Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών

ΗΥ-252 – Αντικειμενοστρεφής Προγραμματισμός
Βασίλης Χριστοφίδης

Πρόοδος (3 ώρες)
Ημερομηνία: 21 Νοεμβρίου 2009

Ονοματεπώνυμο:
Αριθμός Μητρώου:

Άσκηση 1 (10 μονάδες) // Ορθότητα κώδικα Java

Σας δίνεται το παρακάτω πρόγραμμα Java:

```
package A;
public class Person {
     protected String name;
     private int age;

     public void older(Person other) {
          if (other.age > this.age)
               System.out.println ("You are older than me");
     }
}
package B;
import A.Person;
public class Student extends Person {
     public void tryPerson(Person other) {
          name = " Dimitris";
          other.name = "Dimitra";
     }
     public void tryStudent(Student other) {
          other.name = "Vassilis";
     }
}
package C;
import A.Person;
import B.Student;
class TestAccess {
     public static void main(String args) {
          Student s1 = new Student();
          Student s2 = new Student();
          Person p = new Person();
          s1.tryPerson(p);
          s1.tryStudent(s2);
     }
}
```

Για κάθε μια από τις παρακάτω εντολές υπογραμμίστε το «ΟΚ» εάν δεν παραβιάζονται και «ΚΟ» εάν παραβιάζονται οι κανόνες πρόσβασης της Java.

	Κλάση	Μέθοδος	Γραμμή	Απάντηση	
1	Person	older()	if (other.age > this.age)	ΟΚ	ΚΟ
2	Student	tryPerson()	name="Dimitris"	ΟΚ	ΚΟ
3	Student	tryPerson()	other.name="Dimitra"	ΟΚ	ΚΟ
4	Student	tryStudent()	other.name="Vassilis"	ΟΚ	ΚΟ

Λύση:
1 OK 2 OK 3 KO 4 OK

Άσκηση 2 (20 μονάδες) // Αρχικοποίηση Αντικειμένων

Σας δίνεται η ακόλουθη κλάση Java:

```
class A {
 int i = foo();
 A(int i) {
  this.i = i +10;
 }
 int foo() {return 8;}
}
```

(α) **(6 μονάδες)** Μας ενδιαφέρει να δημιουργήσουμε ένα καινούργιο στιγμιότυπο της κλάσης A: `A a = new A(15);`
Ποια είναι η τιμή της μεταβλητής στιγμιοτύπων i του αντικειμένου a μέχρι την ολοκλήρωση της εκτέλεσης της μεθόδου κατασκευής της κλάσης A; Με ποια σειρά ανατίθενται οι τιμές στις μεταβλητές του παραπάνω κώδικα και ποιοι κανόνες της Java την υπαγορεύουν;

> **Λύση:**
> When the object a is created, i gets its default value: since i is an int the default value is 0. Next the initializers are executed once. In this case i is assigned the value returned by foo() which is 8. Lastly the body of the constructor is executed, and I gets the value 25: 15 passed as parameter + 10.

(β) **(10 μονάδες)** Σας δίνεται η ακόλουθη υποκλάση της A:

```
class B extends A {
 int i = 55;
 B(int i) {
  super(i+4);
  this.i = super.i +3;
 }
 int foo() {return i;}
}
```

Μας ενδιαφέρει να δημιουργήσουμε ένα καινούργιο στιγμιότυπο της κλάσης B: `B b = new B(15);`
Ποια είναι η τιμή των μεταβλητών στιγμιοτύπων του αντικειμένου β μέχρι την ολοκλήρωση της εκτέλεσης της μεθόδου κατασκευής της κλάσης B; Με ποια σειρά ανατίθενται οι τιμές στις μεταβλητές του παραπάνω κώδικα και ποιοι κανόνες της Java την υπαγορεύουν;

> **Λύση:**
> The object created has two i fileds, one from class A and one from B. Both will get the default value 0. Then the initializers and constructor from A will be executed. The i field declared in A will be initialized to the value returned by foo(). This method has been overloaded in B, where it returns the current value of the i declared in B i.e. 0. Thus, the i declared in A is initialized to 0. Then the constructor of A is executed with parameter 15+4. Thus, i declared in A is assigned to 29.
> Next the initializers and constructor from B are executed. The i declared in B gets the value 55. Finally the body of the B constructor is executed. It sets the i declared in B to the value of the i declared in A+3, that is 32.

