Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

HY-252 – Αντικειμενοστρεφής Προγραμματισμός
Βασίλης Χριστοφίδης

Επαναληπτική Εξέταση (3 ώρες)
Ημερομηνία: 21 Σεπτεμβρίου 2012

Θέμα 1	Θέμα 2	Θέμα 3	Θέμα 4	Θέμα 5	Θέμα 6	Σύνολο
/25+5	/21	/11	/16	/15	/12	/100+5

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Άσκηση 1 (25+5 μονάδες) // Κατανόηση κώδικα Java
Σας δίνετε το ακόλουθο πρόγραμμα Java οι κλάσεις του οποίου βρίσκονται στο ίδιο πακέτο :

```
class Test {
  public static void main(String[] args){
     A a = new B();
     a.incr(10);
     System.out.println(a);
  }
}
public class A {
  int i;
  public void incr(int d){
     if (d > 0){
          this.i += d;
          this.incr(d-1);
     }
  }
  public String toString(){
     return "A's i is: " + i;
  }
}
public class B extends A {
  int i;
  public void incr(int d){
     if (d > 0){
        this.i += d;
        super.incr(d/2);
     }
  }
  public String toString(){
     return super.toString() + " B's i is: " + i;
  }
}
```

(α) (5 μονάδες) Τι θα τυπωθεί στην στάνταρ έξοδο της μεθόδου `main()`;

Λύση:
When the `incr()` method is called from main, the method invoked is the one in `B`. Both classes provide an `i` field, which by default is 0. A trace of the method calls is the given below

```
B.incr(10)[A.i = 0, B.i = 0]
A.incr(5)[A.i = 0, B.i = 10]
B.incr(4)[A.i=5; B.i=10]
A.incr(2)[A.i=5; B.i=14]
B.incr(1)[A.i=7; B.i=14]
A.incr(0)[A.i=7; B.i=15]
```
Thus, the main method prints: A's i is: 7 B's i is: 15

(β) (5 μονάδες) Θεωρήστε το αντικείμενο το οποίο δημιουργείται στην μέθοδο `main()`, και την κλήση της μεθόδου `incr()`. Η εντολή "`this.i += d`" εμφανίζεται στο σώμα της μεθόδου τόσο στην κλάση `A` όσο και στην κλάση `B`. Είναι η ίδια μεταβλητής "`i`" που τροποποιείται και στις δύο κλάσεις; Η αναφορά "`this`" αναφέρεται στο ίδιο αντικείμενο και στις δύο μεθόδους;

Λύση:
No, there is one 'i' from the A part of the object, and one from the B part. Yes, this refers to the same object (a B object) in both places.

(γ) (15 μονάδες) Στην κλάση `A` η μέθοδος `incr()` δηλώνεται με δημόσια δικαιώματα πρόσβασης (public). Θα άλλαζε η συμπεριφορά του παραπάνω προγράμματος εάν η μέθοδος `incr()` δηλωνόταν σαν προστατευόμενη (protected); Δώστε μια σύντομη εξήγηση της απάντησής σας.
Στην κλάση `B` η μέθοδος `incr()` δηλώνεται με δημόσια δικαιώματα πρόσβασης (public). Θα άλλαζε η συμπεριφορά του παραπάνω προγράμματος εάν η μέθοδος `incr()` δηλωνόταν σαν προστατευόμενη (protected) ενώ στην κλάση `A` παραμένει δημόσια (public); Δώστε μια σύντομη εξήγηση της απάντησής σας.
Εάν οι κλάσεις A και B βρίσκονταν σε ένα πακέτο `X` και η κλάση `Test` σε ένα άλλο πακέτο Y, θα δίνατε διαφορετική απάντηση σε μία από τις προηγούμενες ερωτήσεις; Δώστε μια σύντομη εξήγηση της απάντησής σας.

Λύση:
Yes, it will be OK to declare it protected, one can override a protected method to become public.
It is not OK to declare `B.incr()` protected, as one cannot override a public method to become unpublic. In general, one can make members more visible, not less.
If `A` and `B` were in a different package than `Test`, making `A.incr()` protected would hide it from `Test`, and the program could not compile. Making `B.incr()` protected would still be illegal.

(δ) (5 μονάδες bonus) Είναι έγκυρο να ορίσουμε την κλάση B σαν μέλος-εσωτερική κλάση της A; Δώστε μια σύντομη εξήγηση της απάντησής σας, προσδιορίζοντας και τις αλλαγές που πρέπει να γίνουν στις κλάσεις `Test`, `A` ή `B` στην περίπτωση που η απάντησή σας είναι καταφατική.

