Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

ΗΥ-252 – Αντικειμενοστρεφής Προγραμματισμός
Γιάννης Τζίτζικας

Πρόοδος (3 ώρες)
Ημερομηνία: 2 Δεκεμβρίου 2007
Σύνολο μονάδων: 10.5 (δηλαδή μισή μονάδα bonus)

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Θέμα 1 (10 μονάδες) // Θεωρία Αντικειμενοστρεφούς Σχεδίασης

α) (**5 μονάδες**)
Ποια είναι τα οφέλη από τη σχεδίαση μέσω συμβολαίου;

β) (**5 μονάδες**)
Ως γνωστόν, η κατασκευάστρια μέθοδος μιας υποκλάσης μπορεί καλέσει την κατασκευάστρια μέθοδο της υπερκλάσης της (χρησιμοποιώντας την εντολή `super` στη πρώτη γραμμή του κώδικα της). Αυτή η δυνατότητα είναι τελικά απαραίτητη ή θα μπορούσαμε να αρχικοποιήσουμε τις μεταβλητές στιγμιοτύπων της υπερκλάσης και αλλιώς; Δικαιολογήστε την απάντηση σας.

Ενδεικτική Λύση:
(α)
1. Καταμερισμός αρμοδιοτήτων π.χ. των ελέγχων που πρέπει να γίνουν (ο πελάτης έχει την ευθύνη των preconditions, άρα αποφεύγομαι τις περιπτώσεις όπου κανείς δεν κάνει ελέγχους και όπου αμφότεροι κάνουν ελέγχους
2. οδηγεί σε καλύτερα τεκμηριωμένο κώδικα
3. βοηθάει τον έλεγχο και την αποσφαλμάτωση (η επιθυμητή συμπεριφορά είναι τεκμηριωμένη)

(β)
If all the instance variables were public, protected, or package accessible in the same package as the subclass, then the above claim would be correct. However, private instance variables cannot be accessed from another class. Therefore the subclass would be unable to initialize them without a super call.

Άσκηση 2 (20 μονάδες) // βασικά
Ποιες από τις παρακάτω γραμμές θα μεταγλωττιστούν χωρίς κανένα πρόβλημα;

		Κώδικας	Απάντηση και Δικαιολόγηση (αν λάθος)
2μ	1)	`float f=1.3;`	
2μ	2)	`char c="a";`	
2μ	3)	`byte b=257;`	
2μ	4)	`boolean b=null;`	

2μ	5)	int i=10;	
2μ	6)	import java.awt.*; package Mypackage; class Myclass {}	
2μ	7)	package MyPackage; import java.awt.*; class MyClass{}	
	8)	package MyPackage; import java.awt.*; class MyClass{}	<<ιδιο με 7>>

3μ	9)	class Base {}
		class Sub extends Base {}
		public class Tester{
		public static void main(String a[]){
		Base b=new Base();
		Sub s=(Sub) b;
		}
		}	

3μ	10)	class Base {}
		class Sub extends Base {}
		public class Tester{
		public static void main(String a[]){
		Base b=new Base();
		Sub s= b;
		}
		}	

ΛΥΣΗ

We will have no problem with 5, 7,8 and 9.

1) float f=1.3; Will not compile because the default type of a number with a floating point component is a double. This would compile with a cast as in float f=(float) 1.3

2) char c="a"; Will not compile because a char (16 bit unsigned integer) must be defined with single quotes. This would compile if it were in the form char c='a';

3) byte b=257; Will not compile because a byte is eight bits. Take of one bit for the sign component you can define numbers between -128 to +127

4) a boolean value can either be true or false, null is not allowed

6) will not compile because any package declaration must come before any other code

9) Will compile but will throw runtime error

10) Will not compile

(on 9 and 10):

Without the cast to sub you would get a compile time error. The cast tells the compiler that you really mean to do this and the actual type of b does not get resolved until runtime. Casting down the object hierarchy is a problem, as the compiler cannot be sure what has been implemented in descendent classes. Casting up is not a problem because sub classes will have the features of the base classes.