(γ) **(4 μονάδες)** Υποθέστε ότι οι κλάσεις A και B ορίζονται όπως παραπάνω. Θα τυπώσει ο παρακάτω κώδικας τους ίδιους ή διαφορετικούς αριθμούς; Δώστε μια σύντομη εξήγηση.

```
B b = new B(15);
A a = b;
System.out.println(a.i);
System.out.println(b.i);
```

Λύση:
It will print different numbers, namely 29 and 32, because fields are not overloaded. More precisely, a.i refers to the filed declared in the class which has used to declare the reference variable a, i.e. the class A. In the same way, b.i refers to the filed declared in class B, so a.i and b.i refer to different i fields.

Άσκηση 3 (10 μονάδες) // Χειρισμός Συμβολοσειρών

Υλοποιήστε μία μέθοδο `sentenceFilter()` η οποία απαριθμεί τις λέξεις που ένας χρήστης δίνει στην στάνταρ είσοδο και απαλείφει αυτές που αντιστοιχούν σε θέση με άρτιο αριθμό. Η μέθοδος δέχεται σαν όρισμα ένα αντικείμενο `Scanner` το οποίο χρησιμοποιείται για να διαβάζει τις λεκτικές μονάδες (tokens) της εισόδου μέχρι να βρει κάποιο από τα σημεία στίξης (punctuation point) τελεία (period '.'), ερωτηματικό (question mark '?') ή θαυμαστικό (exclamation '!'). Η μέθοδος θα πρέπει να επιστρέφει (και όχι να τυπώνει) την συμβολοσειρά (`String`) με όλες τις λέξεις που δεν απαλείφονται διαχωριζόμενες από κενούς χαρακτήρες. Σε ορισμένες περιπτώσεις η επιστρεφόμενη συμβολοσειρά μπορεί να περιέχει την τελευταία λέξη που καταλήγει σε κάποιο σημείο στίξης. Για παράδειγμα, εάν ο χρήστης εισάγει την πρόταση:

```
My, what a delicious-looking durian!
1   2    3 4                   5
```

η μέθοδος επιστρέφει την συμβολοσειρά: `"My, a durian!"`
Ορισμένες άλλες φορές η λέξη με το σημείο στίξης μπορεί να μην συμπεριληφθεί (δηλ. όταν είναι άρτια αρίθμησης) στην επιστρεφόμενη συμβολοσειρά. Σε κάθε περίπτωση η μέθοδος θα πρέπει να εξασφαλίζει ότι το αρχικό γράμμα της πρώτης λέξης που περιλαμβάνεται είναι κεφαλαίο. Για παράδειγμα, εάν ο χρήστης εισάγει την πρόταση:

```
this horrible exam still is not too too hard methinks.
```

η μέθοδος επιστρέφει την συμβολοσειρά: `"This exam is too hard"`

```
public static String sentenceFilter(Scanner input) {
    System.out.println("Please enter a sentence:");
```

Λύση:

```
String ret = input.next(); // read in the first token
// capitalize the first letter of the first word
ret=
ret.substring(0,1).toUpperCase()+ret.substring(1,ret.length());
String nextToken = ret;
int wordCount = 1;
while(!nextToken.endsWith(".") && !nextToken.endsWith("?") &&
         !nextToken.endsWith("!") ) {
     nextToken = input.next();
     wordCount++;
     // for odd-numbered words, add them to the variable "ret"
     if(wordCount % 2 == 1) {
          ret = ret + " " + nextToken;
     }
}
return ret;
}
```

Άσκηση 4 (11 μονάδες) // Σχεδίαση Κλάσεων και Κληρονομικότητα
Μας ενδιαφέρει η υλοποίηση ενός προγράμματος το οποίο επεξεργάζεται μια πραγματική τιμή διπλής ακρίβειας (double) χρησιμοποιώντας μια ακολουθία πράξεων σε σωλήνωση (pipeline). Το πρόγραμμα δέχεται σαν είσοδο έναν πίνακα με τα αντικείμενα χειρισμού (Manipulator) που καθορίζουν τις πράξεις που εφαρμόζονται στον πραγματικό αριθμό. Για

