Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών
HY-252 – Αντικειμενοστρεφής Προγραμματισμός
Βασίλης Χριστοφίδης
Πρόοδος (3 ώρες)
Ημερομηνία: 24 Νοεμβρίου 2012

Άσκηση1	Άσκηση2	Άσκηση3	Άσκηση4	Άσκηση5	Άσκηση6	Άσκηση7	Σύνολο
/16	/20	/10	/12	/23	/12	/11	/104

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Άσκηση 1 (16 μονάδες) // Στατικοί και Δυναμικοί Τύποι Αντικειμένων

Σας δίνονται οι ακόλουθοι ορισμοί κλάσεων Java:

```
public class A {
      A() { System.out.println("A.A()"); }
      public void f(A arg) { System.out.println("A.f(A)"); }
      public void g(A arg) { System.out.println("A.g(A)"); }
}
public class B extends A {
      B() { System.out.println("B.B()"); }
      public void f(B arg) { System.out.println("B.f(B)"); }
      public void g(A arg) { System.out.println("B.g(A)"); }
      public static void h(A arg) { System.out.println("B.h(A)"); }
}
public class C extends B {
      C() { System.out.println("C.C()"); }
      public void f(A arg) { System.out.println("C.f(A)"); }
      public void f(B arg) { System.out.println("C.f(B)"); }
      public void g(B arg) { System.out.println("C.g(B)"); }
      public static void h(A arg) { System.out.println("C.h(A)"); }
}
```

Ποια είναι η έξοδος του ακόλουθου προγράμματος όταν όλες οι εντολές εκτελεστούν ακολουθιακά (π.χ.εκτυπώνεται xxx,δημιουργείται εξαίρεση yy,υπάρχει λάθος μεταγλώττισης zz);

```
1. A a = new A();
2. A ab = new B();
3. B bc = new C();
4. C c = new C();
5. a.f(bc);
6. bc.h(c)
7. bc.f(ab);
8. c.g(a);
9. c.g(bc);
10. ab.h(ab);
11. ab.g(ab);
```

Λύση:

```
1. A.A()
2. A.A() B.B()
3. A.A() B.B() C.C()
4. A.A() B.B() C.C()
5. A.f(A)
6. B.h(a)
7. C.f(A)
8. B.g(A)
9. C.g(B)
10. compile time error
11. B.g(A)
```

Άσκηση 2 (20 μονάδες) // Κατανόηση κώδικα Java

Σας δίνονται οι παρακάτω ορθοί ορισμοί (δηλ. χωρίς προβλήματα μεταγλώττισης ή εκτέλεσης) τεσσάρων κλάσεων, `ClassA`, `ClassB`, `ClassC` και `TestClass`. Τι θα εκτυπωθεί κατά την εκτέλεση της μεθόδου `main()` της `TestClass`.

```
public abstract class ClassA {
     private int num1;
     private int num2;
     public ClassA (int n1, int n2) {
          num1 = n1;
          num2 = n2;
     }
     public abstract int sumNums();
     public abstract int subNums();
     public abstract int subNums(char c);
     public int timesNums() {
          return num1 * num2;
     }
     public int getNum1 () {
          return num1;
     }
     public int getNum2 () {
          return num2;
     }
} // end ClassA

public class ClassB extends ClassA {
     private int num1;
     public ClassB (int n1, int n2, int n3) {
          super(n1, n2);
          num1 = n3;
     }
     public ClassB (int n) {
          super(n, n + 1);
          num1 = n;
     }
     public int sumNums () {
          return num1 + getNum1() + getNum2();
     }
     public int timesNums() {
          return super.timesNums() * num1;
     }
     public int subNums () {
          return num1 - super.getNum2() - super.getNum1();
     }
     public int subNums (char dir) {
          if (dir == 'f')
               return super.getNum1() - super.getNum2() - num1;
          else
               return subNums();
     }
     public int getNum1 () {
          return num1;
     }
} //end ClassB

public class ClassC extends ClassB {
     private int num1;
```

```
      public ClassC (int n1, int n2, int n3, int n4) {
            super(n1, n2, n3);
            num1 = n4;
      }
      public ClassC (int n) {
            super(n);
            num1 = n;
      }
      public int timesNums() {
            return super.timesNums() * num1;
      }
} //end ClassC

public class TestClass {
      public static void main (String[] args) {
      ClassA b = new ClassB(2, 3, 4);
      System.out.println(b.getNum1());
```

Λύση: 4

```
      System.out.println(b.sumNums());
```

Λύση: 11

```
      System.out.println(b.timesNums());
```

Λύση: 24

```
      System.out.println(b.subNums('f'));
```

Λύση: -5

```
      System.out.println(b.subNums());
```

Λύση: -1

```
      ClassA c = new ClassC(3);
      System.out.println(c.timesNums());
```

