Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

ΗΥ-252 – Αντικειμενοστρεφής Προγραμματισμός

Διδάσκων: Γιάννης Τζίτζικας

Τελική Εξέταση (3 ώρες)

Ημερομηνία: 25 Ιανουαρίου 2008
Σύνολο μονάδων: 120 (δηλαδή δύο μονάδες bonus)

Άσκηση 1 (5 μονάδες) // κατανόηση διαχείρισης μνήμης και garbage collector

Θεωρείστε ότι έχουμε ορίσει μια κλάση Person με μια κατασκευάστρια μέθοδο η οποία δέχεται ως παράμετρο μια συμβολοσειρά. Ο παρακάτω κώδικας θα δημιουργήσει προβλήματα μνήμης; Πιο συγκεκριμένα: πότε θα δημιουργήσει και πότε όχι; Δώστε μια σύντομη απάντηση.

```
int i=0;
while (1) {
      Person p = new Person("anonymous"+(i++));
      // hidden code
}
```

Λύση
Αν ο κρυμμένος κώδικας κρατάει αναφορές στα αντικείμενα που δημιουργούνται τότε θα έχουμε πρόβλημα μνήμης.
Αν ο κρυμμένος κώδικας δεν κρατάει αναφορές τότε δεν θα έχουμε πρόβλημα μνήμης διότι ο garbage collector θα αποδεσμεύει το χώρο που καταλαμβάνουν τα νέα αντικείμενα που δημιουργούνται

Άσκηση 2 (5 μονάδες) // απλή διαχείριση συλλογών

Είναι ο παρακάτω κώδικας λογικά σωστός; Δηλαδή θα επιτελέσει το επιδιωκόμενο έργο; Αν όχι βρείτε το λάθος/λάθη και δικαιολογείστε σύντομα.

```
/* removes from a vector v all references pointing to an object obj */
void removeElementFromVector(Vector v, Object obj) {
      int i=0;
      while (i < v.size()) {
            if (v.elementAt(i)==obj)
                  v.removeElementAt(i);
            i++;
      }
}
```

Λύση
Ο παραπάνω κώδικας θα αποτύχει να σβήσει συνεχόμενες εμφανίσεις του obj. Θα ήταν σωστός αν υπήρχε ένα `else` πριν την εντολή `i++`

Άσκηση 3 (20 μονάδες) // εξαιρέσεις

Ο παρακάτω κώδικα δεν μπορεί να μεταγλωττιστεί. Εξηγείστε το γιατί.
Κατόπιν δώστε δύο διαφορετικούς τρόπους συμπλήρωσης του κώδικα ώστε να μπορεί να μεταγλωττιστεί.
Ο ένας από αυτούς να μην απαιτεί καμία αλλαγή του κώδικα της class B.
Ο δεύτερος τρόπος να έχει μια έκδοση όπου οι υπογραφές των μεθόδων της B να παραμείνουν ως έχουν και μια έκδοση όπου τα σώματα των μεθόδων της B να παραμείνουν ως έχουν (ενώ οι υπογραφές τους μπορεί να διαφέρουν).

```
class A {
      public void methodA(int a){
            if (a <0) throw new XException();
      }
```

```
}

class B extends A {
	public void methodA(int a) {
		super.methodA(a);
	}
}
```

Λύση

(5 μονάδες)
Δεν θα μεταγλωττιστεί διότι δεν έχει δηλωθεί η εξαίρεση XException

(7 μονάδες)
Τρόπος Α:
Δηλώνουνε το XException ως unckecked exception. Αυτό μπορεί να γίνει με μια δήλωση της μορφής:
`class XException extends RuntimeException {}`
Δεν χρειάζεται να κάνουμε καμία άλλη αλλαγή στον κώδικα.

(8 μονάδες)
Τρόπος Β:
Δηλώνουμε το XException ως checked exception, π.χ. με μια δήλωση της μορφής:
`class XException extends Exception {}`
Σε αυτή την περίπτωση πρέπει να αλλάξουμε την επικεφαλίδα της methodA της class A, δηλαδή να προσθέσουμε το throws XException.
Η methodA της class B μπορεί είτε να κάνει και αυτή throws (και πρέπει να ενημερώσουμε την υπογραφή της) ή στο σώμα η εντολή `super.methodA(a)` να είναι μέσα σε ένα try catch και η υπογραφή της να παραμείνει ως έχει.

