Οικονομικό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής

3222: Προγραμματισμός Υπολογιστών με Java (Μ-Ω)

(Εαρινό εξάμηνο 2021 – 2022)

2η Σειρά Ασκήσεων

Βαθμολογία: 10 μονάδες (5% του τελικού βαθμού)

Ημερομηνία Παράδοσης: 2/5/2022

Επικοινωνία: xsk@aueb.gr

Άσκηση 1η (Παραδοτέο: App1.java) - 1 μονάδα

Δίνεται το αρχείο **App1.java** που περιέχει τις παρακάτω τάξεις:

```
abstract class Akolou8ia {
    abstract double oros(int i);
}// Akolou8ia

class App1 {
    public static void main(String[] args) {
          Akolou8ia a = new TetragwnikiAkolou8ia();
          System.out.println(a8roisma(a,100));

    }// main
    static double a8roisma(Akolou8ia ak, int n) {
          double sum = 0;
          for ( int i = 1; i <= n; i++ )
               sum += ak.oros(i);
          return sum;
    }// a8roisma
}//App1
```

Κάνετε τις απαραίτητες προσθήκες/ορισμούς ΧΩΡΙΣ ΝΑ ΔΙΑΓΡΑΨΕΤΕ/ΑΛΛΑΞΕΤΕ τους ορισμούς των τάξεων ***Akolou8ia*** και ***App1*** έτσι ώστε όταν εκτελεστεί το πλήρες πρόγραμμα, να εμφανίζεται η τιμή του αθροίσματος $1^2+2^2+3^2+\ldots+100^2$

Συμπληρώστε το αρχείο **App1.java**

Άσκηση 2η (Παραδοτέο: App2.java) - 1 μονάδα

Τα αποτελέσματα των εκλογών μεταξύ τριών υποψηφίων σε μια χώρα με πέντε εκλογικές περιφέρειες ήταν τα ακόλουθα:

Εκλογική Περιφέρεια	Υποψήφιος Α	Υποψήφιος Β	Υποψήφιος Γ
1	182	41	202
2	145	85	325
3	195	15	115
4	110	24	407
5	255	11	357

Να γραφεί η εφαρμογή ***App2*** η οποία:

1) Αρχικοποιεί τα στοιχεία **ενός δισδιάστατου πίνακα** 5 γραμμών και 3 στηλών, όπως παραπάνω. Οι διαστάσεις του πίνακα πρέπει να έχουν δηλωθεί ως σταθερές (final).

2) Εμφανίζει τον πίνακα (όπως παραπάνω).

3) Υπολογίζει και εμφανίζει τον συνολικό αριθμό ψήφων που έλαβε κάθε υποψήφιος.

4) Υπολογίζει και εμφανίζει το ποσοστό που έλαβε κάθε υποψήφιος επί του συνόλου των ψήφων.

5) Εάν κάποιος υποψήφιος έλαβε πάνω από 50% των ψήφων να εκτυπώνει το όνομά του (Γ, στην περίπτωση αυτή) ως νικητή των εκλογών.

6) Εάν δεν υπάρχει υποψήφιος που να έλαβε πάνω από το 50% των ψήφων να δηλώνεται ότι θα διεξαχθεί επαναληπτικός γύρος μεταξύ των δύο πρώτων και να εκτυπώνονται τα στοιχεία τους και τα ποσοστά που αυτοί έλαβαν.

