Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

ΗΥ-252 – Αντικειμενοστρεφής Προγραμματισμός
Βασίλης Χριστοφίδης

Τελική Εξέταση (3 ώρες)
Ημερομηνία: 25 Ιανουαρίου 2013

Άσκηση 1	Άσκηση 2	Άσκηση 3	Άσκηση 4	Άσκηση 5	Άσκηση 6	Άσκηση 7	Άσκηση 8	Σύνολο
/15	/10	/18	/10	/10	/16	/22	/09	/110

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Άσκηση 1 (18 μονάδες) Στατικοί Γενικοί Τύποι

α) **(5 μονάδες)** Σας δίνεται ο ακόλουθος κώδικας Java. Ποιές τις παρακάτω αναθέσεις (assignments) είναι έγκυρες (δηλ. χωρίς προβλήματα κατά τον έλεγχο των τύπων);

```
class Box<T>{}
class SuperBox<T> extends Box<T>{}
I. Box<Object> b = new Box<String>();
II. Box<String> b = new SuperBox<String>();
III Box<Object> b = new SuperBox<String>();
```

- I, II, III
- μόνο η II
- μόνο η I και III
- μόνο η I

Λύση:
II only

β) **(5 μονάδες)** Ποιές τις παρακάτω αναθέσεις είναι έγκυρες (δηλ. χωρίς προβλήματα κατά τον έλεγχο των τύπων);

i) `Set<String> s = new Set<String>();`
ii) `Set<String> s = new HashSet<String>();`
iii) `HashSet<String> s = new Set<String>();`
iv) `HashSet<String> s = new TreeSet<String>();`

Λύση:
ii) `Set<String> s = new HashSet<String>();`

γ) **(5 μονάδες)** Συμφωνείται με την εκτίμηση ότι οι γενικοί τύποι στην Java καθιστούν έγκαιρη την αναγνώριση λαθών χρήσης τύπων από τον χρόνο εκτέλεσης στον χρόνο μετάφρασης των προγραμμάτων; Δικαιολογήστε σύντομα την απάντησή σας δίνοντας ένα παράδειγμα.

Λύση:
Without using generics, any instance of Object can be placed in a Collection. When the client accesses one of those instances, it can (try to) apply operations that may not apply to it. Generics eliminate this problem from run-time by adding type-checking to the client to make sure that the operations can be applied to instances of the subtype of Objects declared in the generic.

Άσκηση 2 (10 μονάδες) Χειρισμός Εξαιρέσεων

Γράψτε την έξοδο της εκτέλεσης του παρακάτω προγράμματος.

Σημείωση: `IllegalArgumentException` και `NullPointerException` είναι υποκλάσεις της `RuntimeException`.

```
public class TryCatchMystery {
     public static void main (String[] args) {
          try {
               method1();
               method2();
          } catch (IllegalArgumentException e) {
               System.out.println("main IllegalArgumentException");
          } catch (RuntimeException e) {
               System.out.println("main RuntimeException");
          }
     }
     public static void method1() {
          System.out.println("entered method1");
          try {
               method2();
          } catch (IllegalArgumentException e) {
            System.out.println("method1 IllegalArgumentException");
            throw new NullPointerException();
          } catch (NullPointerException e) {
               System.out.println("method1 NullPointerException");
               throw new NullPointerException();
          }
          System.out.println("exited method1");
     }
     public static void method2() {
          System.out.println("entered method2");
          throw new IllegalArgumentException();
     }
}
```

Λύση:

```
entered method1
entered method2
method1 IllegalArgumentException
main RuntimeException
```

Άσκηση 3 (18 μονάδες) Βασική Λειτουργικότητα Αντικειμένων

Σας δίνεται οι ακόλουθες κλάσεις Java.

```
public class OneDPoint {
     double x;
     public boolean equals(Object o) {
          if (o instanceof OneDPoint) {
               OneDPoint other = (OneDPoint) o;
               return x == other.x;
          } else {
               return false;
          }
     }
}
public class TwoDPoint extends OneDPoint {
     double y;
     public boolean equals(Object o) {
          if (o instanceof TwoDPoint) {
               TwoDPoint other = (TwoDPoint) o;
               return x == other.x && y == other.y;
          } else {
               return false;
```

```
        }
    }
}
```

α) **(6 μονάδες)** Ισχύει η συμμετρική ιδιότητα για την σχέση δύο οποιαδήποτε στιγμιότυπων της κλάσης `OneDPoint` που ορίζει η παραπάνω μέθοδος `equals();`Καλείστε να αποδείξετε ότι εάν η λογική έκφραση `p1.equals(p2)` είναι αληθής τότε και η `p2.equals(p1)` είναι αληθής για οποιαδήποτε δύο αντικείμενα `p1` και `p2` τύπου `OneDPoint`. Για την απόδειξη της αντίθετης περίπτωσης αρκεί να βρείτε ένα κατάλληλο αντιπαράδειγμα.

