Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

ΗΥ-252 – Αντικειμενοστρεφής Προγραμματισμός
Βασίλης Χριστοφίδης

Πρόοδος (3 ώρες)
Ημερομηνία: 19 Νοεμβρίου 2005

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Άσκηση 1 (20 μονάδες)

a) (**8 μονάδες**) Η μετάθεση (shuffle) δύο συμβολοσειρών είναι η σειρά που αποτελείται από τον πρώτο χαρακτήρα της πρώτης συμβολοσειράς, τον πρώτο χαρακτήρα της δεύτερης συμβολοσειράς, το δεύτερο χαρακτήρα της πρώτης συμβολοσειράς, το δεύτερο χαρακτήρα της δεύτερης συμβολοσειράς, κοκ. Όταν η σάρωση μιας από τις δύο συμβολοσειρές τερματίζει τότε το υπόλοιπο τμήμα της άλλης συμβολοσειράς τελειώνει τη μετάθεση. Για παράδειγμα, η μετάθεση του "DOG" και της "cat" είναι " DcOaGt". Η μετάθεση της "cat" και των "BUFFALO" είναι "cBaUtFFALO". Η μετάθεση της "cat" και της "cat" είναι " ccaatt". Η μετάθεση της κενής συμβολοσειράς "" και της "cat" είναι "cat". Γράψτε μια αναδρομική μέθοδο **shuffle**() που παίρνει σαν παράμετρο τις δύο συμβολοσειρές και επιστρέφει τη μετάθεσή τους.

```
public static String shuffle(String str1, String str2) {
```

Λύση:

```
if (str1.length() == 0)
    return str2;
else if (str2.length == 0)
      return str1;
     else
      return str1.substring(0,1) + str2.substring(0,1) +
           shuffle(str1.substring(1), str2.substring(1));
}
```

b) Θεωρήστε την ακόλουθη λανθασμένη εκδοχή της μεθόδου **containsOneMoreThan**() στην κλάση **StringToCheck**.

```
public class StringToCheck {
  protected String myString;
  // Constructor taking a string as parameter
  public StringToCheck (String s) {
       myString = s;
  }
  // pre (REQUIRES): s is not null.
  // post (EFFECT): returns true when myString is the result of inserting
  // exactly one character into s; returns false otherwise.
  public boolean containsOneMoreThan (String s) {
     if (s.length( )==0) {
      return true;
   } else if (myString.length( )==0) {
      return false;
   } else {
      StringToCheck remainder= new StringToCheck (myString.substring(1));
      if (myString.charAt(0) == s.charAt(0)) {
         return remainder.containsOneMoreThan (s.substring(1));
      } else {
         return remainder.containsOneMoreThan (s);
      }
   }
}
```

(i) (**4 μονάδες**) Αποτυγχάνει στην εκτέλεσή της η παραπάνω μέθοδος ακόμα και όταν το όρισμά της ικανοποιεί τις προ-συνθήκες? Δώστε μια σύντομη εξήγηση.

Λύση:
The answer we normally expect is "no", since the `length` comparisons guarded the subsequent uses of `charAt` and `substring`. However, the method precondition did not rule out a value of `null` for `myString`. Under this light, we will award 2 points for the answer "yes", provided that your explanation mentions the possibility of `myString` containing `null` (Whether you get 2 or 4 points depends on the rest of your answer).
A common error was to claim that a crash would result from `myString` or `s` containing exactly one character. Applying `substring(1)` to a one-character string gives the empty string and produces no error. The erroneous claim may have resulted from confusing the empty string with a null String reference.

(ii) (**8 μονάδες**) Περιγράψτε όλα τα ζευγάρια των συμβολοσειρών `myString` και `s` για τα οποία η μέθοδος `containsOneMoreThan()` θα πρέπει να επιστρέφει ψευδή απάντηση, αλλά ωστόσο η μέθοδος αποτυγχάνει. Για μια μερική απάντηση στο ερώτημα, δώστε ένα τουλάχιστο ζευγάρι των συμβολοσειρών για το οποίο η μέθοδος `containsOneMoreThan()` εμφανίζει αυτή την λανθασμένη συμπεριφορά.

