υλοποίηση Αναδρομικών Μεθόδων

Σε αυτή την άσκηση καλείστε να γράψετε δύο μεθόδους που υλοποιούν έναν απλό αλγόριθμο ταξινόμησης πινάκων (selection sort). Πιο συγκεκριμένα, ο αλγόριθμος εξετάζει, σε μια πρώτη φάση, την μικρότερη τιμή από ένα μη ταξινομημένο κομμάτι του πίνακα (το οποίο προσδιορίζεται από μια θέση `start` μέχρι το τέλος του πίνακα). Στην συνέχεια, ο αλγόριθμος ανταλλάσσει την μικρότερη τιμή (`smallest`) με αυτή που βρίσκεται ακριβώς μια θέση μετά το ταξινομημένο κομμάτι του πίνακα. Ο αλγόριθμος τερματίζει όταν όλος ο πίνακας ταξινομηθεί σε αύξουσα διάταξη. Η μέθοδος `sort(int[] a)` που ταξινομεί έναν πίνακα βασίζεται στις μέθοδους `rSort(int[] a, int start)` και `findMinLocation(int[] a, int start)`, τις οποίες και θα πρέπει να υλοποιήσετε αναδρομικά (δηλ. μην χρησιμοποιήσετε “for”, “do/while” και “while”

βρόγχους!), καθώς και στην μέθοδο swap(int[] a, int pos1, int pos2) που σας δίνεται στην συνέχεια. Υποθέστε ότι όλες οι μέθοδοι ορίζονται στην ίδια κλάση Java.

```
// Calls to this method will sort the supplied array of int's
// from smallest to largest using the selection sort algorithm.
public static void sort(int[] a) {
  rSort(a, 0);
} // end sort

public static int findMinLocation(int[] a, int start) {
// ADD CODE HERE
```

Λύση:

```
if (start < a.length - 1) {
    int smallest = findMinLocation(a, start + 1);
    if (a[start] < a[smallest])
        return start;
    else
        return smallest;
} // end if
return a.length - 1;
```

```
} // end findMinLocation
public static void rSort(int[] a, int start) {
// ADD CODE HERE
```

Λύση:

```
if (start < a.length - 1) {
   int smallest = findMinLocation(a, start);
   swap(a, smallest, start);
   rSort(a, start + 1);
} // end if
```

```
} // end rSort

public static void swap(int[] a, int pos1, int pos2) {
  int temp = a[pos1];
  a[pos1] = a[pos2];
  a[pos2] = temp;
} // end swap
```

Άσκηση 7 (15 μονάδες) Θεωρία Αντικειμενοστρεφούς Κώδικα

α) Εξηγήστε σύντομα τη διαφορά μεταξύ ενός τύπου αναφοράς σε αντικείμενα (reference type) και ενός βασικού τύπου (primitive type) στην Java.

Λύση:

Reference types are class names, interface names, and array types. A variable of reference type does not contain the data itself, but rather a pointer to a heap-allocated object. Primitive types are types such as int, boolean, and char. Variables of primitive type contain the data itself.

β) Εξηγήστε σύντομα τη διαφορά μεταξύ ενός στατικού τύπου (static type) και ενός δυναμικού τύπου (dynamic type) στην Java.

Λύση:
Static types are types of expressions in the program. They are determined at compile time from declarations. Dynamic types are the type of objects created at runtime, usually by a new instruction.

γ) Εξηγήστε σύντομα τη διαφορά μεταξύ μιας στοίβας (stack) και ενός σωρού (heap) αναφορικά με την διαχείριση μνήμης ενός προγράμματος Java.

Λύση:
Objects are stored on the heap. The stack is used when methods are called; ”stack frames” (a.k.a. ”activation records”) are stored there, which contain method arguments, local variables, and return values.