Άσκηση 4 (15 μονάδες) // reflection

Γράψτε ένα μικρό πρόγραμμα το οποίο να διαβάζει μέσω του JOptionPane μια συμβολοσειρά. Κατόπιν να ελέγχει αν υπάρχει κλάση με αυτό το όνομα και αν ναι να τυπώνει όλα τα ονόματα των μεθόδων της κλάσης αυτής. Αν δεν υπάρχει κλάση με αυτό το όνομα τότε να ζητάει από το χρήστη μια άλλη συμβολοσειρά και αυτό να συνεχίζεται έως ότου ο χρήστης δώσει ένα string που αντιστοιχεί σε κλάση που υπάρχει.
Σημείωση:
Απο την κλάση Class, αρκεί να χρησιμοποιήσετε την στατική μέθοδο forName(String) και τη μέθοδο Medhod[] getDeclaredMethods(). Μπορείτε να διαβάσετε μια συμβολοσειρά μέσω του JOptionPane ως εξής: String userName = JOptionPane.showInputDialog("Give me your name");

Λύση

```
import java.lang.reflect.*;
import javax.swing.JOptionPane;

class Utilities {
	static final boolean printClassInformation(String str) {
		try {
			Class c = Class.forName(str);
			Method[] ms = c.getDeclaredMethods();
			for (int i = 0; i < ms.length; i++)
			 System.out.println(ms[i].toString());
		} catch(ClassNotFoundException e) {
			System.out.println(e);
			return false;
		}
		return true;
	}
}


class testtest {
	public static void main(String[] arg){
		boolean ok=false;
		do {
			String str = JOptionPane.showInputDialog("Give a class name");
			ok = Utilities.printClassInformation(str);
		} while (!ok) ;
	}
}
```

```
ΟΔΗΓΙΕΣ ΔΙΟΡΘΩΣΗΣ
1/ Ορθότητα ροής ελέγχου και κώδικα (10 μονάδες)
2/ try-catch για την εξαίρεση (3 μονάδες)
3/ Δόμηση κώδικα (2 μονάδες)
```

Άσκηση 5 (10 μονάδες) // κατανόηση κώδικα και συλλογών

Ο παρακάτω κώδικας είναι γραμμένος σε ένα αρχείο με το όνομα test.java. Ποιο θα είναι το αποτέλεσμα της εκτέλεσης αυτού του προγράμματος; Περιγράψτε σύντομα (με λιγότερες από 10 γραμμές) τι κάνει αυτό το πρόγραμμα και τι θα τυπώσει στην κονσόλα.

```
import java.io.*;
import java.util.*;

class Mysterious {
  public  void dowork( String param ) {
   if ( param==null ) {    System.exit( 1 );  }
   StreamTokenizer in = null;
   try {
       in = new StreamTokenizer( new BufferedReader (
                                 new FileReader ( param ) ) );
   } catch ( FileNotFoundException e ) {
           System.err.println( e.getMessage() );
           System.exit(1);
   }
   try {
           HashMap map = new HashMap();
           Integer one = new Integer(1);
           while ( ( in.nextToken() != in.TT_EOF ) ) { // while we haven't reached EOF
                    if ( in.ttype == in.TT_WORD ) { // the current token is a word
                        Integer freq = ( Integer ) map.get(in.sval); //in.sval is the token
                        if ( freq == null )
                           freq = one;
                        else
                           freq = new Integer( freq.intValue() + 1 );
                           map.put( in.sval, freq );
                    }
           }
           SortedSet tmp = new TreeSet(map.keySet());
           String z = (String)tmp.last();
           System.out.println(z + " " + map.get(z));
       } catch ( IOException e ) {
              System.err.println(e.getMessage() );
              System.exit( 1 );
       }
  }
}
class mtester {
       public static void main( String args[] ) {
              Mysterious m = new Mysterious();
              m.dowork("test.java");
       }
}
```

Απάντηση.
Θα τυπώσει στην κονσόλα τη γραμμή
z 3 (3 μονάδες)
Συγκεκριμένα το πρόγραμμα αυτό διαβάζει τον πηγαίο κώδικα του εαυτού του, βρίσκει τις διαφορετικές λέξεις που εμφανίζονται σε αυτό και πόσες φορές εμφανίζεται η κάθε μία τους. Τέλος τυπώνει την λέξη που είναι τελευταία λεξικογραφικά και το πλήθος των εμφανίσεων της. Στην προκειμένου (και αφού οι χαρακτήρες είναι λατινικοί) η τελευταία λέξη είναι η "z" η οποία εμφανίζεται 3 φορές. (7 μονάδες)

Άσκηση 6 (15 μονάδες) // maps

Μία συνάρτηση $f : A \rightarrow B$ λέγεται ένα προς ένα (1-1) όταν αντιστοιχίζει κάθε όρισμα σε αποκλειστικά δική του τιμή, δηλαδή όταν διαφορετικά ορίσματα απεικονίζονται σε διαφορετικές τιμές. Πιο συγκεκριμένα αν $a \neq a'$ τότε $f(a) \neq f(a')$.

