Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

ΗΥ-252 – Αντικειμενοστρεφής Προγραμματισμός
Βασίλης Χριστοφίδης

Τελική Εξέταση (3 ώρες)
Ημερομηνία: 18 Ιανουαρίου 2006

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Άσκηση 1 (11 μονάδες)

Σας δίνεται το παρακάτω πρόγραμμα Java το οποίο δοκιμάζει την δυναμική διανομή των μεθόδων (dynamic method dispatch) στην περίπτωση που εξειδικεύονται επίσης τα ορίσματα των υποσκελισμένων μεθόδων (covariant method specialization).

```
class A{
 public void test(X x){
 System.out.println("A.X run");
 }//test
 public void test2(X a, X b){
 System.out.println("A.XX run");
 }//test2
 }//A
class B extends A{
 public void test(Y x){
 System.out.println("B.Y run");
 }//test
 public void test2(X a, Y b){
 System.out.println("B.XY run");
 }//test2
 public void test2(Y a, X b){
 System.out.println("B.YX run");
 }//test2
 public void test2(Y a, Y b){
 System.out.println("B.YY run");
 }//test2
 }//B
class X{
 }//X
class Y extends X{
 }//Y
public class TestCovariant {
 public static void main(String args[]) {
 //FIRST WE SET UP ALL THE OBJECTS, NOTICE THE LAST IS SPECIAL
 A tempA=new A();
 B tempB=new B();
 X tempX=new X();
 Y tempY=new Y();
 X tempZ=new Y(); //Dynamic dispatch test variable

 System.out.println("Test dispatch on B with one variable.");
```

```
 System.out.println("Last two should be the same if dynamic
dispatch works.");
 tempB.test(tempX);
 tempB.test(tempY);
 tempB.test(tempZ);

 System.out.println("Test dispatch on B with two variables.");
 System.out.println("The second should be identical to the first
if dynamic dispatch works.");
 tempB.test2(tempX, tempX); {Αυτό το test δεν έχει νόημα}
 tempB.test2(tempX, tempX);
 System.out.println("---");
 tempB.test2(tempX, tempY);
 tempB.test2(tempX, tempZ);
 System.out.println("---");
 tempB.test2(tempY, tempX);
 tempB.test2(tempZ, tempX);
 System.out.println("---");
 tempB.test2(tempY, tempY);
 tempB.test2(tempZ, tempZ);
 }//main
}//CovariantTest
```

Τι θα τυπωθεί στην στάνταρ έξοδο κατά την εκτέλεση της μεθόδου **main**() της κλάσης **TestCovariant**;
Σημείωση: Παρατηρείστε ότι η μεταβλητή tempZ έχει στατικό τύπο **X** αλλά ο δυναμικός της τύπος είναι **Y**.

Λύση:

```
 Test dispatch on B with one variable.
 Last two should be the same if dynamic dispatch works.
 A.X run
 B.Y run
 A.X run
 Test dispatch on B with two variables.
 The second should be identical to the first if dynamic
dispatch works.
 A.XX run
 A.XX run
 ---
 B.XY run
 A.XX run
 ---
 B.YX run
 A.XX run
 ---
 B.YY run
 A.XX run
```

Άσκηση 2 (11 μονάδες)

Σας δίνεται το παρακάτω πρόγραμμα Java το οποίο κατά την εκτέλεσή του μπορεί να εγείρει εξαιρέσεις πολλαπλών τύπων. Τι θα τυπωθεί στην στάνταρ έξοδο κατά την εκτέλεση της μεθόδου **main**() της κλάσης **Excepts**;

Σημείωση: Παρατηρείστε τις σχέσεις κληρονομικότητας μεταξύ των κλάσεων εξαιρέσεων.

```
import java.io.*;
public class Excepts {
  private static String mystery(int count) throws Exception {
    if (count == 1)
          throw new FileNotFoundException();
    else if (count == 2)
          throw new IOException();
         else if (count == 3)
                 throw new NumberFormatException();
               else if (count == 4)
                  return "ABC";
                     else
                       throw new RuntimeException();
  } // end mystery

  public static void main(String args[]) {
    for (int i = 1; i <= 5; i++) {
         try {
              System.out.println(mystery(i));
              System.out.println("YES");
             }
         catch (FileNotFoundException e) {
              System.out.println("FNF");
             }
         catch (IOException e) {
              System.out.println("IO");
             }
         catch (RuntimeException e) {
              System.out.println("RT");
             }
         catch (Exception e) {
              System.out.println("EX");
             }
         System.out.println("NO");
    } // end for
  } // end main
}// end class Excepts
```

Λύση:

```
FNF
NO
IO
NO
RT
NO
ABC
YES
NO
RT
NO
```

In general, the output for each `i` in the `for` loop was worth 2 points. If you messed up a lot of YES/NOs, you lost up to 4 points total.