Λύση: 108

```
      System.out.println(c.sumNums());
```

Λύση: 10

```
      System.out.println(c.getNum1());
```

Λύση: 3

```
      System.out.println(c.getNum2());
```

Λύση: 4

```
      ClassA cc = new ClassC(5, 6, 7, 8);
      System.out.println(cc.sumNums());
```

Λύση: 20

```
      } //end main
} //end TestClass
```

Άσκηση 3 (10 μονάδες) //Χειρισμός Εξαιρέσεων

Ποια είναι ή έξοδος του παρακάτω προγράμματος Java;

```
1. public class Test {
2.     public static void aMethod() throws Exception {
3.        try {
4.            throw new Exception();
5.        }
6.        finally {
7.           System.out.print("finally ");
8.        }
9.     }
10.    public static void main(String args[]) {
11.       try {
12.           aMethod();
13.       }
14.       catch (Exception e) {
15.           System.out.print("exception ");
16.       }
17.       System.out.print("finished");
18.    }
19.}
```

Δικαιολογείστε την απάντησή σας.

```
A.   finally
B.   exception finished
Γ.   finally exception finished
Δ.   compilation fails
```

Λύση: Γ

This is what happens:
(1) The execution of the try block (line 3) completes abruptly because of the throw statement (line 4).
(2) The exception cannot be assigned to the parameter of any catch clause of the try statement therefore the finally block is executed (line 6) and "finally" is output (line 7).
(3) The finally block completes normally, and then the try statement completes abruptly because of the throw statement (line 4).
(4) The exception is propagated up the call stack and is caught by the catch in the main method (line 14). This prints "exception".
(5) Lastly program execution continues, because the exception has been caught, and "finished" is output (line 17).

Άσκηση 4 (12 μονάδες) // Χειρισμός Πινάκων και Ισχυρισμοί

Υποθέστε ότι χρησιμοποιούμε πίνακες Java δύο διαστάσεων που περιέχουν ακέραιους. Η συνάρτηση `flip()`, αναστρέφει τον πίνακα από αριστερά προς τα δεξιά. Εδώ, σας δίνεται ένα JUnit test, που σας υποδεικνύει τη σωστή συμπεριφορά της μεθόδου. (Η μέθοδος `assertArrayEquals` ελέγχει στοιχείο-προς-στοιχείο την ισότητα των πινάκων.)

```
int[][] input=new int[] {{1, 2, 3, 4},{5, 6, 7, 8},{9,10,11,12}};
int[][] expectedOutput=new int[]{{4,3,2,1},{8,7,6,5},{12,11,10,9}};
flip(input);
assertArrayEquals(input, expectedOutput);
```

Συμπληρώστε την υλοποίηση της μεθόδου `flip()`. Σημειώστε ότι η μέθοδος αντί να επιστρέφει ένα νέο πίνακα με ανεστραμμένα περιεχόμενα, θα πρέπει να αναδιοργανώνει τις τιμές του πίνακα που περνάτε ως όρισμα. Μπορείτε να υποθέσετε ότι τόσο ο πίνακας εισόδου όσο και οι υποπίνακες του δεν περιλαμβάνουν `null` τιμές και έχουν ίδιο μήκος.

```
public static void flip(int[][] input) {
```

Λύση:

```
for (int row = 0; row < input.length; row++) {
   for (int col = 0; col < (input[row].length / 2); col++) {
       int ocol = input[row].length - col - 1;
       int tmp = input[row][col];
       input[row][col] = input[row][ocol];
       input[row][ocol] = tmp;
   }
}
```

```
}
```

Άσκηση 5 (23 μονάδες) // Υλοποίηση Κλάσεων και Κληρονομικότητα

Σας ζητείται να επεκτείνεται την ήδη υπάρχουσα κλάση `Point`, η οποία αναπαριστά δισδιάστατες 2-D (x, y) συντεταγμένες. Η κλάση `Point` περιλαμβάνει τους ακόλουθους κατασκευαστές (constructors) και μεθόδους:

```
public Point()// constructs the point (0, 0)
public Point(int x, int y)//constructs a point with the given x/y
coordinates
public void setLocation(int x, int y) // sets the coordinates to the
given values
public int getX() // returns the x-coordinate
public int getY() // returns the y-coordinate
public String toString() //returns a String in standard "(x, y)" notation
public double distanceFromOrigin() //returns the distance from the origin
(0, 0), computed as the square root of (x² + y²)
```