Λύση: No Counterexample
`OneDPoint p1 = new OneDPoint();`
`p1.x = 5;`
`TwoDPoint p2 = new TwoDPoint();`
`p2.x = 5; p2.y = 10;`
`p1.equals(p2)` ➔ `true`
`p2.equals(p1)` ➔ `false`

β) **(6 μονάδες)** Σας δίνεται η παρακάτω τροποποιημένη υλοποίηση της μεθόδου `equals()` στην κλάση `TwoDPoint`.

```
public class TwoDPoint extends OneDPoint {
    protected double y;
    public boolean equals(Object o) {
        if (o instanceof OneDPoint) {
            return o.equals(this);
        } else if (o instanceof TwoDPoint) {
            TwoDPoint other = (TwoDPoint) o;
            return x == other.x && y == other.y;
        } else {
            return false;
        }
    }
}
```

Ισχύει η ανακλαστική ιδιότητα για την σχέση με τον εαυτό του ενός οποιοδήποτε στιγμιότυπου της κλάσης `OneDPoint` που ορίζει η παραπάνω μέθοδο `equals();` Καλείστε να αποδείξετε ότι η λογική έκφραση `p.equals(p)` είναι αληθής για οποιοδήποτε αντικείμενο `p` τύπου `OneDPoint`. Για την απόδειξη της αντίθετης περίπτωσης αρκεί να βρείτε ένα κατάλληλο αντιπαράδειγμα.

Λύση: No Counterexample:
For any `OneDPoint p`, execution of `p.equals(p)` suffers an infinite loop and stack overflow.

γ) **(6 μονάδες)** Σας δίνεται η ακόλουθη τροποποιημένη υλοποίηση της μεθόδου `equals()` στην κλάση `TwoDPoint`.

```
public class TwoDPoint extends OneDPoint {
    protected double y;
    public boolean equals(Object o) {
        if (o instanceof TwoDPoint) {
            TwoDPoint other = (TwoDPoint) o;
            return x == other.x && y == other.y;
        } else if (o instanceof OneDPoint) {
            return o.equals(this);
        } else {
            return false;
        }
    }
}
```

Ισχύει η μεταβατική ιδιότητα μεταξύ δύο οποιαδήποτε στιγμιοτύπων της κλάσης `OneDPoint` που ορίζει η παραπάνω μέθοδο `equals();` Καλείστε να αποδείξετε ότι εάν η λογική έκφραση `p1.equals(p2)` και `p2.equals(p3)` είναι αληθής τότε και η `p1.equals(p3)` είναι αληθής για οποιαδήποτε τρία αντικείμενα p1, p2 και p3 τύπου `OneDPoint`. Για την απόδειξη της αντίθετης περίπτωσης αρκεί να βρείτε ένα κατάλληλο αντιπαράδειγμα.

Λύση: No Counterexample:
TwoDPoint a = new TwoDPoint(); a.x = 5; a.y = 10;
OneDPoint b = new OneDPoint(); b.x = 5;
TwoDPoint c = new TwoDPoint(); c.x = 5; c.y = 20;
p1.equals(p2) → true
p2.equals(p3) → true
p1.equals(p3) → false

Άσκηση 4 (10 μονάδες) Βασική Λειτουργικότητα Αντικειμένων

Η κλάση `Toy` αναπαριστά διαφορετικά παιχνίδια που πωλεί ένα κατάστημα.

```
public class Toy {
     private String name;       // Όνομα παιχνιδιού, πχ "Furby"
     private String description; // περιγραφή, πχ "annoying"
     private int price;         // τιμή σε euros
     private int sku;           // απόθεμα (αποθήκη)
```