Άσκηση 3 (28 μονάδες)

Το τυποποιημένο πλαίσιο συλλογών της Java (JCF) περιλαμβάνει διάφορες αφηρημένες κλάσεις συλλογών (abstract collection classes), οι οποίες συγκεντρώνουν τον κοινό κώδικα για διάφορες συγκεκριμένες κλάσεις συλλογών (concrete collection classes), και παρέχουν μια βάση για νέες υλοποιήσεις αυτών των συλλογών. Για παράδειγμα το πλαίσιο περιλαμβάνει μια αφηρημένη κλάση `AbstractCollection` που υλοποιεί (μερικώς) τη διεπαφή `Collection`. Ακολουθώντας αυτή την προσέγγιση, σας δίνεται παρακάτω η διεπαφή `MyCollection`, η οποία υποστηρίζει λιγότερες μεθόδους από ότι η τυποποιημένη διεπαφή `Collection` διατηρώντας ωστόσο το αρχικό συμβόλαιό τους (δηλ. υπογραφές, προ, μετά και αμετάβλητες συνθήκες).

```
import java.util.* ;
public interface MyCollection {
  boolean add(Object o);
  void clear();
  boolean contains(Object o);
  boolean isEmpty();
  Iterator iterator();
  boolean remove(Object o);
  int size();
}
```

Γράψτε μια αφαιρετική κλάση `AbstractMyCollection` η οποία υλοποιεί την διεπαφή `MyCollection`, και η οποία περιλαμβάνει μόνο τις ακόλουθες τρεις αφαιρετικές (abstract) μεθόδους: `add()`, `iterator()`, and `size()`. Όλες οι υπόλοιπες μέθοδοι πρέπει να είναι συγκεκριμένες (concrete) δηλ. θα πρέπει να έχουν υλοποιηθεί. Η κλάση `AbstractMyCollection` δεν θα πρέπει να έχει γνωρίσματα/πεδία δηλ. μεταβλητές στιγμιοτύπων (instance variables). Θυμηθείτε ότι οι συλλογές αντικειμένων μπορούν να περιέχουν και αναφορές `null`, και κατά συνέπεια τα ορίσματα στις μεθόδους `add()`, `contains()`, and `remove()` μπορούν επίσης να είναι `null`. Η υλοποίησή σας θα γίνει σταδιακά στα παρακάτω 2 υπο-ερωτήματα.

(α) (**16 μονάδες**) Στην πρώτη έκδοση της κλάσης `AbstractMyCollection`, που θα πρέπει να υλοποιήσετε, μπορείτε να υποθέσετε ότι οι μέθοδοι `contains()` και `remove()` δεν καλούνται ποτέ με όρισμα `null`. Μπορείτε δηλαδή να παραλείψετε τον έλεγχο για τις περιπτώσεις όπου το όρισμά τους είναι `null` (Παρατηρείστε ότι η μέθοδος `clear()` δεν επιστρέφει κάποια τιμή).

Λύση:

```
import java.util.*;
public abstract class AbstractMyCollection
                                  implements MyCollection {
  protected AbstractMyCollection () { }
  public abstract boolean add(Object o);
  public void clear() {
   Iterator it = iterator();
   while (it.hasNext()) {
          it.next();
          it.remove();
   }
  } // 3 points
```

```
  public boolean contains(Object o) {
   Iterator it = iterator();
   while (it.hasNext())
    if (o.equals (it.next()))
        return true;
   return false;
  }  // 3 points
  public boolean isEmpty() {
   return (size()==0);
  }  // 3 points
  public abstract Iterator iterator();
  public boolean remove (Object o) {
   Iterator it = iterator();
   while (it.hasNext())
     if (o.equals(it.next())) {
        it.remove();
        return true;
     }
   return false;
  }  // 3 points
  public abstract int size();
}
```

(β) (**12 μονάδες**) Στην τελική έκδοση της κλάσης AbstractMyCollection, θα πρέπει να υλοποιήσετε εκ νέου τις μεθόδους contains() και remove(), έτσι ώστε να συμπεριφέρονται σωστά στις περιπτώσεις όπου το όρισμά τους είναι null.