Άσκηση 3 (15 μονάδες) **// εμβέλεια, αφηρημένες κλάσεις και μέθοδοι**

Είναι παρακάτω κώδικας σωστός; Αν όχι βρείτε τα λάθη και δικαιολογείστε σύντομα.

```
abstract final class  aA {
      abstract public void  setValue(int v);
      abstract private void  setValue();
      public  int getValue();
```

```
}

class A extends aA {
      int val;
      public void setValue(int v) { val=v;}
      public void setValue()    {val=18};
      public int getValue() {return val;}
}
```

Λύση
Λάθη:
(1) (5 μονάδες) Δεν έχει νόημα να δηλωθεί μια αφηρημένη (abstract) κλάση ως τελική (final) καθώς τότε δεν μπορούν να δημιουργηθούν υποκλάσεις από αυτήν.
(2) (5 μονάδες) Η μέθοδος getValue της aA έπρεπε είτε να έχει δηλωθεί ως abstract (αφού δεν έχει σώμα), ή να είναι υλοποιημένη μέσα στην aA (έτσι ώστε να μπορεί να χρησιμοποιηθεί κατάλληλα από τις υποκλάσεις της).
(3)(5 μονάδες) Η μέθοδος abstract private void setValue δεν έχει νόημα (δεν έπρεπε να είναι private). Αυτό ισχύει γιατί μία private μέθοδος δεν μπορεί να την προσπελάσει κανένας άλλος παρά μόνο η κλάση στην οποία δηλώνεται.

Παρατηρήσεις:

- Το παρακάτω συντακτικό λάθος δεν λήφθηκε υπ'όψιν στην βαθμολόγηση καθώς πρόκειται για τυπογραφικό λάθος : public void setValue() {val=18};

Άσκηση 4 (20 μονάδες) // polymorphism and hiding
Ποια είναι η έξοδος του παρακάτω κώδικα και γιατί; (δικαιολογείστε σύντομα)

```
class Small {
      public int value;
      public Small()  { value = 10; }
      public int getValue() { return value; }
}
class Big extends Small {
      public int value;
      public Big() { value = 40; }
      public int getValue() { return value-10; }
}
public class Tester {
      public static void main (String args[]) {

      Small small     =  new Small();
      Small smallBig  =  new Big();
      Big   big       =  new Big();

       System.out.println(small.getValue());
       System.out.println(smallBig.getValue());
       System.out.println(big.getValue());
       System.out.println(small.value);
       System.out.println(smallBig.value);
       System.out.println(big.value);
       small = (Small) big;
       System.out.println(small.getValue());
       System.out.println(small.value);
       big = (Big) small;
       System.out.println(big.getValue());
       System.out.println(big.value);

  }
}
```

Λύση: The `println()` statements print the following numbers:

10 (trivial)
30 (uses Big's getValue() that overrides Small's)
30 (trivial)
10 (trivial)
10 (sos)
40 (trivial)
30 (uses Big's getValue that overrides Small's)
10 (sos)
30 (uses Big' getValue that overrides Small's)
40 (sos)

Βαθμολόγηση: 2 μονάδες για κάθε σωστή γραμμή.

Άσκηση 5 (20 μονάδες) // πολυμορφισμός

Τι θWill not compile because the default type of a number with a floating point component is a doubleα εκτυπώσει το παρακάτω πρόγραμμα; Κάποιες εντολές μπορεί να είναι λανθασμένες. Σε αυτήν την περίπτωση εξηγήστε σύντομα το λάθος.

```
class Animal {
  public void sayHi() {smile(); System.out.println("Hi!"); }
  public void smile() {;}
  public void sayHiAnimal() {sayHi();}
}

class Dog extends Animal {
  public void sayHi() { System.out.print("Guv."); super.sayHi();  }
  public void smile() { System.out.print("Gavvv."); super.smile();  }
  public void sayHiDog() { sayHi();  }
}

public class AnimalTest {
    public static void main (String args[]) {
           Animal an1 = new Animal();
           Animal an2 = new Dog();
           Dog dog = new Dog();

           an1.sayHiAnimal();
           an2.sayHiAnimal();
           an1.sayHi();
           an2.smile(); System.out.println();

           dog.sayHi();
           dog.sayHiAnimal();
           dog.sayHiDog();

           an1.sayHiDog();
           an2.sayHiDog();
     }
 }
```

ΛΥΣΗ
Μόνο οι δυο τελευταίες είναι λάθος.
Οι υπόλοιπες εντολές θα τυπώσουν τα εξής

```
Hi!
Guv.Gavvv.Hi!
Hi!
Gavvv.
Guv.Gavvv.Hi!
Guv.Gavvv.Hi!
Guv.Gavvv.Hi!
```

Η προτελευταία εντολή είναι λάθος διότι η sayHiDog δεν είναι μέθοδος του Animal. Η τελευταία εντολή είναι λάθος διότι η sayHiDog δεν είναι μεθοδος του Animal, και η an2 είναι μεταβλητή τύπου Animal αν και κρατά ένα αντικείμενο τύπου Dog.