Θα πρέπει να παρέχουμε τις μεθόδους `equals()` και `hashCode()` για την κλάση και θεωρούμε τις ακόλουθες δυνατές υλοποιήσεις.

```
     public boolean equals(Object other) { // equals version 1
          if (!(other instanceof Toy)) return false;
          Toy t = (Toy) other;
          return this.sku == t.sku;
     }
     public boolean equals(Object other) { // equals version 2
          if (!(other instanceof Toy)) return false;
          Toy t = (Toy) other;
          return this.name.equals(t.name) && this.sku == t.sku;
     }
     public boolean equals(Object other) { // equals version 3
          if (!(other instanceof Toy)) return false;
               Toy t = (Toy) other;
          return this.name.equals(t.name) &&
                 this.description.equals(t.description);
     }
     public int hashCode() { // hashcode version 1
          return sku;
     }
     public int hashCode() { // hashcode version 2
          return name.hashCode()+sku;
     }
     public int hashCode() { // hashcode version 3
          return name.hashCode();
     }
}
```

Συμπληρώστε τον ακόλουθο πίνακα τοποθετώντας ένα X σε κάθε κενό όπου οι αντίστοιχες μέθοδοι `hashCode()` και `equals()` που δίνονται προηγουμένως έχουν συμβατές υλοποιήσεις– δηλαδή όποτε ισχύει ότι `a.equals(b)`, τότε η `a.hashCode() =b.hashCode()`.

	equals version 1	**equals version 2**	**equals version 3**
hashCode version 1			
hashCode version 2			
hashCode version 3			

Λύση:

	equals version 1	equals version 2	equals version 3
hashCode version 1	X	X	
hashCode version 2		X	
hashCode version 3		X	X

Άσκηση 5 (10 μονάδες) Προγραμματισμός Βασισμένος σε Συμβόλαια

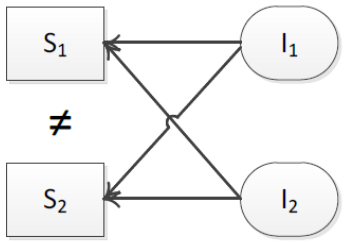
Θυμηθείτε την διαφορά μεταξύ του καθορισμού (**S**pecification) και της υλοποίησης (**I**mplementation) ενός Αφαιρετικού Τύπου Δεδομένων (ΑΔΤ): ο καθορισμός ενός ΑΔΤ αποτελεί τον λογικό ορισμό της συμπεριφοράς των αφαιρετικών τιμών του ενώ η υλοποίησή του βασίζεται σε συγκεκριμένες δομές δεδομένων και αλγορίθμους που επιτρέπουν έναν προγραμματιστικό χειρισμό του που ικανοποιεί (satisfies) τον καθορισμό. Είναι δυνατόν να ικανοποιούνται όλες οι παρακάτω συνθήκες συγχρόνως; .

`S1 is satisfied by I1 and S2 is satisfied by I1 and S1 is satisfied by I2 and S2 is satisfied by I2 and S1 != S2`

Δικαιολογήστε σύντομα την απάντησή σας.

Λύση:

This is basically setting up the following situation:



It is possible: the simplest abstraction description is that this holds whenever S1 is stronger than S2 or vice versa. There are a number of concrete examples. One given a couple of times was (roughly) I1 and I2 as `LinkedList` and `ArrayList`, respectively, and S1 and S2 being `List` and `Collection`, respectively.

Common Errors:

Some students gave answers of the form: "Because of x, y or z, nothing stops this situation from arising." This is different from justifying why it can happen.

Άσκηση 6 (16 μονάδες) Επαναλήπτες Συλλογών Αντικειμένων

Συμπληρώστε την υλοποίηση της κλάσης Μισθοδοσία (`Payroll`) έτσι ώστε ένας κώδικας πελάτης της μεθόδου `getLuckyIterator()` να παίρνει έναν σαρωτή (Iterator) που διασχίζει την συλλογή υπαλλήλων (`Employee`) που περιέχει η μισθοδοσία και να επιστρέφει υπαλλήλους με μισθό μεγαλύτερο ή ίσο των 50000. Θυμηθείτε ότι η διεπαφή `Iterator` έχει τρείς μεθόδους από τις οποίες μόνο οι ακόλουθες δύο μας ενδιαφέρουν στην άσκηση:

```
1. public boolean hasNext()
2. public Object next()
```