Λύση:
Yes, it is legal to have a subclass as an inner class. The only change necessary is in Test, as one will have to instantiate the B class differently. `A a = new A().newB();`

Άσκηση 2 (21 μονάδες) Χειρισμός εξαιρέσεων

Σας δίνεται ο ακόλουθος κώδικας Java:

```
class Front extends ArrayIndexOutOfBoundsException {}
public class ExceptionFunction {
   public static boolean change(int[] course, int index){
          try {
               System.out.println("Start change");
               if (index < 0) { throw new Front(); }
               course[index] = 1;
          }
          catch (Front e) {
               course [-index] = 1;
               System.err.println("OutOfBounds: " +
          index);
               return false;
          }
          catch (ArrayIndexOutOfBoundsException e) {
               course[index - 7] = 1;
               System.err.println("OutOfBounds: " +
               index);
               return true;
          }
          finally {
               System.out.println("In Final Block");
          }
          return false;
     }
     public static void main( String args[] ) {
          try {
               int students[] = new int[5];
               System.out.println("Main");
               System.out.println(change(students,
               Integer.parseInt(args[0])));
               System.out.println("After Change");
          }
          catch (ArrayIndexOutOfBoundsException e) {
               System.out.println("Over Here");
          }
     }
}
```

Τι θα τυπωθεί στην στάνταρ έξοδο μετά τις παρακάτω εκτελέσεις του προγράμματος

(α) (5 μονάδες) `java ExceptionFunction 2`

Λύση:

```
Main
Start change
In Final Block
false
After Change
```

(β) (6 μονάδες) `java ExceptionFunction -2`

Λύση:

```
Main
Start change
OutOfBounds: -2 //inverse occurrence of results is
likely, due to different buffers of standard output
```

```
and standard error that may be flushed at different
times
In Final Block
False
After Change
```

(γ) (6 μονάδες) `java ExceptionFunction 10`

Λύση:
```
Main
Start change
OutOfBounds: 10
In Final Block
true
After Change
```

(δ) (4 μονάδες) `java ExceptionFunction -10`

Λύση:
```
Main
Start change
In Final Block
Over Here
```

Άσκηση 3 (11 μονάδες) // Βασική Λειτουργικότητα Αντικειμένων

Θυμηθείτε την διεπαφή Comparator της Java:

```
public interface Comparator {
     public int compare (Object left, Object right);
}
```

Ορίστε μια κλάση `ReverseIntegerComparator` που υλοποιεί την διεπαφή `Comparator` για την σύγκριση δύο αντικείμενων τύπου `Integer`. Ως συνήθως, η μέθοδος επιστρέφει -1 εάν το `left` είναι αριθμητικά μεγαλύτερο από το `right`, 0 εάν είναι αριθμητικά ίσα και 1 εάν το `left` είναι αριθμητικά μικρότερο από το `right`.

```
public class ReverseIntegerComparator implements
Comparator {
  public int compare(Object left, Object right) {
```

Λύση:
```
if (left instanceof Integer && right instanceof
Integer) {
     Integer Ileft = (Integer) left;
     Integer Iright = (Integer) right;
     if (Ileft.intValue() > Iright.intValue())
                  return -1;
     else if (Ileft.intValue() < Iright.intValue())
                return 1;
     else
                return 0;
}
else
     throw new IllegalArgumentException();
```

```
  }
}
```

Άσκηση 4 (16 μονάδες) Χειρισμός Πλαισίου Συλλογών Αντικειμένων
Δεδομένου ενός δυαδικού δένδρου αναζήτησης (binary search tree) υλοποιήστε μια μέθοδο `between(...)` η οποία παίρνει σαν παράμετρο ένα κατώτατο (lower) και ανώτατο (upper) άκρο και επιστρέφει σε ένα `HashSet` όλα τα στοιχεία του δένδρου που βρίσκονται σε αυτό το διάστημα `[lower, upper]` (συμπεριλαμβανομένων και των δυο άκρων).
Βοήθεια: Για την εισαγωγή ενός στοιχείου x σε ένα `HashSet h` χρησιμοποιήστε την μέθοδο `h.add(x)`. Μπορείτε επίσης να χρησιμοποιήσετε μια βοηθητική μέθοδο `betweenHelper()` η οποία παίρνει σαν παράμετρο ένα επιπλέον `HashSet` το οποίο αποθηκεύει σταδιακά τα στοιχεία του δένδρου που ικανοποιούν την συνθήκη.

```
import java.util.HashSet;
class TreeNode<E extends Comparable> {
 E value;
 TreeNode<E> left;
 TreeNode<E> right;
 HashSet<E> between(Comparable lower, Comparable upper){
```

Λύση:
```
return betweenHelper(lower,upper, new HashSet<E>());
```

```
 }
 HashSet<E> betweenHelper(Comparable lower,
                          Comparable upper,HashSet<E> h) {
```