Λύση:

```
public boolean contains (Object o) {
   Iterator it = iterator();
   if (o == null)
      while (it.hasNext()) {
         if (it.next() == null )
            return true;
      }
   else
      while (it.hasNext())
         if (o.equals(it.next()))
            return true;
   return false;
} // 6 points
public boolean remove(Object o) {
    Iterator it = iterator();
    if (o == null)
       while (it.hasNext()) {
         if (it.next() == null) {
            it.remove();
            return true ;
         }
       }
    else
       while (it.hasNext())
         if (o.equals(it.next())) {
             it.remove();
             return true;
         }
    return false;} // 6 points
```

Άσκηση 4 (15 μονάδες)

Όπως έχουμε παρουσιάσει στο μάθημα, η Java προσφέρει πολλές υλοποιήσεις Αφαιρετικών Τύπων Δεδομένων (ΑΤΔ) για τον χειρισμό συλλογών αντικειμένων οι οποίες επιδεικνύουν διαφορετικές επιδόσεις εκτέλεσης στις λειτουργίες που υποστηρίζουν, όπως:

- Λίστες υλοποιημένες με Πίνακες (`ArrayList`)
- Διπλά Συνδεδεμένες Λίστες (`LinkedList`)
- Σύνολα και Απεικονίσεις υλοποιημένες με Δυαδικά Δέντρα Αναζήτησης (`TreeSet`, `TreeMap`)
- Σύνολα και Απεικονίσεις υλοποιημένες με Πίνακες Κατακερματισμού (`HashSet`, `HashMap`)

Για κάθε μία από τις παρακάτω εφαρμογές, περιγράψτε ποιος πιστεύετε ότι είναι ο καταλληλότερος ΑΤΔ για να σχεδιάσετε (ως προς την λειτουργικότητα της διεπαφής που προσφέρει) και να υλοποιήσετε (ως προς τις επιδόσεις της συγκεκριμένης δομής δεδομένων που χρησιμοποιεί) τις λειτουργίες που σας ζητούνται στην εκφώνηση. Δικαιολογήστε σύντομα την απάντησή σας.

(α) (**5 μονάδες**) *Κατάλογος ενός ηλεκτρονικού καταστήματος παιχνιδιών*. Σε αυτή την εφαρμογή, η πιο συχνή λειτουργία είναι η αναζήτηση (lookup) της περιγραφής ενός παιχνιδιού δεδομένου του μοναδικού κωδικού του. Επίσης προβλέπουμε συχνές ενημερώσεις των πληροφοριών που αφορούν ένα παιχνίδι (πχ. το απόθεμα) κάθε φορά που γίνεται μια σχετική παραγγελία. Τέλος, προσθήκες και διαγραφές παιχνιδιών από τον κατάλογο του καταστήματος δεν εμφανίζονται αρκετά συχνά.

> **Λύση:**
> Lookups for a toy number should be efficient as possible. A `HashMap` is best for this since lookups require constant time.

(β) (**5 μονάδες**) *Έλεγχος συναλλαγών μιας εταιρίας πιστωτικών καρτών*. Η εφαρμογή αυτή καταγράφει κάθε συναλλαγή (δαπάνη, πληρωμή) που η εταιρία πραγματοποιεί εξυπηρετώντας εκατοντάδες συναλλαγές ανά δευτερόλεπτο των πελατών της. Οι καταγραμμένες συναλλαγές διαβάζονται σειριακά έτσι ώστε να ελεγχθούν για την ακρίβειά τους τα στοιχεία κάθε συναλλαγής με αυτά του λογαριασμού του πελάτη που διαθέτει η εταιρία.

> **Λύση:**
> A `LinkedList` is best for this application since the operations that need to be efficient are appending a new item at the end and (less important) accessing the items in the list sequentially.

