Οικονομικό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής

3222: Προγραμματισμός Υπολογιστών με Java

(Εαρινό εξάμηνο 2022 – 2023)

2η Σειρά Ασκήσεων

Ημερομηνία Παράδοσης: 12/5/2023

Έκδοση 1η: 28/4/2023

Άσκηση 1

Η κλάση **Customer** περιέχει τα δεδομένα ενός πελάτη που είναι:

- το **όνομα** του πελάτη (τύπου **String**),
- ο **κωδικός** του πελάτη (τύπου **String**) και
- η **ημερομηνία εγγραφής** του πελάτη στο πελατολόγιο (τύπου **myDate**).

1. Ορίστε την κλάση **myDate** με δεδομένα τρεις ακέραιους αριθμούς που παριστάνουν την ημέρα, το μήνα και το έτος εγγραφής του πελάτη στο πελατολόγιο ως ιδιωτικές μεταβλητές της κλάσης. Επίσης ορίστε τον κατασκευαστή και όποια άλλη μέθοδο χρειαστεί.

2. Ορίστε την κλάση **Customer** που περιέχει τα δεδομένα του πελάτη και μία μέθοδο κατασκευαστή που παίρνει ως παραμέτρους δύο συμβολοσειρές (όνομα και κωδικός) και τρεις ακεραίους (ημερομηνία εγγραφής). Δηλώστε τις μεταβλητές που περιέχουν τα δεδομένα ως ιδιωτικές. Επίσης να ορίσετε όποια άλλη μέθοδο θα σας χρειαστεί.

Δημιουργείστε τα αρχεία **Customer.java** και **myDate.java**

Άσκηση 2

Η κλάση **CustomerList** δημιουργεί και διαχειρίζεται μια λίστα πελατών σαν έναν μονοδιάστατο πίνακα αντικειμένων της κλάσης **Customer**.

Ορίστε την κλάση **CustomerList** που περιέχει έναν **μονοδιάστατο πίνακα myList** (έστω 50 θέσεων). Σε κάθε θέση του πίνακα αποθηκεύεται ένα αντικείμενο τύπου Customer.

Η **CustomerList** περιέχει (ορίζει) τις μεθόδους:

- **InsertCustomer:** διαβάζει δεδομένα ενός πελάτη και τα αποθηκεύει στη λίστα (πίνακα).
- **LookupCustomer:** ψάχνει στη λίστα έναν πελάτη με βάση τον κωδικό του και εμφανίζει το όνομά του και την ημερομηνία εγγραφής του. Αν ο πελάτης δεν υπάρχει στη λίστα εμφανίζει ένα κατάλληλο μήνυμα.
- **DisplayList:** εμφανίζει, τα στοιχεία όλων των πελατών που υπάρχουν στη λίστα.

Δημιουργείστε το αρχείο **CustomerList.java**

Άσκηση 3

Γράψτε μια εφαρμογή **CustomerApp** που δημιουργεί μια λίστα πελατών ως ένα αντικείμενο της κλάσης **CustomerList** και μέσω μιας λίστας επιλογών **(menu)** εκτελεί επαναληπτικά τις πράξεις με βάση τις επιλογές του χρήστη.

Οι πράξεις είναι η εισαγωγή πελάτη στη λίστα, αναζήτηση πελάτη στη λίστα με βάση τον κωδικό του και η εμφάνιση ολόκληρης της λίστας πελατών:

1. Εισαγωγή πελάτη

2. Αναζήτηση πελάτη με κωδικό

3. Εμφάνιση λίστας πελατών

0. Έξοδος

Δημιουργείστε το αρχείο **CustomerApp.java**

Άσκηση 4

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

Voting District	A	B	C
1	182	41	202
2	145	85	325
3	195	15	115
4	110	24	407
5	255	11	357

Να γραφεί η εφαρμογή **VotingApp** η οποία:

1) Αρχικοποιεί τα στοιχεία **ενός δισδιάστατου πίνακα** 5 γραμμών και 3 στηλών, όπως παραπάνω.

2) Εμφανίζει τον πίνακα.

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. Τα ποσοστά να εμφανίζονται με δύο δεκαδικά ψηφία (για παράδειγμα 23,45%).