παράδειγμα, ένα αντικείμενο χειρισμού μπορεί να είναι ο υπολογισμός της τετραγωνικής ρίζας, ο επόμενος να διαιρεί με το 2, κλπ.
Ο παρακάτω κώδικας Java δημιουργεί έναν πίνακα με αντικείμενα χειρισμού που αναπαριστούν default πράξεις όπως η στρογγυλοποίηση του πραγματικού, η απαλοιφή του δεκαδικού μέρους, η αύξηση κατά ένα της πραγματικής τιμής, ο τετραγωνισμός της και ο υπολογισμός του λογαρίθμου της με βάση το 8.

```
public static void main(String [] args) {
     if (args.length == 0) System.exit(0); //quit
     Manipulator[] mArray = new Manipulator[] {
          new Manipulator(),
          new AddOneManipulator(),
          new SquareManipulator(),
          new LogarithmManipulator(8.0)
     };
     process(mArray, Double.parseDouble(args[0]));
   }

public static double process(Manipulator[] manipulators, double
value) {
     for (Manipulator m : manipulators) {//loop over array elements
          value = m.manipulate(value);
     }
     return value;
   }
```

Γράψτε τις κλάσεις Java `Manipulator`, `AddOneManipulator`, `SquareManipulator`, και `LogarithmManipulator` που υλοποιούν τις αντίστοιχες πράξεις σε έναν πραγματικό έτσι ώστε ο παραπάνω κώδικας πελάτης να εκτελείται κανονικά.

Βοήθεια: Μπορείτε να χρησιμοποιήσετε τις μεθόδους `Math.floor(double value)` που στρογγυλεύει προς την πλησιέστερη ακέραια τιμή έναν πραγματικό, και `Math.log10(double value)` που υπολογίζει τον λογάριθμο με βάση το 10 μιας τιμής. Για να υπολογίσετε τον λογάριθμο με βάση το b χρησιμοποιήστε τον τύπο `Math.log10(value)/Math.log10(b)`.

Λύση:

```
class Manipulator {
    public double manipulate (double value) {
        return Math.floor(value);
    }
} // 2 points
class AddOneManipulator extends Manipulator {
    public double manipulate (double value) {
        return value + 1;
    }
} // 2 points
class SquareManipulator extends Manipulator {
    public double manipulate (double value) {
        return value*value;
    }
} // 2 points
class LogarithmManipulator extends Manipulator {
    private double base;
    public LogarithmManipulator (double base) {
        this.base = base;
    } // 2 points
    public manipulator (double value) {
        return Math.log10(value)/Math.log10(base);
    } // 2 points
} // 5 points
```

Άσκηση 5 (14 μονάδες) // Χειρισμός Πινάκων και Αντικειμένων

Σας δίνονται οι παρακάτω κλάσεις `Point` και `PointArray`:

```
class Point {
    private int x;
    private int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    public Point clone() {
        return new Point(x, y);
    }
    public boolean equals(Object other) {
        if (!(other instanceof Point)) {
            return false;
        }
        Point point = (Point) other;
        return point.x == this.x && point.y == this.y;
    }
}
class PointArray {
    private Point[] array;

    public PointArray(Point[] array) {
        this.array = array;
    }
    public Point[] getArray() {
        return array;
    }
    public PointArray clone() {
        //...
    }
}
```

Έστω ότι έχουμε:

```
Point[] points = …;
PointArray array = new PointArray(points)
PointArray clone = array.clone();
```

Υλοποιήστε τη μέθοδο `clone()` της κλάσης `PointArray` έτσι ώστε να ισχύει ότι:

α) (**4 μονάδες**) Οι πίνακες των array και clone να δείχνουν στο ίδιο αντικείμενο:
`array.getArray() == clone.getArray()`

Λύση:

```
public PointArray clone() {
    return new PointArray(array);
}
```

β) (**5 μονάδες**) Οι πίνακες των array και clone να δείχνουν σε διαφορετικά αντικείμενα, αλλά τα στοιχεία των δύο πινάκων να είναι τα ίδια:
`array.getArray() != clone.getArray()` και
`array.getArray()[i] == clone.getArray()[i]`, για κάθε επιτρεπτό i (από 0 μέχρι το μέγεθος του πίνακα)

Λύση:

```
public PointArray clone() {
    return new PointArray(array.clone());
}
```

That or manually copy the elements of the array

γ) (**5 μονάδες**) Όπως παραπάνω, αλλά να τα στοιχεία των πινάκων να μην είναι ίδια, αλλά ίσα όταν συγκρίνονται με τη μέθοδο `equals()`:
`array.getArray() != clone.getArray()` και
`array.getArray()[i] != clone.getArray()[i]`, για κάθε i, και
`array.getArray()[i].equals(clone.getArray()[i])`

Σημείωση: Μπορείτε να αλλάξετε την κλάση `Point` εφόσον το ~~κρίνεται~~ κρίνετε απαραίτητο.