Στην συνέχεια σας δίνεται ένα παράδειγμα κώδικα πελάτη (`PayrollTester`) που χρησιμοποιεί την λειτουργικότητα του σαρωτή που πρέπει να υλοποιηθεί στην κλάση

Payroll. Η έξοδος που τυπώνεται από την εκτέλεση της μεθόδου main() της PayrollTester είναι:

```
Dimitris, 50000
Kostas, 80000
```

```
public class PayrollTester {
     public static void main(String[] args) {
          Payroll payroll = new Payroll();
          payroll.employees.add(new Employee("Dimitris", 50000,0));
          payroll.employees.add(new Employee("Vassilis", 49000,0));
          payroll.employees.add(new Employee("Kostas", 80000,0));
          payroll.employees.add(new Employee("Giorgos", 40000,0));
          Iterator it = payroll.getLuckyIterator();
          while (it.hasNext())
               System.out.println(it.next());
     }
}
public class Employee {
     public String name;
     public double salary;
     public Employee(String initName, double initSalary) {
          name = initName;
          salary = initSalary;
     }
     public String toString() { return name + ", $" + salary; }
}
public class Payroll {
     public Vector<Employee> employees = new Vector<Employee>();
     public Payroll() {}
     public Iterator getLuckyIterator() {
```

Λύση:

```
          return new LuckyIterator(); \\1point
     }
class LuckyIterator implements Iterator {
```

Λύση:

```
     int index; // 1point
     public LuckyIterator() {
          index = -1;
          findNext();
     } //2points
     private void findNext() {
          index++;
          while ((index < employees.size()) &&
                 (employees.get(index).getSalary() < 50000)) {
                 index++;
          } //4 points
          if (index == employees.size())
               index = -1;
     } //2 points
     public boolean hasNext() {
               return index != -1;
     } //2 points
     public Object next() {
               Object data = employees.get(index);
               findNext();
               return data;
     } //4points
     public void remove(){}
}
```

Άσκηση 7 (22 μονάδες) Χειρισμός Πλαισίου Συλλογών Αντικειμένων

Μας ενδιαφέρει η ολοκλήρωση μιας πρώτης υλοποίησης (Implementation A) της κλάσης `IntStringMap` η οποία αναπαριστά έναν πίνακα αντιστοίχισης (Map) με κλειδί (key) ένα ακέραιο (`int`) και τιμή (value) μια συμβολοσειρά (`String`). Σε αυτή την υλοποίηση χρησιμοποιούμε έναν πίνακα συμβολοσειρών με όνομα `myValues` όπου το περιεχόμενο ενός κελιού είναι η τιμή ενώ η θέση (index) της είναι το κλειδί κάθε ζεύγους του πίνακα αντιστοίχισης. Για παράδειγμα το ζεύγος (4, "Hello") θα αποθηκευτεί σαν `myValues[4] = "Hello";`
Εάν κάποιο κλειδί δεν περιέχεται στον πίνακα αντιστοίχισης (Map), ο πίνακας στην αντίστοιχη θέση θα περιέχει την κενή συμβολοσειρά "". Μπορείτε να ελέγξετε εάν κάποια συμβολοσειρά `myString` ισούται με την κενή συμβολοσειρά χρησιμοποιώντας την κλήση `myString.equals("");`
Για λόγους απλότητας υποθέστε ότι ανά πάσα στιγμή ένα κλειδί που πρέπει να χειριστείτε στον πίνακα αντιστοίχισης είναι ένας θετικός ακέραιος η τιμή του οποίου είναι μικρότερη από την χωρητικότητα (capacity) του πίνακα. Υποθέστε ακόμα ότι η αρχική χωρητικότητα ενός πίνακα είναι μεγαλύτερη του μηδενός.
α) **(10 μονάδες)** Συμπληρώστε την υλοποίηση της κλάσης `IntStringMap`. Μην ξεχάσετε να ενημερώνετε το λογικό μέγεθος του πίνακα (`size`) όποτε αυτό είναι απαραίτητο.

```
public class IntStringMap {
     private int capacity;
     private int size;
     private String[] myValues;
     // Creates a Map that can hold up to capacity number of pairs
     public IntStringMap(int capacity) {
          myValues = new String[capacity];
          this.capacity = capacity;
          for (int i = 0; i < capacity; i++) {
               myValues[i] = ""; // initialize to ""
          }
          size = 0; // empty to begin with
     }
//Returns true if this map contains a mapping for the specified key
     public boolean containsKey(int key) {
```