(γ) (**5 μονάδες**) Λίστα παιχτών με την υψηλότερη βαθμολογία για ένα ηλεκτρονικό παιχνίδι. Όποτε το παιχνίδι τελειώνει, η βαθμολογία ενός παίκτη καταγράφεται. Οι δέκα υψηλότερες βαθμολογίες εμφανίζονται σε φθίνουσα αρίθμηση κάθε φορά που ένα καινούργιο παιχνίδι αρχίζει.

> **Λύση:**
> A sorted `ArrayList` is fine. There are only a small number of items in the list and inserting a new value is done very infrequently (whenever a game is over), so we don't need a fancy data structure to maintain a large sorted collection efficiently.

Άσκηση 5 (10 μονάδες)

Ποια είναι η πολυπλοκότητα (complexity) του χρόνου εκτέλεσης (στην χειρότερη περίπτωση) των παρακάτω μεθόδων που χειρίζονται πίνακες (array); Υποθέστε ότι ο πίνακας έχει μήκος (length) N και ότι η μέθοδος `compareTo()` εκτελείται σε σταθερό χρόνο.

(α) (**2 μονάδες**)

```
boolean inOrder(Comparable[] a, int i) {
  if (i >= a.length-1)
      return true;
  else
      return a[i].compareTo(a[i+1]) < 0;}
```

Λύση:
In general the method performs one comparison of objects. Given that `compareTo()` executes in constant time this method also executes in constant time.

(β) (**2 μονάδες**)

```
int countX(Comparable[] a, Comparable x) {
   int count = 0;
   for (int j=0; j<a.length; j++) {
       if (x.compareTo(a[j])=0) count++;}
   return count;}
```

Λύση:
The loop body executes N times. Each time involves one comparison of objects, and may involve an integer addition. Overall complexity is O(N).

(γ) (**3 μονάδες**)

```
int mostRepeats(Comparable[] a) {
   int most=1;
   for (int j=0; j<a.length; j++) {
        int count = countX(a,a[j]);
        if (count>most) most=count;}
  return most-1;}
```

Λύση:
The loop executes N times. Each time involves a call of `countX()`, whose complexity is O(N). Overall the complexity is $O(N^2)$.

(δ) (**3 μονάδες**)

```
boolean isSorted(Comparable[] a) {
    for (j=0; j<a.length; j++)
         if (!inOrder(a,j)) return false;
    }
```

Λύση:
At worst the loop executes N times, each time calling an O(1) method: in this case complexity is O(N). At best the loop executes once only: in this case complexity is O(1). If the array, a, is Filled randomly then we would expect to exit after one pass through the loop with a probability of 0.5, after two passes with probability 0.25, after three passes with probability 0.125, etc. On average the number of times through the loop will be slightly less than 2. On average, then, this method is constant time.

Άσκηση 6 (20 μονάδες)

Το συμβόλαιο της μεθόδου Arrays.sort(Object[] a) που δίνεται στο πακέτο java.util απαιτεί «όλα τα στοιχεία του πίνακα που ταξινομούνται να υλοποιούν την διεπαφή Comparable. Επιπλέον, όλα τα στοιχεία του πίνακα θα πρέπει να συγκρίνονται ανά δύο (δηλ. η κλήση της μεθόδου e1.compareTo(e2) δεν θα πρέπει να εγείρει την εξαίρεση ClassCastException για οποιαδήποτε στοιχεία e1 και e2 του πίνακα)». Σας δίνεται ακολούθως η διεπαφή Comparable:

```
package java.lang;
public interface Comparable {
  //Returns a negative number if the current object less than obj,
  //a positive number if the current object is greater than obj,
  //or zero if they are equal.
  public int compareTo(Object o);
}
```

Θέλουμε να ταξινομήσουμε έναν πίνακα αντικειμένων της κλάσης Point χρησιμοποιώντας την μέθοδο Arrays.sort(). Υλοποιήστε τις μεθόδους **equals**() και **compareTo**() της κλάσης Point που σας δίνεται παρακάτω:

```
public class Point implements Comparable {
  // Cartesian coordinates of a point on a plane
  private double x, y;
  public Point (double Value, double Value') {
    x = Value;
    y = Value';
  }
  public double getX() { return x; }
  public double getY() { return y; }
  public boolean equals(Object o) {
```