Μία συνάρτηση $f : A \rightarrow B$ λέγεται επί, όταν δεν υπάρχει στοιχείο στο B που να μην είναι η εικόνα κάποιου στοιχείου στο A. Δηλαδή αν για κάθε $b \in B$ υπάρχει $a \in A$ τέτοιο ώστε $b = f(a)$.

Η αντίστροφη αντιστοίχιση f^{-1} της συνάρτησης f είναι η αντιστοίχιση από το B στο A, που ορίζεται ως εξής: $f^{-1}(b) = a$ ανν $f(a) = b$. Αν μια f είναι 1-1 και επί, τότε η αντίστροφη της είναι συνάρτηση.

Συμπληρώστε τα τρία κενά στον παρακάτω κώδικα ώστε η κλάση να προσφέρει την προσδοκώμενη λειτουργικότητα.

```
final class FunctionUtilities {
	static boolean isEnaProsEna(Map f){
		Set s = new HashSet(f.values());
		if (f.size() == s.size()) return true;
		else return false;
	}
	static boolean isEpi(Map f){
		//complete code
	}
	static boolean isAntistrepsimi(Map f){
		//complete code
	}

	static Map reverse(Map f){
		//complete code
	}
```

Λύση

```
final class FunctionUtilities {
	static boolean isEnaProsEna(Map f){
		Set s = new HashSet(f.values());
		if (f.size() == s.size()) return true;
		else return false;
	}
	static boolean isEpi(Map f){
		return true;
	}
	static boolean isAntistrepsimi(Map f){
		return (isEnaProsEna(f) && isEpi(f));
	}

	static Map reverse(Map f){
		if (isAntistrepsimi(f)) {
			Map g = new HashMap();
			Set<Map.Entry> s = f.entrySet();
			for (Map.Entry e: s) {
				g.put(e.getValue(), e.getKey());
			}
			return g;
		}
		else throw new IllegalArgumentException("Not reversible");
	}
```

Οδηγίες Βαθμολόγησης
5μ για την isEpi
5μ για την isAntistrepsimi
5μ για την reverse
(αφαίρεση 2μ εάν δεν έχει γίνει ο έλεγχος isAntistrepimi)

Άσκηση 7 (30 μονάδες) // σωστό ή λάθος

Μαρκάρετε με Σ τις Σωστές και με Λ τις λανθασμένες.

ΑΑ	Πρόταση	Σ/Λ	Απάντηση
1	Σε μια member inner class μπορούμε να χρησιμοποιήσουμε το this	Σ	Σωστό
2	Οι κατασκευάστριες γεννήτριες πρέπει να είναι πάντα public	Λ	Λάθος
3	Αν έχουμε σε μια μέθοδο την εντολή throw new IllegalArgumentException()	Λ	Λάθος

	τότε πρέπει να έχουμε δηλώσει αυτήν την εξαίρεση στην υπογραφή της μεθόδου		
4	Η δημιουργία πολλών αντικειμένων μπορεί να οδηγήσει σε StackOverflowError	Λ	Λάθος
5	Αν έχουμε σε μια μέθοδο την εντολή throw new Exception() τότε πρέπει να έχουμε δηλώσει αυτήν την εξαίρεση στην υπογραφή της μεθόδου	Σ	Σωστό (εκτός αν η εν λόγω εντολή είναι σε try-catch)
6	Αν έχουμε σε μια μέθοδο την εντολή throw new RuntimeException() τότε πρέπει να έχουμε δηλώσει αυτήν την εξαίρεση στην υπογραφή της μεθόδου	Λ	Λάθος
7	Η κλάση TreeMap μας επιτρέπει να παριστάνουμε γράφους	Λ	Συναρτήσεις που τα κλειδιά τους είναι ταξινομημένα
8	Η κλάση TreeSet μας επιτρέπει να παριστάνουμε δυαδικά δέντρα	Λ	Σύνολα. (απλά τα κρατά ταξινομημένα)
9	Με το Reflection API της java μπορούμε να φορτώσουμε μια κλάση αν γνωρίζουμε το όνομα της	Σ	
10	Με το Reflection API της java μπορούμε να πάρουμε πληροφορίες για τις υποκλάσεις μιας κλάσης	Λ	
11	Με το Reflection API της java μπορούμε να πάρουμε πληροφορίες για τις κατασκευάστριες μεθόδους της κλάσης που δέχονται παραμέτρους	Σ	
12	Η χρήση generics επιτρέπει στον compiler να εντοπίσει σφάλματα τα οποία αλλιώς θα τα αντιλαμβανόμασταν κατά τη διάρκεια της εκτέλεσης	Σ	