Συμπληρώστε το αρχείο **App2.java**

Άσκηση 3η (Παραδοτέο: App3.java) - 1 μονάδα

Δίνεται το αρχείο **App3.java** που περιέχει όλες τις παρακάτω τάξεις:

```
class Shape {
	void draw() {}
	void erase() {}
}
class Circle extends Shape {
	void draw() {
		System.out.println("Circle.draw()");
	}
	void erase() {
		System.out.println("Circle.erase()");
	}
}
class Square extends Shape {
	void draw() {
		System.out.println("Square.draw()");
	}
	void erase() {
		System.out.println("Square.erase()");
	}
}
class Triangle extends Shape {
	void draw() {
		System.out.println("Triangle.draw()");
	}
	void erase() {
		System.out.println("Triangle.erase()");
	}
}
class App3 {
	static Shape randShape() {
		switch((int)(Math.random() * 3)) {
			default:
			case 0: return new Circle();
			case 1: return new Square();
			case 2: return new Triangle();
		}
	}
```

```
    public static void main(String[] args) {
        _____________________________________ //1
        for _________________________________ //2
            _________________________________
        for _________________________________ //3
            _________________________________
    }//main
}//App3
```

Στο κύριο πρόγραμμα συμπληρώστε τα κενά (μην προσθέσετε νέες γραμμές-εντολές), ειδικότερα:

1. Ορίστε έναν ***πίνακα*** 10 θέσεων. Σε κάθε θέση του πίνακα πρέπει να μπορεί να αποθηκεύεται ένα σχήμα δηλαδή ένα αντικείμενο τύπου ***Circle*** ή ***Square*** ή ***Triangle***.

2. Γεμίστε τον πίνακα με σχήματα δηλαδή αντικείμενα που έχουν δημιουργηθεί με τυχαίο τρόπο μέσω της μεθόδου **randShape**().

3. Εμφανίστε τα σχήματα του πίνακα, δηλαδή για κάθε σχήμα του πίνακα καλέστε τη μέθοδο **draw**() (πολυμορφισμός).

Συμπληρώστε το αρχείο **App3.java**

Άσκηση 4η (Παραδοτέο: App4.java) -1 μονάδα

Δίνεται το αρχείο **App4.java** που περιέχει όλες τις παρακάτω τάξεις:

```
abstract class Animal {

	String name;
	int animals = 0;
	Animal(String n) {
		name = n;
		animals++;
	}
	abstract void speak();
	int numberOfAnimals() {
		return animals;
	}
}

class Dog extends Animal {

	String sound = "woof";
	int dogs = 0;

      // Fill your code here
}

class Cat extends Animal {

	String sound = "miaou";
	int cats = 0;

      // Fill your code here
}
```

```
class App4 {

    public static void main(String[] args) {

        Animal[] animal = {   new Cat("stella"),
                              new Cat("ziggy"),
                              new Dog("azor") };


        System.out.println("Cats: " + Cat.numberOfCats());
        System.out.println("Dogs: " + Dog.numberOfDogs());
        System.out.println("Animals: "+Animal.numberOfAnimals());

        for ( int i = 0; i < animal.length; i++ )
            animal[i].speak();
    }
}
```

Η εφαρμογή ***App4*** δημιουργεί δύο αντικείμενα τύπου ***Cat*** και ένα τύπου ***Dog***. Στη συνέχεια εμφανίζει το πλήθος των αντικειμένων που δημιούργησε, ως εξής:

```
cats: 2
dogs: 1
animals: 3
```

Τέλος εμφανίζει την ιδιότητα **sound** κάθε αντικειμένου:

```
stella: miaou
ziggy: miaou
azor: woof
```

Διορθώστε και **συμπληρώστε** τις κλάσεις ***Animal***, ***Dog*** και ***Cat*** έτσι ώστε το πρόγραμμα να μην έχει συντακτικά σφάλματα και όταν εκτελείται να λαμβάνουμε τα παραπάνω αποτελέσματα **ΧΩΡΙΣ ΝΑ ΑΛΛΑΞΕΤΕ** το κύριο πρόγραμμα.

Συμπληρώστε το αρχείο **App4.java**

Άσκηση 5η (Παραδοτέα: Clock.java, AMPMClock.java) – 2 μονάδες

Σε αυτή την άσκηση θα χρειαστείτε την τάξη ***Clock*** (άσκηση 5 από την 1η σειρά σκήσεων) για να ορίσετε την τάξη ***AMPMClock***, στην οποία θα υπάρχει η επιλογή εμφάνισης της ώρας με την ένδειξη **πμ** ή **μμ** (δηλαδή προ/μετά μεσημβρίας).