Πρέπει να ορίσετε μία νέα κλάση που ονομάζεται `Point3D` η οποία επεκτείνει την `Point` μέσω κληρονομικότητας. Η `Point3D` θα πρέπει να συμπεριφέρεται όπως η μητρική κλάση με τη διαφορά ότι θα αναφέρεται σε τρισδιάστατες αντί για δυσδιάστατες συντεταγμένες (x,y,z). Οι επιπλέον μέθοδοι της κλάσης `Point3D` που πρέπει να υλοποιηθούν είναι:

```
public Point3D() //constructs the point (0, 0, 0)
public Point3D(int x,int y,int z) //constructs a point with given x/y/z
coordinates
public void setLocation(int x,int y,int z)//sets coordinates to the given
values
public int getZ() //returns the z-coordinate
```

Κάποιες από τις μεθόδου της κλάσης `Point` που θα πρέπει να υλοποιηθούν ξανά στην Point3D γιατί εμφανίζουν διαφορετική συμπεριφορά είναι:

- Όταν καλείται η αρχική έκδοση στην `Point` της μεθόδου `setLocation(x,y)` θα πρέπει να αναθέτει τις x- και y-συντεταγμένες του 3-D σημείου όπως καθορίζει το συμβόλαιο της `Point`, ενώ η z-συντεταγμένη θα τεθεί ίση με 0.
- Όταν ένα 3-D σημείο τυπώνεται με χρήση της μεθόδου `toString()`, θα πρέπει να επιστρέφει συμβολοσειρά: "(x, y, z)", ώστε να εμφανίζονται και οι τρεις συντεταγμένες.
- Η απόσταση ενός 3-D σημείου από την αρχή των αξόνων, υπολογίζεται χρησιμοποιώντας και τις τρεις συντεταγμένες. Η απόσταση ισούται με την τετραγωνική ρίζα του $(x^2 + y^2 + z^2)$.

Πρέπει επίσης να φτιάξετε αντικείμενα τύπου `Point3D`, τα οποία θα είναι συγκρίσιμα το ένα με το άλλο, χρησιμοποιώντας τη διεπαφή (interface) `Comparable`. Τα 3-D σημεία συγκρίνονται ως προς τη x-συντεταγμένη, στην συνέχεια ως προς την y-συντεταγμένη και τέλος ως προς την z-συντεταγμένη. Δηλαδή, ένα αντικείμενο `Point3D` με μικρότερη τη x-συντεταγμένη, θεωρείται ότι “προηγείται από” ένα άλλο αντικείμενο με μεγαλύτερη τη x-

συντεταγμένη. Αν δύο αντικείμενα Point3D έχουν την ίδια x-συντεταγμένη, αυτό με τη μικρότερη y- συντεταγμένη θεωρείται ότι "προηγείται". Αν τα δύο αντικείμενα έχουν ίδιες τις x και y-συντεταγμένες, αυτό με τη μικρότερη z-συντεταγμένη θεωρείται ότι "προηγείται". Αν δύο αντικείμενα έχουν τις ίδιες x, y και z-συντεταγμένες τότε θεωρούνται "ίσα".

```
public class Point3D extends Point implements Comparable {
```

Λύση:

```
private int z;  //1 point
public Point3D() {
  this(0, 0, 0);
}//2 points
public Point3D(int x, int y, int z) {
  // super(x, y);
  // this.z = z;
  setLocation(x, y, z);
}//2 points
public int getZ() {
  return z;
} //2 points
public String toString() {
  return "(" + getX() + ", " + getY() + ", " + z + ")";
}//2 points
public void setLocation(int x, int y) {
  setLocation(x, y, 0);
}//2 points
public void setLocation(int x, int y, int z) {
  super.setLocation(x, y);
  this.z = z;
}//3 points
public double distanceFromOrigin() {
  return Math.sqrt(getX() * getX() + getY() * getY() + z * z);
}//2 points
public int compareTo(Point3D other) {
  if (getX() != other.getX()) {
    return getX() - other.getX();
  } else if (getY() != other.getY()) {
    return getY() - other.getY();
  } else {
    return z - other.z
  }
}//7 points
}
```

Άσκηση 6 (12 μονάδες) // Υλοποίηση Αφαιρετικών Τύπων Δεδομένων

Το πακέτο `java.util` περιέχει μια διεπαφή (interface) για την αναπαράσταση μια ουράς στην οποία τα αντικείμενα προστίθενται (add) στο ένα άκρο (tail) και αφαιρούνται (pool) από το άλλο (front). Θεωρούμε την παρακάτω παραλλαγή αυτής της διεπαφής για την αναπαράσταση ουρών από συμβολοσειρές όπου η μέθοδος `poll()` αφαιρεί και επιστρέφει μια συμβολοσειρά από την κεφαλή της ουράς (ή επιστρέφει `null` εάν η ουρά είναι κενή):

```
/**
* This interface represents a collection of objects
* called a "queue" in which new Strings are added at the
* end of the queue and removed from the front, giving
```

```
* rise to a typical first-come/first-served waiting line.
*/
public interface MinimalStringQueue {
 /** Adds a new String to the end of the queue */
 public void add(String str);
 /** Removes and returns the first String (or null if queue is
empty) */
 public String poll();
 /** Returns the number of entries in the queue. */
 public int size();
}
```