Λύση:
```
  if (value.compareTo(upper) <= 0 &&
      value.compareTo(lower) >= 0)
     h.add(value);
  if (left != null && value.compareTo(lower) >= 0)
     h = left.betweenHelper(lower, upper, h);
  if (right != null && value.compareTo(upper) <= 0)
     h = right.betweenHelper(lower, upper, h);
  return h;
```

```
 }
}
```

Άσκηση 5 (15 μονάδες) // Γενικές Διαπαφές και Κλάσεις
Σας δίνεται η ακόλουθη γενική κλάση `DynamicArray<E>` η οποία υλοποιεί την γενική διεπαφή `Array<E>`:

```
public class DynamicArray<E> implements Array<E> {
  private E[] data; // the data part of the array
  private int size; // logical size of the array
  public DynamicArray(int initalCapacity) {...}
  public DynamicArray() {...}
  public DynamicArray(DynamicArray<E> b) {...}
  // interface methods are implemented here.
 }
```

(α) (5 μονάδες) Υλοποιήστε την μέθοδο `resize()` η οποία δημιουργεί έναν πίνακα διπλάσιου μεγέθους από αυτόν που αναφέρετε από την μεταβλητή `data` και αντιγράφει τα στοιχεία του στον καινούργιο πίνακα.

```
private void resize() {
```

Λύση:
```
int newCapacity = 2 * data.length;
E[] newData = (E[]) new Object[newCapacity];
for (int k = 0; k < data.length; k++) {
   newData[k] = data[k];
  }
data = newData;
}
```

(β) (5 μονάδες) Υλοποιήστε την μέθοδο add() η οποία προσθέτει ένα στοιχείο (element) που δίνεται σαν παράμετρος στο τέλος του πίνακα εάν υπάρχει διαθέσιμος χώρος. Διαφορετικά επεκτείνει τον πίνακα. Η μέθοδος επιστρέφει πάντοτε *αληθές*.

```
public boolean add(E element) {
```

Λύση:
```
if (size == data.length)
   resize();
data[size] = element;
size++;
return true;
}
```

γ) (5 μονάδες) Υλοποιήστε την μέθοδο add() η οποία προσθέτει ένα στοιχείο στην θέση κ που δίνεται σαν παράμετρος, μεταθέτοντας τα στοιχεία που έπονται κατά μια θέση. Σε περίπτωση που δεν υπάρχει διαθέσιμος χώρος για αυτή την εισαγωγή επιστρέφει μια εξαίρεση IndexOutOfBoundsException.

```
public void add(int k, E element) {
```

Λύση:
```
if (k < 0 || k > data.length)
  throw new IndexOutOfBoundsException("add: index
out of bounds");
//Shift element to make room for new one.
//This works if element is added at end (k=size())
for (int j = size-1; j >= k; j--)
  data[j+1] = data[j];
data[k] = element; // add the element
size++;
}
```

Άσκηση 6 (12 μονάδες) //Προγραμματισμός Βασισμένος σε Συμβόλαια

Σας δίνονται οι ακόλουθες κλάσεις και διεπαφές:

```
public class FooClass {
     public int getValue ( ) { . . . }
     public void run ( ) { . . . }
}
public interface Openable {
     public void open ( ) ;
```

```
}
public interface Closable {
     public void close ( ) ;
}
```

Θέλουμε να γράψουμε μια επέκταση της κλάσης FooClass η οποία επεκτείνει και τις δύο παραπάνω διεπαφές:

```
public class BarClass extends FooClass,
                        implements Openable, Closable {...}
```

(α) (3 μονάδες) Ποιες μεθόδους πρέπει να υλοποιεί η κλάση `BarClass` έτσι ώστε να μπορεί να είναι συντακτικά σωστή;

> **Λύση:**
> `open()` and `close()`

(β) (3 μονάδες) Ποιες από τις παρακάτω εντολές είναι σωστές (υποθέτοντας ότι οι κλάσεις διαθέτουν μεθόδους κατασκευής);

```
i. BarClass a = new FooClass ( ) ;
ii. FooClass b = new BarClass ( ) ;
iii. Openable c = new FooClass ( ) ;
iv. FooClass d = new Closable ( ) ;
```

> **Λύση:**
> `i.`

(γ) (3 μονάδες) Ποιες από τις παρακάτω εντολές της μεθόδου `funMethod()` είναι σωστές;

```
public void funMethod (BarClass a , FooClass b) {
i.     a.close ( ) ;
ii.    a.run ( ) ;
iii.   b.getValue ( ) ;
iv.    a.open ( ) ;
}
```

> **Λύση:**
> `i., , iii., iv.`

(δ) (3 μονάδες) Εξηγήστε την σχέση μεταξύ μιας κλάσης και ενός αντικειμένου.

> **Λύση:**
> A class describes the type of a group of objects. The type is the data in an object and the operations that can be performed on it. An object is some data in memory whose type is defined by a class.