Λύση:

```
          return !myValues[key].equals(""); \\2points
```

```
     }
    //Returns true if at least one key-value pair has the value val
     public boolean containsValue(String val) {
```

Λύση:

```
          for (int i = 0; i < capacity; i++) {
               if (myValues[i].equals(val))
                    return true;
               }
          }
          return false; \\5points
```

```
     }
     // Returns the value associated with this key, otherwise ""
     public String get(int key) {
          return myValues[key];
     }
     // Puts this key-value pair in the Map
     public void put(int key, String value) { ... }
```

```
     // Removes the mapping for this key from this map
     void remove(int key) {
```

Λύση:

```
          if (containsKey(key)) {
               myValues[key] = "";
               size--;
               \\3points
          }

     }
}
```

β) **(12 μονάδες)** Θεωρήστε μια εναλλακτική υλοποίηση (Implementation B) της κλάσης `IntStringMap` η οποία επιτρέπει τα κλειδιά να είναι οποιοσδήποτε ακέραιος (δηλ ενδεχομένως αρνητικός, μεγαλύτερος της χωρητικότητας του πίνακα, κλπ). Σε αυτή την περίπτωση αντί για έναν πίνακα συμβολοσειρών χρησιμοποιούμε ένα πίνακα με αντικείμενα `IntStringMapItems` που περιγράφονται στην συνέχεια:

```
class IntStringMapItem {
     public int key;
     public String value;
     IntStringMapItem (int k, String v) {
          key = k;
          value = v;
     }
}
```

Η κλάση `IntStringMap` θα μπορούσε να υλοποιηθεί ως εξής:

```
public class IntStringMap {
     private int capacity;
     private int size;
     private IntStringMapItem[] myPairs;
     // Creates a Map that can hold up to capacity number of pairs
     public IntStringMap(int capacity) {
          myPairs = new IntStringMapItem[capacity];
          this.capacity = capacity;
          for (int i = 0; i < capacity; i++) {
               myPairs [i] = null; // initialize to null
          }
          size = 0;
     }
```

Οι υπόλοιπες μέθοδοι μπορούν να υλοποιηθούν παρόμοια ενώ η συμπεριφορά τους θα είναι η αναμενόμενη. Για κάθε μία από τις παρακάτω μεθόδους γράψτε εάν ο χρόνος εκτέλεσής τους αναμένεται να είναι:
(i) Σταθερός (constant)
(ii) Γραμμικός (linear) ως προς το λογικό μέγεθος του Map (δηλ του πλήθους των ζευγών που περιέχει)
(iii) Γραμμικός (linear) ως προς την χωρητικότητα του Map (δηλ του μέγιστου πλήθους ζευγών που μπορεί να περιέχει)
(iv) Λογαριθμικός (logarithmic) ως προς το λογικό μέγεθος του Map

	Implementation A	**Implementation B**
containsKey		
containsValue		
remove		

Λύση:
Part of earning the extra credit for this assignment is understanding approximately how you would implement the rest of this class without actually doing so.

	Implementation A	Implementation B
containsKey	i	ii
containsValue	iii	ii
remove	i	ii

Άσκηση 8 (9 μονάδες) Θεωρία Αντικειμενοστρεφούς Προγραμματισμού

ά) **(3 μονάδες)** Ποια είναι η πιο σημαντική ομοιότητα μεταξύ μιας διεπαφής (interface) και μιας αφαιρετικής κλάσης (abstract class);

Λύση:
Neither one can be instantiated.

β) **(3 μονάδες)** Δώστε ένα παράδειγμα στο οποίο προτιμάτε να χρησιμοποιήσετε μια διεπαφή από μια αφαιρετική κλάση.

Λύση:
When specifying one of multiple supertype behaviors, since a class can implement many interfaces.

γ) **(3 μονάδες)** Δώστε ένα παράδειγμα στο οποίο προτιμάτε να χρησιμοποιήσετε μια αφαιρετική κλάση από μια διεπαφή.

Λύση:
When giving default behavior for a set of related data abstractions, since an interface cannot provide a method implementation.