5) Εάν κάποιος υποψήφιος έλαβε πάνω από 50% των ψήφων να εκτυπώνει το όνομά του (C, 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.

Η εφαρμογή πρέπει να δίνει σωστά αποτελέσματα με οποιαδήποτε αρχικά δεδομένα (αριθμούς ψήφων).

Δημιουργείστε το αρχείο **VotingApp.java**

Άσκηση 5

Έστω ότι το αρχείο **ShapeApp.java** περιέχει όλες τις παρακάτω κλάσεις:

```
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 ShapeApp {
	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
}
```

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).

Δημιουργείστε το αρχείο **ShapeApp.java**

Άσκηση 6

Στην άσκηση αυτή θα κατασκευάσετε ένα ψηφιακό ρολόι που εμφανίζει την ώρα στη **24-ωρη μορφή**, δηλαδή από **00:00:00** ως **23:59:59**.

Το ρολόι θα αναπαρίσταται στο πρόγραμμά σας ως ένα αντικείμενο της κλάσης **Clock** την οποία θα πρέπει πρώτα να ορίσετε και υλοποιήσετε σύμφωνα με τις παρακάτω προδιαγραφές:

1. Τα **δεδομένα** της κλάσης **Clock** είναι **τρεις ακέραιοι** αριθμοί που παριστάνουν την ώρα, τα λεπτά και τα δευτερόλεπτα, με **ορατότητα protected**.

2. Οι **μέθοδοι** της κλάσης **Clock** πρέπει να είναι οι εξής:

- **setHour**(int h): Θέτει την ώρα στην τιμή h.
  Θεωρήστε ότι πάντα ισχύει $0 \leq h \leq 23$.

- **setMin**(int m): Θέτει τα λεπτά στην τιμή m.
  Θεωρήστε ότι πάντα ισχύει $0 \leq m \leq 59$.

- **setSec**(int s): Θέτει τα δευτερόλεπτα στην τιμή s.
  Θεωρήστε ότι πάντα ισχύει $0 \leq s \leq 59$.

- **tick**(): προχωράει το ρολόι κατά 1 δευτερόλεπτο, αλλάζοντας κατάλληλα τις τιμές των μεταβλητών (δεδομένων).

- **toString**(): επιστρέφει την ώρα ως String στη μορφή **ΩΩ:ΛΛ:ΔΔ**, όπου ΩΩ είναι τα δύο ψηφία που δείχνουν την ώρα, ΛΛ τα ψηφία των λεπτών και ΔΔ αυτά των δευτερολέπτων.

Εφόσον χρειαστεί μπορείτε να ορίσετε στην κλάση και επιπλέον μεθόδους.

Δημιουργείστε το αρχείο **Clock.java**

Άσκηση 7

Θα πρέπει να υλοποιήσετε ένα κυρίως πρόγραμμα **ClockApp.java** το οποίο θα δημιουργεί ένα ψηφιακό ρολόι ως ένα αντικείμενο της κλάση **Clock.**

Το πρόγραμμα ζητά και διαβάζει από το χρήστη την τρέχουσα **ώρα**, τα **λεπτά** και τα **δευτερόλεπτα** και εμφανίζει τη νέα ώρα σε 24-ωρη μορφή για κάθε δευτερόλεπτο **(πραγματικού χρόνου)** που περνάει, για τα επόμενα τρία λεπτά.

Μπορείτε να κάνετε χρήση της έτοιμης συνάρτησης:

TimeUnit.SECONDS.sleep(i)

του πακέτου:

java.util.concurrent.TimeUnit

η οποία συνάρτηση, όταν κληθεί, «παγώνει» την εκτέλεση του προγράμματός σας για **i** δευτερόλεπτα. Για το λόγο αυτό, στην αρχή της κλάσης, να κάνετε **import** το πακέτο. Επιπλέον η δήλωση της main θα πρέπει ως εξής:

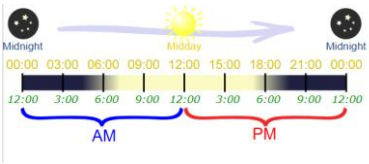
public static void main (String args[]) throws Exception