Σας ζητείται να υλοποιήσετε την διεπαφή `MinimalStringQueue` σε μια κλάση `StringQueue` η οποία αποθηκεύει στην μεταβλητή στιγμιοτύπων (`waitingLine`) τύπου `Vector` τις συμβολοσειρές της ουράς. Ένα παράδειγμα χρήσης της κλάσης `StringQueue` με συμβολοσειρές από την χριστουγεννιάτικη ιστορία του Κάρολου Ντίκενς σας δίνεται στην συνέχεια:

```
StringQueue queue = new StringQueue ();
queue.add("Christmas Past");
queue.add("Christmas Present");
queue.add("Christmas Future");
```

Η κλήση `queue.size()` θα επιστέψει 3 ενώ η πρώτη κλήση της `queue.poll()` θα επιστρέψει `"Christmas Past"` η δεύτερη `"Christmas Present"`, η τρίτη `"Christmas Future"`, και η τέταρτη `null`.

```
*
* File: StringQueue.java
* ----------------------
* This program implements the MinimalStringQueue interface
* using a Vector for internal storage.
*/
import java.util.*;
/** Implements a Vector queue */
public class StringQueue implements MinimalStringQueue {
```

Λύση:

```
/* Private instance variables */
private Vector waitingLine; // 1 point
/** Creates a new empty queue. */
public StringQueue() {
  waitingLine = new Vector();
} // 2 points
/** Adds a new String to the end of the queue */
public void add(String str) {
  waitingLine.addelement(str);
} // 2 points
/** Removes and returns the first String (or null if queue is
empty) */
public String poll() {
  if (waitingLine.isEmpty()) return null;
  String first = (String) waitingLine.elementAt(0);
  waitingLine.removeElementAt(0);
  return first;
} // 5 points
/** Returns the number of entries in the queue. */
public int size() {
  return waitingLine.size();
} //2 points
```

```
}
```

Άσκηση 7 (11 μονάδες) // Θεωρία Αντικειμενοστραφούς Κώδικα

(α) **(6 μονάδες)** Για κάθε μία από τις ακόλουθες περιπτώσεις υποδείξτε εάν ενδείκνυται η δημιουργία μιας ελεγμένης (checked) ή μη-ελεγμένης (unchecked) εξαίρεσης. Δικαιολογήστε σύντομα την απάντησή σας.
(i) Ένας κώδικας πελάτης χρησιμοποιεί ένα όρισμα (argument) που παραβιάζει μια πρωθύστερη (precondition) συνθήκη.
(ii) Ένας κώδικας πελάτης χρησιμοποιεί ένα όνομα αρχείου το οποίο δεν μπορεί να βρεθεί.
(iii) Ένας υλοποιητικός κώδικας παρουσιάζει ένα σφάλμα το οποίο μπορεί να εντοπιστεί από δικούς του ελέγχους.

Λύση:
Checked exceptions represent unusual or unexpected events such as being unable to open a file or a dropped network connection. While they are not necessarily expected to occur, they are things a client might want to handle. It is appropriate to document them as part of a method's interface if they might be thrown by the method, or require that the method catch them if they should not be visible outside
On the other hand, unchecked exceptions generally indicate programming errors. They could appear anywhere in the code and documenting them would pointlessly clutter up interfaces.
Thus, we should use a checked exception when it is essential that the client handle the condition. We should use se an unchecked exception if the client can prevent the exception (due to knowledge about the application's logic or due to a check), or if the exception is so severe that the client can't do anything about it.
(i) The client supplied an argument that violates a precondition. unchecked
(ii) The client specified a file name, but the file cannot be found. checked
(iii) The implementation contains a bug that its own self-checks detected. unchecked

(β) **(5 μονάδες)** Υποθέστε ότι δεν μπορούμε να δημιουργήσουμε στιγμιότυπα μια κλάσης Α. Πώς θα μπορούσε ένας κώδικας πελάτης να χρησιμοποιήσει την κλάση Α; Δικαιολογήστε σύντομα την απάντησή σας.

Λύση:
There are two reasons that a class cannot be instantiated: its constructors always throw an exception, or the class is abstract. In either case, a client can call static methods that C defines. If the class is abstract, its non-static methods can be used by a subclass. (For full credit for the latter answer, you would have had to mention that the class was abstract.)