Η ***AMPMClock*** θα πρέπει να **επεκτείνει** την ***Clock***. Συγκεκριμένα, τα αντικείμενα *AMPMClock* θα δέχονται επιπλέον το εξής μήνυμα: **setAMPM(boolean yes)**: Στα επόμενα μηνύματα **toString**(), η ώρα που επιστρέφεται χρησιμοποιεί την ένδειξη πμ ή μμ εάν η yes έχει τιμή true. Στην περίπτωση που η yes έχει τιμή false, η ώρα δε θα χρησιμοποιεί την ένδειξη πμ ή μμ (δηλ. όπως γίνονταν και στην ***Clock***).

Για παράδειγμα, οι εντολές:

```
AMPMClock clock = new AMPMClock();
clock.setHour(16);
clock.setMin(28);
clock.setSec(58);
println(clock.toString());
clock.setAMPM(true);
println(clock.toString());
clock.tick();
clock.tick();
println(clock.toString());
clock.setAMPM(false);
clock.tick();
println(clock.toString());
```

θα πρέπει να εμφανίζουν:

```
16:28:58
04:28:58 μμ
04:29:00 μμ
16:29:01
```

Αξιοποιήστε όσο τον δυνατόν περισσότερο την ***κληρονομικότητα*** έτσι ώστε:

Α) <u>να μην αλλάξετε τον ορισμό της τάξης *Clock*</u> της 1ης Σειράς Ασκήσεων (θα χρειαστεί μόνο να αλλάξετε τους τελεστές πρόσβασης στα δεδομένα της τάξης *Clock* από private σε protected έτσι ώστε να έχει πρόσβαση σε αυτά η υποκλάση *AMPMClock*).

Β) να μην επαναλάβετε τη λειτουργικότητα της `Clock` στον ορισμό της τάξης `AMPMClock` (μέσω της κληρονομικότητας).

Συμπληρώστε το αρχείο **AMPMClock.java**.

Μην ξεχάσετε να υποβάλετε μαζί και το αρχείο **Clock.java**

Άσκηση 6η (Παραδοτέο: AMPMClockApp.java) - 1 μονάδα

Θα πρέπει να υλοποιήσετε την εφαρμογή ***AMPMClockApp*** η οποία θα δημιουργεί ένα ψηφιακό ρολόι ως ένα αντικείμενο της τάξη ***AMPMClock*** και θα το αρχικοποιεί στην ώρα 16:28:58.

Κατόπιν θα εμφανίζει τη νέα ώρα κάθε δευτερόλεπτο (πραγματικού χρόνου) που περνάει, για τα επόμενα 3 λεπτά με την ένδειξη **πμ** ή **μμ** (δηλαδή προ/μετά μεσημβρίας).

Μπορείτε να κάνετε χρήση της έτοιμης συνάρτησης TimeUnit.SECONDS.sleep(i) του πακέτου java.util.concurrent.TimeUnit η οποία «παγώνει» την εκτέλεση του προγράμματός σας για i δευτερόλεπτα.

Συμπληρώστε το αρχείο **AMPMClockApp.java**.

Άσκηση 7η (Παραδοτέα: Student.java, StudentList.java) – 2 μονάδες

Έστω η τάξη ***StudentList*** που παριστάνει μια λίστα φοιτητών σαν έναν μονοδιάστατο πίνακα αντικειμένων της τάξης ***Student***.

Η τάξη ***Student*** περιέχει τα δεδομένα ενός φοιτητή που είναι το ονοματεπώνυμό του (συμβολοσειρά), ο αριθμός μητρώου του (συμβολοσειρά) και ο βαθμός του (ακέραιος αριθμός) στο μάθημα «**Προγραμματισμός Υπολογιστών με Java**».