Λύση:
Examining the base cases, method `containsOneMoreThan()` returns false given any empty string `myString`. If `myString` is initially empty, that's the correct result, since an empty string can't possibly be the result of inserting a single character into anything. We thus must examine the recursive cases to see if somehow `myString` can run out at the same time as or before s. There are two possibilities: either the first characters match, so the remainders of the two strings are compared or they don't match, so the rest of `myString` is compared to s. If both remainders are compared, `myString`'s length relative to `s`'s stays the same. Thus we consider the simplest alternative, namely the case where `myString` shrinks but s doesn't, yet `myString` is still the result of inserting a single character into s. This happens for the strings "ab" and "b". More generally, it happens for any `myString` that results from adding a single character in a position other than the end of s. Here are some examples:

```
myString    s          explanation
xabc        abc        returns false, should return true
axbc        abc        returns false, should return true
abxc        abc        returns false, should return true
abcx        abc        returns true, correctly
abcc        abc        returns true, correctly
```

This question worth 8 points, was graded as follows. You earned 2 point for a correct example, provided that you also didn't provide any incorrect examples. You earned 6 points for specifying an infinite set of strings that was a subset of the correct set; saying all pairs `myString` and `s` where `myString` is the result of appending a character to s—exactly the opposite of what we intended—earned 1 points out of 8.

Άσκηση 2 (30 μονάδες)

Σας δίνονται οι παρακάτω κλάσεις Java **Room** και **Player**.

Room.java

```
/**
 * A room in a maze. A Room has a description, and references to
   adjacent rooms in the four cardinal directions.
 */
public class Room {
 private Room north, east, south, west;
 private String description;
 // Create a room with the given description.
  public Room(String description) {
    this.description = description;
 }
```

```
 //Get the room adjacent to the North.
 public Room getNorth() { return north; }
 // Get the room adjacent to the East.
 public Room getEast() { return east; }
 //Get the room adjacent to the South.
 public Room getSouth() { return south; }
 //Get the room adjacent to the West.
 public Room getWest() { return west; }
 //Set which room is adjacent to the North.
 public void setNorth(Room room) { north = room; }
 //Set which room is adjacent to the East.
 public void setEast(Room room) { east = room; }
 //Set which room is adjacent to the South.
 public void setSouth(Room room) { south = room; }
 //Set which room is adjacent to the West.
 public void setWest(Room room) { west = room; }
 //Get the room's description.
 public String getDescription() { return description; }
}
```

Player.java

```
/**
 * A player in a maze. A Player has an immutable name and a mutable
   location, a Room. The player can move from its current location
   in the four cardinal directions.
 */
public class Player {
 private String name;
 private Room location, oldLocation;
 //Create a new Player with given name and initial location.
 public Player(String name, Room location) {
   this.name = name;
   this.location = oldLocation = location;
 }
 // Get the player's name.
 public String getName() { return name; }
 //Get the description of the current location
 public String where() {
   return location.getDescription();
 }
 //Go North from the current location.
 public void goNorth() {
   oldLocation = location;
   location = location.getNorth();
 }
 //Go East from the current location.
 public void goEast() {
   oldLocation = location;
   location = location.getEast();
 }
 //Go South from the current location.
 public void goSouth() {
   oldLocation = location;
   location = location.getSouth();
}
 //Go West from the current location.
 public void goWest() {
   oldLocation = location;
   location = location.getWest();
 }
 //Go back to the last location the player was in before the current
 //location. If the player hasn't moved yet, the player's location does
 //not change.
 public void goBack() {
   Room temp = location;
   location = oldLocation;
   oldLocation = temp;
 }
}
```