Λύση:
```
public PointArray clone() {
    Point[] newArray = new Point[array.length];
    for (int i = 0; i < newArray.length; i++) {
        newArray[i] = array[i].clone();
        // or implement methods getX() and getY() in class
        // Point and then have:
        // newArray[i] =
        //     new Point(array[i].getX(), array[i].getY());
        // or implement clone method in Point and have:
        // newArray[i] = array[i].clone()
    }
    return new PointArray(newArray);
}
```

Άσκηση 6 (12 μονάδες) // Χειρισμός Εξαιρέσεων

(α) **(2 μονάδες)** Σας δίνεται η παρακάτω κλάση `ExException`:
```
public class ExException {
  public ExException() {
    super();
  }
}
```
Είναι νόμιμη η εντολή **`throw new`** `ExException()`; Δώστε μια σύντομη εξήγηση.

Λύση:
The throw statement is illegal because class `ExException` is not a subclass of class `Throwable`.

(β) **(10 μονάδες)** Σας δίνεται η παρακάτω κλάση `C`:
```
public class C {
  public static void m1(int p1) {
   try {
       System.out.println("1: " + p1);
       p1= m2(p1 + 1);
       System.out.println("2: " + p1);
   } catch (ArithmeticException e) {
       System.out.println("3: " + p1);
   }
   System.out.println("4: " + p1);
 }
 public static int m2(int q1) {
   q1= q1 + 1;
   try {
       System.out.println("5: " + q1);
       if (q1 != 10)
          throw new IOException();
       System.out.println("6: " + q1);
   } catch (ArithmeticException e) {
       System.out.println("7: " + q1);
       q1= q1 / 0;
```

```
        System.out.println("8: " + q1);
    } catch (IOException e) {
        System.out.println("9: " + q1);
        q1= q1 / 0;
        System.out.println("10: " + q1);
    }
    System.out.println("11: " + q1);
    return q1+2;
  }
}
```

Τι θα τυπωθεί στην στανταρ έξοδο του προγράμματος όταν η παρακάτω εντολή εκτελεστεί;
`> C.m1(5);`

> **Λύση:**
> 1: 5
> 5: 7
> 9: 7
> 3: 5
> 4: 5

Άσκηση 7 (23 μονάδες) // Αφαιρετικοί Τύποι Δεδομένων

Σας δίνεται το παρακάτω συμβόλαιο του Αφαιρετικού Τύπου Δεδομένων **Ημερομηνία** (`Date`) χωρίς να προσδιορίζεται το έτος της: για παράδειγμα Δεκέμβριος 21, Ιανουάριος 20:

```
class Date {
/** Make a new Date object with the specified month and day.
  * Requires: "month" must be between 1 and 12. "day" must be
  * between 1 and the number of days in that month (29 in case of
  * February).
*/
public Date(int month, int day) { ... }
/** The month (1-12). */
public int month() { ... }
/** The day (1-31). */
public int day() { ... }
/** A human readable representation, e.g., "May 17" */
public String toString() { ... }
/** The next day of the year. */
public Date tomorrow() { ... }
/** The previous day of the year. */
public Date yesterday() { ... }
}
```

(α) **(3 μονάδες)** Ποιες από τις μεθόδους του συμβολαίου του ΑΤΔ `Date` είναι κατασκευαστές (constructors), παρατηρητές (accessors) ή μετασχηματιστές (transformers); Στην τελευταία περίπτωση, αλλοιώνουν ή όχι την κατάσταση των στιγμιότυπων του ΑΤΔ (πρόκειται για mutable ή immutable ΑΤΔ);

> **Λύση:**
> Constructor~~s~~: Date~~, tomorrow, yesterday.~~
> Accessors: month, day, ~~tomorrow, yesterday,~~ toString
> Transformers: ~~none~~tomorrow (applicative), yesterday (applicative)
> This is an immutable data abstraction.