Έτσι, αν για παράδειγμα ο χρήστης δώσει ως τρέχουσα ώρα, λεπτά και δευτερόλεπτα τους ακεραίους 23, 59 και 58 αντίστοιχα, το πρόγραμμα θα εμφανίζει:

```
23:59:58
23:59:59
00:00:00
00:00:01
00:00:02
00:00:03
κλπ
```

Δημιουργείστε το αρχείο **ClockApp.java**

Άσκηση 8



Σε αυτή την άσκηση θα χρειαστείτε την κλάση **Clock** για να ορίσετε την κλάση **AMPMClock**, στην οποία θα υπάρχει η επιλογή εμφάνισης της ώρας σε **μορφή 12 ωρών** με την ένδειξη **am** ή **pm** (δηλαδή προ/μετά μεσημβρίας).

Η **AMPMClock** θα πρέπει να επεκτείνει (κληρονομεί) την κλάση **Clock**.

Συγκεκριμένα, τα αντικείμενα **AMPMClock** θα δέχονται επιπλέον το εξής μήνυμα:

- **setAMPM(boolean yes):** Στα επόμενα μηνύματα **toString**(), η ώρα που επιστρέφεται είναι σε **μορφή 12 ωρών**, χρησιμοποιεί δηλαδή την ένδειξη **am** ή **pm** εάν η **yes** έχει τιμή **true**. Στην περίπτωση που η **yes** έχει τιμή **false**, η ώρα δε θα χρησιμοποιεί την ένδειξη pm ή am, δηλαδή θα είναι σε **μορφή 24 ωρών** όπως γίνονταν και στην **Clock**.

Για παράδειγμα, με βάση τα παραπάνω, οι εντολές:

```
AMPMClock clock = new AMPMClock();
clock.setHour(23);
clock.setMin(59);
clock.setSec(58);
System.out.println(clock.toString());
clock.setAMPM(true);
System.out.println(clock.toString());
clock.tick();
clock.tick();
System.out.println(clock.toString());
clock.setAMPM(false);
clock.tick();
System.out.println(clock.toString());
```

24-hour clock	12-hour clock
00:01	12:01 AM
01:00	1:00 AM
02:00	2:00 AM
03:00	3:00 AM
04:00	4:00 AM
05:00	5:00 AM
06:00	6:00 AM
07:00	7:00 AM
08:00	8:00 AM
09:00	9:00 AM
10:00	10:00 AM
11:00	11:00 AM
12:00	12:00 PM
13:00	1:00 PM
14:00	2:00 PM
15:00	3:00 PM
16:00	4:00 PM
17:00	5:00 PM
18:00	6:00 PM
19:00	7:00 PM
20:00	8:00 PM
21:00	9:00 PM
22:00	10:00 PM
23:00	11:00 PM
00:00	12:00 AM

θα πρέπει να εμφανίζουν:

```
23:59:58
11:59:58 pm
12:00:00 am
00:00:01
```

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

Α) να μην αλλάξετε τον ορισμό της κλάσης **Clock**

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

Δημιουργείστε το αρχείο **AMPMClock.java**

Άσκηση 9

Θα πρέπει να υλοποιήσετε την εφαρμογή **AMPMClockApp** η οποία θα δημιουργεί ένα ψηφιακό ρολόι ως ένα αντικείμενο της κλάση **AMPMClock.**

Το πρόγραμμα ζητά και διαβάζει από το χρήστη την τρέχουσα **ώρα**, τα **λεπτά** και τα **δευτερόλεπτα.** Επιπλέον ζητά και διαβάζει από το χρήστη τη μορφή της ώρας (12 για **12-ωρη μορφή** με am/pm ή 24 για **24-ωρη μορφή**) και εμφανίζει στη μορφή αυτή τη νέα ώρα σε κάθε δευτερόλεπτο **(πραγματικού χρόνου)** που περνάει, για τα επόμενα τρία λεπτά.

Δημιουργείστε το αρχείο **AMPMClockApp.java**

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. Δημιουργείστε 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
7. Για οποιαδήποτε απορία μπορείτε να επικοινωνείτε με τους διδάσκοντες στα εργαστήριά σας την κ. Μαρία Τογαντζή (mst@aueb.gr) και τον κ. Χρήστο Καλέργη (xsk@aueb.gr).

Καλή Επιτυχία!