(a) (**8 μονάδες**) Υποθέστε ότι οι κλάσεις **Room** και **Player** έχουν μεταγλωττιστεί, και εκτελούμε στην συνέχεια τις παρακάτω εντολές από κάποιον κώδικα πελάτη:

```
Room r1 = new Room("here");
Room r2 = new Room("there");
r1.setNorth(r1);
r1.setSouth(r2);
r2.setNorth(r1);
```

Για κάθε μία από τις παρακάτω εντολές διευκρινίστε εάν θα εκτελεστούν σωστά από τον διερμηνέα της Java ή θα εμφανιστούν λάθη. Στην πρώτη περίπτωση δώστε επίσης αυτό που τυπώνεται στην στάνταρ έξοδο του προγράμματος.

```
(a) Player a = new Player("alice", r1);
(b) a.goNorth();
(c) a.where();
(d) a.goWest();
(e) a.where();
(f) a.goBack();
(g) a.where();
(h) Player b = new Player("bob");
```

Λύση:
```
(a) Player a = new Player("alice", r1); OK
(b) a.goNorth(); OK
(c) a.where(); ”here”
(d) a.goWest(); OK
(e) a.where(); error
(f) a.goBack(); OK
(g) a.where(); ”here”
(h) Player b = new Player("bob"); error
```
Grading Guideline: Credit is given only for correct answer. If answer to part d is ”error”, part e should be ”here” as the correct answer.

(b) (**4 μονάδες**) Η τρέχουσα εκδοχή των μεθόδων **goNorth**(), **goEast**(), . . . στην κλάση **Player** δεν χειρίζεται καλά την περίπτωση όπου το δωμάτιο (Room) στην αντίστοιχη κατεύθυνση είναι null. Δώστε μια υλοποίηση της μεθόδου **goNorth**() η οποία χειρίζεται σωστά την περίπτωση των null.

```
public void goNorth() {
```

Λύση:
```
  if (location.getNorth() != null) {
      oldLocation = location;
      location = location.getNorth(); }
```
Grading Guideline:
+2 points for checking if getNorth() returns null
+1 for updating location correctly
+1 for updating oldLocation in correct place (after checking for null)

```
}
```

(c) (**7 μονάδες**) Υλοποιήστε την μέθοδο runNorth() η οποία ζητά από έναν παίχτη να μετακινηθεί βόρεια (North) για το πολύ maxSteps βήματα, όπου κάθε βήμα αντιστοιχεί με μια μετακίνηση του παίχτη κατά ένα δωμάτιο στην βορινή κατεύθυνση. Εάν ο παίχτης δεν μπορεί να μετακινηθεί περισσότερο είτε γιατί δεν υπάρχει διαθέσιμο δωμάτιο προς τη βορινή κατεύθυνση (null), είτε γιατί έχουν ήδη διανυθεί maxSteps βήματα, η μέθοδος τερματίζει. Η μέθοδος επιστρέφει τον συνολικό αριθμό των βημάτων που έχουν διανυθεί.

```
public int runNorth(int maxSteps) {
```

Λύση:
```
  int steps = 0;
  while (location.getNorth() != null && steps < maxSteps) {
        oldLocation = location;
```

```
            location = location.getNorth();
            steps++;
    }
    return steps;
```

Grading Guideline:
+2 points for each condition check (location.getNorth() and steps).
+1 points for initializing steps or loop variable.
+1 points for updating location correctly. Only 1 point is given if oldLocation is not updated.
+1 points for incrementing number of steps.
+1 point for returning correct value if getNorth() is never null.
+1 point for returning correct value in the case of getNorth() is null.
-1 for having small syntax errors