(β) **(4 μονάδες)** Το συμβόλαιο των μεθόδων `tomorrow()` και `yesterday()` είναι ελλιπές γιατί δεν λαμβάνει υπόψη τα δίσεκτα χρόνια (leap years). Τροποποιήστε την υπογραφή και την σημασιολογία της μεθόδου `tomorrow()` ώστε να διορθωθεί αυτό το πρόβλημα. Όλες οι λογικά σωστές λύσεις θα γίνουν δεκτές σε αυτό το ερώτημα.

Λύση:

```
/** Creates a new Date representing the next day of the year.
 ** Requires that the current day is not December 31. For
 ** February 28, the next day is February 29 if the year is a
 ** leap year, March 1 otherwise.
 @param leap_year whether the year is a leap year.
*/
public Date tomorrow(boolean leap_year) { ... }
// int year is also acceptable
```

Some people simply assumed that it was a leap year. But this violated the original intent of the data abstraction, so it wasn't a good solution.

(γ) **(2 μονάδες)** Μας ενδιαφέρει τώρα η υλοποίηση του ΑΤΔ από μια κλάση `Date` χρησιμοποιώντας μια μοναδική μεταβλητή στιγμιοτύπων (`day_count`) που μετρά τις ημέρες του χρόνου από την 1 Ιανουαρίου. Ποια είναι, εάν φυσικά υπάρχει, η αμετάβλητη συνθήκη αναπαράστασης (representation invariant) για την υλοποιητική κλάση `Date`.

```
public class Date {
// The number of days elapsed since the beginning of a leap year,
// inclusive. Thus, January 1 is represented by 1 and December 31,
// by 366.
private int day_count;
…
}
```

Λύση:

The invariant is that day count must be between 1 and 366.

(δ) **(6 μονάδες)** Υλοποιήστε την μέθοδο `tomorrow()` στην κλάση `Date` χρησιμοποιώντας την μεταβλητή στιγμιοτύπων `int day_count` και την σημασιολογία της μεθόδου που δώσατε στο ερώτημα (β). Στην υλοποίησή σας μπορείτε να χρησιμοποιήσετε όλες μεθόδου του συμβολαίου της `Date` καθώς και ιδιωτικές μεθόδους (συμπεριλαμβανομένων και κατασκευαστών) που εσείς θα ορίσετε.

Λύση:

```
private Date(int c) { day_count = c; }
public Date tomorrow(boolean leap_year) {
  Date ret = new Date(day_count + 1);
  if (!leap_year && day_count == 59)
  ret.day_count = 61;
  return ret;
}
```

If you made December 31 wrap around in `tomorrow()`, you needed to check for that too.

(ε) **(8 μονάδες)** Μας ενδιαφέρει στην συνέχεια να ορίσουμε μια υποκλάση της `Date` έτσι ώστε να λαμβάνουμε υπόψη και τον χρόνο, με το όνομα `YearDate`.

```
class YearDate extends Date {
  private int year;
...
}
```

Υλοποιήστε τις παρακάτω μεθόδους της υποκλάσης `YearDate` χωρίς να τροποποιήσετε την `Date`.

- Την μέθοδο κατασκευής `YearDate(int month, int day, int year)`
- Την μέθοδο πρόσβασης `String toString()`
- Την μέθοδο `YearDate tomorrow()` η οποία επιστρέφει την επόμενη ημέρα ενδεχομένως αυξάνοντας τον χρόνο. Σε αυτή την υλοποίηση μπορείτε να χρησιμοποιήσετε την μέθοδο `isLeapYear(int year)` που δίνεται στην συνέχεια.

```
/** Is this a leap year in the Gregorian calendar. */
```

```
boolean isLeapYear(int year) {
   return (year%4 == 0) && (year%100 != 0) || (year%400 == 0);
}
```

Λύση:

```
YearDate(int month, int day, int year) {
   super(month, day);
   this.year = year;
} 2 points
String toString() {
   return super.toString() + ", " + year;
} 1 point
YearDate tomorrow() {
   Date d = tomorrow(isleapyear(year));
   if (d.month() == 1 && d.day() == 1) {
       return new YearDate(d.month(), d.day(), year+1);
   } else {
       return new YearDate(d.month(), d.day(), year);
   }
} 5 points
```