Α) Αρχικά ορίστε την τάξη φοιτητής ***Student*** που περιέχει, εκτός από τις μεταβλητές αντικειμένου (δεδομένα) του φοιτητή και μία μέθοδο κατασκευαστή που παίρνει ως παραμέτρους δύο συμβολοσειρές και έναν ακέραιο αριθμό που παριστάνουν έναν ορισμένο φοιτητή. Δηλώστε τις μεταβλητές ως ιδιωτικές (private). Επίσης να ορίσετε όποια άλλη μέθοδο θα σας χρειαστεί.

Β) Ορίστε την τάξη λίστα φοιτητών ***StudentList*** που περιέχει έναν **μονοδιάστατο πίνακα myList** (έστω 50 θέσεων). Σε κάθε θέση του πίνακα μπορεί να αποθηκεύεται ένα αντικείμενο τύπου *Student*. Η ***StudentList*** περιέχει (ορίζει) τις μεθόδους InsertStudent, LookupStudent, και DisplayList.

- Η μέθοδος **InsertStudent** διαβάζει δεδομένα ενός φοιτητή και τα αποθηκεύει στη λίστα (πίνακα).
- Η μέθοδος **LookupStudent** ψάχνει στη λίστα ένα φοιτητή με βάση τον αριθμό μητρώου του και εμφανίζει το βαθμό του. Αν ο φοιτητής δεν υπάρχει στη λίστα εμφανίζει ένα κατάλληλο μήνυμα.
- Η μέθοδος **DisplayList** εμφανίζει, τα στοιχεία όλων των φοιτητών που υπάρχουν στη λίστα.

Συμπληρώστε τα αρχεία **Student.java**, **StudentList.java**

Άσκηση 8η (Παραδοτέο: StudentApp.java) – 1 μονάδα

Γράψτε μια εφαρμογή ***StudentApp*** που δημιουργεί μια λίστα φοιτητών ως ένα αντικείμενο της τάξης ***StudentList*** και μέσω ενός **menu** εκτελεί τις πράξεις με βάση τις επιλογές του χρήστη.

Οι πράξεις είναι η εισαγωγή φοιτητή στη λίστα (**InsertStudent**), αναζήτηση φοιτητή στη λίστα (**LookupStudent**) με βάση τον αριθμό μητρώου του και η εμφάνιση ολόκληρης της λίστας (**DisplayList**):

1. Εισαγωγή φοιτητή

2. Αναζήτηση φοιτητή

3. Εμφάνιση λίστας

0. Έξοδος

Συμπληρώστε το αρχείο **StudentApp.java**

Επιπλέον Γενικές Οδηγίες:

1. Η 2η σειρά ασκήσεων πρέπει να υλοποιηθεί **ατομικά**.
2. **Μη χρησιμοποιείτε** στα προγράμματα σας **ελληνικούς χαρακτήρες** ούτε ως σχόλια, ούτε ως μηνύματα προς το χρήστη.
3. Συμπληρώστε **μόνο τα αρχεία java** όπως ζητείται χωρίς να δημιουργήσετε επιπλέον αρχεία.
4. Συμπληρώστε σε όλα τα προγράμματα που θα παραδώσετε το όνομα και τον αριθμό του φοιτητικού σας μητρώου (με λατινικούς χαρακτήρες).
5. Κάθε πρόγραμμα σας πρέπει να μεταγλωττίζεται και να εκτελείται στη **γραμμή εντολών** (όχι μέσω κάποιου IDE).
6. Παρακαλούμε να υποβάλετε όλα τα προγράμματα σας (αρχεία **.java**) ως εργασία στο eclass (2η Σειρά Ασκήσεων), **αφού πρώτα τα συμπιέσετε σε ένα αρχείο** με όνομα τον αριθμό του φοιτητικού σας μητρώου, για παράδειγμα 3210000.zip

Καλή Επιτυχία!