}

(d) Υποθέστε ότι κάθε δωμάτιο `r` χαρακτηρίζεται από ένα επίπεδο ενέργειας, στο οποίο έχετε πρόσβαση μέσω της μεθόδου `r.getEnergy()`.Θέλουμε να τροποποιήσουμε την κλάση `Player` έτσι ώστε οι παίχτες να μπορούν να γνωστοποιήσουν ο ένας στον άλλο το δωμάτιο με την μεγαλύτερη ενέργεια που έχουν επισκεφτεί κατά τις μέχρι τώρα μετακινήσεις τους.

i) (**3 μονάδες**) Δώστε στην κλάση **`Player`** την πλήρη δήλωση μιας μεταβλητής στιγμιοτύπων με τύπο `Room` και όνομα `mostEnergetic`. Αυτή η μεταβλητή προορίζεται να αποθηκεύσει το δωμάτιο με την μεγαλύτερη ενέργεια που οι παίχτες έχουν επισκεφθεί κατά την μέχρι τώρα μετακίνησή τους, και θα πρέπει να διαμοιράζεται από όλα τα στιγμιότυπα της κλάσης `Player`.

Λύση:

```
private static Room mostEnergetic = null;
```

Grading Guideline:
No points were given if static is omitted
-1 point if variable type was omitted
-1 point if value was initialized to an incorrect value (such as dynamic variable ”location”)
-1 point if variable was declared as final

ii) (**8 μονάδες**) Όταν ένας παίχτης επισκέπτεται ένα δωμάτιο, θα πρέπει να γνωστοποιεί στους άλλους ποιο δωμάτιο έχει την μεγαλύτερη ενέργεια. Τροποποιήστε την υλοποίηση της μεθόδου **`goSouth()`** ώστε να ενημερώνει σωστά την μεταβλητή στιγμιοτύπων `mostEnergetic`:

```
public void goSouth() {
```

Λύση:

```
  Room south = location.getSouth();
  if (south != null) {
    if (mostEnergetic == null ||
        south.getEnergy() > mostEnergetic.getEnergy())
        mostEnergetic = south;
    }
    oldLocation = location;
   location = south;
  }
```

Grading Guideline:
+2 points given for checking if mostEnergetic is null
+2 points given for checking if energy is higher than current one
+2 points given for checking if south is not null
+1 point given for updating location
+1 point given for updating oldLocation correctly
-1 for having small syntax errors

}

Άσκηση 3 (30 μονάδες)

Σας δίνονται οι παρακάτω κλάσεις Java **SuperClass** και **SubClass**.

```
public class SuperClass {
  public void doIt() {
        doItMore();
        System.out.println("Super's doIt");
  }
  public void doItMore() {
        System.out.println("Super's doItMore");
  }
  public void doItSuper() {
        System.out.println("doItSuper");
  }
}

public class SubClass extends SuperClass {
  public void doIt() {
        super.doIt();
        System.out.println("Sub's doIt");
  }
  public void doItMore() {
        System.out.println("Sub's doItMore");
        super.doItMore();
  }
        public void doItSub() {
        System.out.println("doItSub");
  }
}
```

Υποθέστε ότι οι κλάσεις **SuperClass** και **SubClass** έχουν μεταγλωττιστεί, και εκτελούμε στην συνέχεια τις παρακάτω εντολές από κάποιον κώδικα πελάτη:

```
SuperClass superObj = new SuperClass();
SuperClass superObj2 = new SubClass();
SubClass subObj = new SubClass();
```

a) (**12 μονάδες**) Για κάθε μια από τις ακόλουθες κλήσεις μεθόδων, δώστε τι θα τυπωθεί στην στάνταρ έξοδο του προγράμματος ή εξηγήστε γιατί η κλήση της μεθόδου είναι λάθος:

1) `superObj.doItSuper();`

Λύση:
`doItSuper`

2) `superObj.doItSub();`

Λύση:
Illegal, because `doItSub` not method of `Super`.

3) `superObj2.doItSuper();`

Λύση:
`doItSuper`

4) `superObj2.doItSub();`

Λύση:
Illegal, because `doItSub` is not a method of `Super`, and `superObj2` is a variable of type `Super` (even though it holds a sub).