Λύση:

```
if (o instanceof Point) {
    Point p = (Point)o;
    if ((Math.abs(p.getX() - x) < 1e-6) &&
        (Math.abs(p.getY() - y) < 1e-6))
         return true;
    else return false;
   }
return false;
```

```
  } // end equals
  public int compareTo(Object o) {
```

Λύση:

```
if (o instanceof Point) {
    Point p = (Point)o;
    double r = Math.sqrt(x * x + y * y);
    double ro = Math.sqrt(Math.pow(p.getX(),2)+
                Math.pow(p.getY(), 2));
    if (r < ro) return -1;
    if (Math.abs(r - ro) < 1e-6)
         return 0;
    else return 1;
}
return -999;
```

```
 } // end compareTo
} // end Point
```

Σημείωση: Για την υλοποίηση των δύο μεθόδων θα πρέπει να λάβετε υπόψη ότι:

- Η ισότητα δύο αντικειμένων `Point` ορίζεται βάση των τιμών των μεταβλητών στιγμιοτύπων `x` και `y`, οι οποίες θα πρέπει να είναι ίσες με ακρίβεια $1x10^{-6}$ (χρησιμοποιήστε στις συγκρίσεις την μέθοδο `Math.abs()`).
- Η διάταξη δύο αντικειμένων `Point` ορίζεται βάση της απόστασής τους από την αρχή του επιπέδου η οποία ορίζεται από την συνάρτηση $\sqrt{x^2 + y^2}$ (χρησιμοποιήστε στις συγκρίσεις τις μεθόδους `Math.abs()`, `Math.sqrt()` και `Math.pow()`). Επίσης, η μέθοδος `compareTo()` θα πρέπει να επιστρέφει `–1` εάν το τρέχον σημείο είναι κοντύτερα στην αρχή του επιπέδου από το σημείο που δίνεται σαν όρισμα στη μέθοδο. Με αυτόν τον τρόπο, ένα αντικείμενο `Point` που βρίσκεται κοντύτερα στην αρχή του επιπέδου θα προηγείται των αντικειμένων που βρίσκονται μακρύτερα όταν ο πίνακας ταξινομείται.
- Εάν το αντικείμενο (`Object`) που δίνεται σαν όρισμα στη μέθοδο `equals()` δεν είναι στιγμιότυπο της κλάσης `Point`, τότε η μέθοδος επιστρέφει `false`.
- Εάν το αντικείμενο (`Object`) που δίνεται σαν όρισμα στη μέθοδο `compareTo()` δεν είναι στιγμιότυπο της κλάσης `Point`, τότε η μέθοδος επιστρέφει `–999`.

Άσκηση 7 (15 μονάδες)

(α) Ποια είναι η κυριότερη διαφορά μεταξύ μιας αφαιρετικής κλάσης Java και ενός αφαιρετικού τύπου δεδομένων;

Λύση:
Abstract data type: declares value set and operations independent of representation, does not prescribe implementation.
Abstract class: can define state (value representation) and implement methods (operations).
Note that an abstract class may contain instance variables while an interface (e.g., ADT) cannot.

(β) Ποια είναι η βασική ευκολία που προσφέρει η Java με τον χειρισμό των πινάκων σαν αντικείμενα;

Λύση:
Avoids element copy: objects are passed by reference to methods.

(γ) Ποιους περιορισμούς συνέπειας θα πρέπει να λαμβάνουμε υπόψη στην υλοποίηση των μεθόδων `equals()`, `hashCode()`, και `compareTo()`;

Λύση:
if `equals()` is true, `hashCode()` cannot differ.
`equals()` is true exactly when `compareTo()` is zero.

(δ) Δώστε δύο εναλλακτικούς τρόπους για να δηλώνετε ότι μία κλάση δεν μπορεί να έχει και αντικείμενα υποκλάσεών της.

Λύση:
```
final class A { ... } // no subclass allowed
class B { private B () { ... } } // subclass cannot chain
to constructor
```

(στ) Τι είναι ένα πακέτο (package) κώδικα στη Java?

Λύση: Set of related classes with package access rights to each other; sources and class files kept together in directory named after package name.