13	Το παρακάτω πρόγραμμα θα περάσει από τον compiler; Αν όχι γιατί;
	`import java.util.*;`
	`class Person {`
	`String name = null;`
	`public static void main(String a[]){`
	`List<Person> personlist = new ArrayList<Person>();`
	`List<Object> generallist = personlist;`
	`}`
	`}`
		Λ	Η τελευταία εκχώρηση είναι λάθος. List<Person> δεν μπορεί να μετατραπεί σε List<Object>

14	Είναι ο παρακάτω κώδικας σωστός;
	`class MyThread extends Thread {`
	`static final int NUM=1000;`
	`public void run() {`
	`for (int i=0; i<NUM; i++)`
	`System.out.print("bam");`
	`}`
	`}`
		Σ	

Άσκηση 8 (10 μονάδες) // ΑΤΔ

Ακολουθεί η περιγραφή των δημόσιων μεθόδων ενός συνόλου από πολύ γνωστούς ΑΤΔ (Αφαιρετικούς Τύπους Δεδομένων) σε μορφή ψευδοκώδικα. Εντοπίστε τα σχεδιαστικά προβλήματα (παράλειψη λειτουργιών, λανθασμένες ή πλεονάζουσες λειτουργίες) και σχολιάστε τα σύντομα.

		Απάντηση

1
(2μ)
	class Stack {
	Stack(int size) ; // constructs a stack of that size
	void push(Object elem); // pushes elem
	Object pop(); // returns and removes the top element
	Boolean isEmpty(); // checks if stack is empty
	}
		Απουσιάζει μια λειτουργία που να ελέγχει αν η στοίβα είναι πλήρης. Άρα θα έπρεπε να υπήρχε και μια Boolean isFull();
		
		Εναλλακτικά, εάν η εσωτερική υλοποίηση επιτρέπει στοίβες δυναμικού μεγέθους (και για το λόγο αυτό δεν έχει δηλωθεί μια isFull()), τότε θα έπρεπε να απουσιάζει η παράμετρος size από την κατασκευάστρια μέθοδο.

2	class Stack {	Απουσιάζει μια λειτουργία που να

(2μ)	Stack() ; // constructs a stack
	void push(Object elem); // pushes elem
	Object top(); // returns the top element
	Boolean isEmpty(); // checks if stack is empty
	}
		αφαιρεί στοιχεία από τη στοίβα.
		Θα έπρεπε να υπήρχε μια
		void pop(); // removes the top element

3
(6μ)
	class Set {
	// adds elem to set
	void add(Object elem);
	// removes elem from set
	void del(Object elem);
	// returns the number of occurences of elem in set
	int getNumOfOccurences(elem)
	}
		(a) σε ένα σύνολο ένα στοιχείο εμφανίζεται το πολύ μια φορά. Άρα αντί της
		int getNumberOfOccurences(elem)
		Θα έπρεπε να έχουμε μια
		Boolean exist(elem)
		(b) Δεν έχουμε τρόπο να πάρουμε τα στοιχεία του συνόλου. Θα έπρεπε να προσφέρεται ένας τρόπος (π.χ. να δίνει έναν iterator).

Άσκηση 9 (10 μονάδες) // εξαιρέσεις

Έστω ότι θέλετε να δηλώσετε μια κλάση με τον εξής τρόπο:

```
class A {
      Class B = Class.forName("reflection.ClassInfo");
}
```

Αν προσπαθήσετε να μεταγλωττίσετε τον κώδικα ο μεταγλωττιστής θα σας πει ότι κώδικας σας δεν είναι έγκυρος διότι η Class.forname() εγείρει εξαιρέσεις. Τι θα μπορούσατε να κάνετε ώστε να ξεπεράσετε αυτό το πρόβλημα (χωρίς να αλλάξετε τη γραμμή που αρχικοποιεί τη μεταβλητή B);

Λύση
Ένας τρόπος είναι να δηλώσουμε ότι όλες οι κατασκευάστριες μέθοδοι της class A εγείρουν την εξαίρεση `ClassNotFoundException`. Π.χ.:

```
class A {
      Class B = Class.forName("reflection.ClassInfo");

      A() throws ClassNotFoundException { }
}
```

Καλή επιτυχία