5) `subObj.doItSuper();`

Λύση:
`doItSuper`

6) `subObj.doItSub();`

Λύση:
`doItSub`

b) (**15 μονάδες**) Για κάθε μια από τις ακόλουθες κλήσεις μεθόδων, δώστε τι θα τυπωθεί στην στάνταρ έξοδο του προγράμματος:

1) `superObj.doIt();`

Λύση:
```
Super's doItMore
Super's doIt
```

2) `superObj2.doItMore();`

Λύση:
```
Sub's doItMore
Super's doItMore
```

3) `subObj.doIt();`

Λύση:
```
Sub's doItMore
Super's doItMore
Super's doIt
Sub's doIt
```

c) (**3 μονάδες**) Ένας συμφοιτητής σας δηλώνει ότι, "Η δυνατότητα να καλέσουμε την `super` ως πρώτη γραμμή μιας μεθόδου κατασκευής αντικειμένων (constructor) είναι μια ευκολία, αλλά δεν είναι πραγματικά απαραίτητο. Η υπο-κλάση μπορεί να αρχικοποιήσει τις μεταβλητές στιγμιοτύπων της υπερ-κλάσης της άμεσα στην μέθοδο κατασκευής αντικειμένων της."

Εξηγήστε εάν η παραπάνω δήλωση είναι (α) Σωστή; (β) Λανθασμένη; (γ) Μερικές φορές σωστή και μερικές φορές λανθασμένη.

Λύση:
If all the instance variables were public, protected, or package accessible in the same package as the subclass the student 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 4 (10 μονάδες)

Σας δίνονται οι παρακάτω κλάσεις C1 και C2 σε μια γλώσσα προγραμματισμού η οποία δεν υποστηρίζει κληρονομικότητα κλάσεων.

```
class C1
{
public int[] s;
public int v1;
public int somme()
{
int somme_s = 0;
for (int i=0; i < s.length; i++)
  somme_s = somme_s + s[i];
return somme_s;
}
}// class
```

```
class C2
{
public int[] s;
public String v2;
public int total()
{
int total_s = 0;
int i = 0;
while (i < s.length)
  {
  total_s = total_s + s[i];
  i = i + 1;
  }
return total_s;
}
}// class
```

Πώς μπορείτε να υλοποιήσετε σε μια γλώσσα προγραμματισμού, όπως η Java, η οποία υποστηρίζει κληρονομικότητα κλάσεων τις παραπάνω κλάσεις; Ορίστε μια καινούργια κλάση C η οποία να γενικεύει την κοινή συμπεριφορά και χαρακτηριστικά των κλάσεων C1 και C2. Τροποποιήστε κατάλληλα τον ορισμό των κλάσεων C1 και C2.

Λύση:

```
class C
{
public int[] s;
public int somme()
{
int somme_s = 0;
for (int i=0; i < s.length; i++)
somme_s = somme_s + s[i];
return somme_s;
}
}// class
```

```
class C1 extends C
{
public int v1;
}// class
```

```
class C2 extends C
{
public String v2;
public int total()
{
return somme();
}
}// class
```

Άσκηση 5 (25 μονάδες)

(i) (**4 μονάδες**) Περιγράψτε τις διαφορές μεταξύ της *προδιαγραφής (specification)* και της *υλοποίησης (implementation)* ενός Αφηρημένου Τύπου Δεδομένων (ΑΤΔ).

Λύση:

A specification is a design document, detailing all of the essential features and requirements of the data type. An implementation is a piece of computer code defining data structures and methods which satisfy the requirements of the specification.

A specification of an ADT is definition of the ADT, usually expressed in terms of a list of valid operations, their requirements, and their outcomes in all possible cases. The specification may be expressed formally or semi-formally, but is expressed independently of any programming language context.

An implementation of an ADT is a section of code in a particular programming language, consisting of data structure declarations and method definitions, which faithfully conform to the ADT specification.