Βαθμολόγηση: 2μ για κάθε σωστή γραμμή εξόδου.
4 μονάδες αν εντοπίσουν ότι οι 2 τελευταίες γραμμές είναι λάθος

Θέμα 6 (20 μονάδες) // ADT

(α) 10 μονάδες

Καθορίστε την προδιαγραφή για ένα ΑΤΔ με όνομα SortedSet για τη διαχείριση διατεταγμένων συνόλων. Καθορίστε την προδιαγραφή του, δηλαδή δώστε τις υπογραφές (signatures) των προβλεπόμενων λειτουργιών του, το είδος τους ανάλογα με την επίδραση που έχουν στην κατάσταση των αντικειμένων (constructors, observers, accessors, transformers), καθώς και τις εκ των προτέρων, τις εκ των υστέρων και τις αμετάβλητες συνθήκες (preconditions, postconditions, invariants).

ΕΝΔΕΙΚΤΙΚΗ ΛΥΣΗ

[observers]
IsEmpty: SSet -> Boolean
Pre: none
Post: True if empty, False otherwise
Exceptions: none
Size: SSet -> Int
Pre: none
Post: returns the number of elements
Exceptions: none
[accessors]
Exist: SSet X Elem ->Boolean
Pre: none
Post: returns True if elem exists, otherwise false
Exceptions: none
ElementAt: SSet x Int -> Elem
Pre: 0 < int < Size()
Post: Returns the element at the i-th position
Exceptions: if preconditions are not specified, produces outofbounds
[transformers]
Add: SSet X Elem -> Sset
Pre: none
Post: if Exist(elem) then do nothing,
Else it adds the element to the right place
Exceptions: none
Del: SSet X Elem -> Sset
Pre: IsEmpty(sset)=false
Post: If Exist(elem)=True it deletes it
Exceptions: none
Constructors
Init: ->SSet
Pre: None
Post: It creates an empty set

INVARIANTS
If ElementAt(i) = e and ElementAt(j)=e' and i<j then e <e ' // the elements should be sorted

EQUATIONAL AXIOMS (some indicative)

isEmpty(sset) = (Size(sset) == 0) // regarding size and additions
If Exist(sset,e) then Size(sset) = Size(Add(sset,e)) // regarding additions and size
If not Exist(sset,e) then Size(sset) = Size(Add(sset,e))-1 // regarding additions and size
If Exist(sset,e) then Size(sset) = Size(Del(sset,e))-1 // regarding deletions
If not Exist(sset,e) then Size(sset) = Size(Del(sset,e)) // regarding deletions

Βαθμολόγηση:
Οι εξαιρέσεις δεν είναι υποχρεωτικές (είναι εκτός ύλης προόδου)
Θα ήταν καλό κάποιος να αναφέρει προϋποθέσεις σχετικά με τη δυνατότητα σύγκρισης των αντικειμένων που εισέρχονται στο SSet.

(β) 10 μονάδες
Δώστε μια υλοποίηση του παραπάνω τύπου χρησιμοποιώντας την κλάση Vector.
Στην υλοποίηση σας εκμεταλλευτείτε (για να βελτιώσετε την επίδοση) το γεγονός ότι τα στοιχεία ενός SortedSet είναι πάντα ταξινομημένα. Επίσης επικαλύψτε την toString() ώστε να δίδει μια συμβολοσειρά αποτελούμενη από όλα τα στοιχεία του πίνακα.

ΕΝΔΕΙΚΤΙΚΗ ΛΥΣΗ

```
class SSet {
      private Vector v;
      public SSet(){v = new Vector();  }
      public boolean isEmpty() {return v.isEmpty();     }
      public int size() {   return v.size(); }
      public boolean Exist(Object o) {return v.contains(o);} //could be
better: i.e. binary search
      public void add(Object e) {
            if (!Exist(e)) {
                  v.add(e);
                  Collections.sort(v); // could be better
            }
      }
      public void delete(Object e) {
            if (Exist(e)) {
                  v.remove(e);
            }
      }
      public Sting toString() {
      StringBuffer sb = new StringBuffer();
            for (int i=0; i < v.size(); i++){
               sb.append(v.elementAt(i)+ "   ");
           }
           return sb.toString();
}
```

Βαθμολόγηση:
* Να έχουν δώσει υλοποίηση που σέβεται τα συμβόλαια των λειτουργιών που όρισαν στο σκέλος (α).
* Να έχουν δώσει την binary search στην Exist(o) ή απλά να έχουν κάνει σχετική νύξη
* Να έχουν βάλει τουλάχιστον ένα σχόλιο ότι τα στοιχεία που τοποθετούνται στο sset πρέπει να υλοποιούν το interface comparable (ή να έχουν αντιμετωπίσει αλλιώς το ζήτημα αυτό)