(ii) (**3 μονάδες**) Περιγράψτε τα μέσα που έχει στην διάθεσή του ένας προγραμματιστής Java για να διακρίνει μεταξύ της *προδιαγραφής* και της *υλοποίησης* ενός Αφηρημένου Τύπου Δεδομένων (ΑΤΔ).

Λύση:

Java provides the concept of an interface, which can be used to define method headers, but not to implement them. A class can be made to implement an interface, in which case the class has to provide implementations for each of the methods whose headers appear in the interface. The interface can be used as a specification of an ADT, and the implementing class as the implementation of the ADT.

(iii) (**10 μονάδες**) Ένας αφηρημένος τύπος δεδομένων για ένα εικοσιτετράωρο ρολόι προσφέρει λειτουργίες για να θέτει το χρόνο, να διαβάζει το χρόνο, και να προχωρά το χρόνο ανά δευτερόλεπτο. Καθορίστε την προδιαγραφή για τον ΑΤΔ εικοσιτετράωρο ρολόι δηλ. δώστε τις υπογραφές (signatures) των προβλεπόμενων λειτουργιών του, το είδος τους ανάλογα με την επίδραση που έχουν στην κατάσταση των αντικειμένων (constructors, accessors, transformers), καθώς και τις εκ των προτέρων, τις εκ των υστέρων και τις αμετάβλητες συνθήκες (preconditions, postconditions, invariants)..

Λύση:

Operations:-

1 Solution:

set: Clock x Integer x Integer x Integer -> Clock	//Applicative transformer
hours: Clock -> Integer	//Accessor (selector)
mins: Clock ->Integer	//Accessor (selector)
secs: Clock ->Integer	//Accessor (selector)
incr: Clock ->Clock	//Applicative transformer

2 Solution:

set: Clock x Integer x Integer x Integer -> void	//Mutative transformer
hours: Clock -> Integer	//Accessor (selector)
mins: Clock ->Integer	//Accessor (selector)
secs: Clock ->Integer	//Accessor (selector)
incr: Clock ->void	//Mutative transformer

Requirements:-
For every (h,m,s) hours(set(h,m,s)) = h //postcondition της set
For every (h,m,s) mins(set(h,m,s)) = m //postcondition της set
For every (h,m,s) secs(set(h,m,s)) = s //postcondition της set
For every t, secs(incr(t)) = (secs(t)+1) % 60 //postcondition της incr
For every t, mins(incr(t)) = (secs(t)<59) ? mins(t) : (mins(t)+1)%60
//postcondition της incr

For every t, hours(incr(t)) = ((secs(t)<59) || (mins(t)<59)) ? hours(t) : (hours(t)+1)%24
//postcondition της incr
Exceptions:-
set(h,m,s) is illegal if not (0<=s<=59 && 0<=m<=59 && 0<=h<=23)
//precondition της set

0<=s<=59 && 0<=m<=59 && 0<=h<=23 //invariant του ADT

(iv) (**8 μονάδες**) Δώστε μια κλάση που υλοποιεί τον ΑΤΔ εικοσιτετράωρο ρολόι. Υλοποιήστε πλήρως μόνο δύο μεθόδους: (α) μια που να επιτρέπει στο ρολόι να προχωρά το χρόνο ανά δευτερόλεπτο, και (β) μια που να επιστρέφει τον αριθμό ωρών, καθώς και όλων των απαραίτητων μεταβλητών στιγμιοτύπων για την λειτουργία αυτών των μεθόδων.

Λύση:
```
public class Clock {
  private int h,m,s;
  public void incr() {
    if (++s > 59) {
       s = 0;
       if (++m > 59) {
           m = 0;
           h = (h+1)%24;
       }
    }
}
  public int hours() {
    return h;
  }
}
```
Alternative approach of storing total number of seconds, with arithmetic to extract hours, is acceptable.