Οικονομικό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής

3222: Προγραμματισμός Υπολογιστών με Java (Μ-Ω)

(Εαρινό εξάμηνο 2021 – 2022)

1η Σειρά Ασκήσεων

Βαθμολογία: 10 μονάδες (5% του τελικού βαθμού)

Ημερομηνία Παράδοσης: 8 / 4 / 2022

Επικοινωνία: xsk@aueb.gr

Άσκηση 1η (Παραδοτέα: App1.java και App1_2.java) – 1 μονάδα

Να γραφεί μια εφαρμογή η οποία:

1. Διαβάζει από το χρήστη ένα θετικό ακέραιο ή μηδέν.
2. Υπολογίζει το παραγοντικό του αριθμού αυτού, με χρήση **μεθόδου** την οποία πρέπει να γράψετε.
3. Τέλος εμφανίζει το παραγοντικό.

Ο υπολογισμός του παραγοντικού του αριθμού n γίνεται ως εξής:

$$n! = n \times (n-1) \times (n-2) \times (n-3) \times \cdots \times 3 \times 2 \times 1$$

Για παράδειγμα, το παραγοντικό του 6 είναι:

$$6! = 1*2*3*4*5*6 = 720$$

Το παραγοντικό του μηδέν είναι 1.

Η κλήση μιας μεθόδου στη java, μπορεί να γίνει είτε μέσω αντικειμένου (instance method) είτε χωρίς τη χρήση αντικειμένου (static method).

Α. Συμπληρώστε τον κώδικα java στο αρχείο **App1.java** έτσι ώστε η **κλήση** της μεθόδου (στη main) να γίνεται **μέσω αντικειμένου.**

Β. Συμπληρώστε τον κώδικα java στο αρχείο **App1_2.java** έτσι ώστε η **κλήση** της μεθόδου (στη main) να γίνεται στατικά, **χωρίς τη δημιουργία αντικειμένου.**

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

Γράψτε μια εφαρμογή που να διαβάζει θετικούς και αρνητικούς ακεραίους (διάφορους του μηδενός). Το πρόγραμμα θα πρέπει να εμφανίζει το πλήθος των ακεραίων που έχει διαβάσει (**Items**), το μέσο όρο τους (**Average**), το πλήθος των αρνητικών ακεραίων (**Negative**), το πλήθος των θετικών ακεραίων (**Positive**), το μέγιστο ακέραιο (**Max**) και τον ελάχιστο ακέραιο (**Min**).

Η εφαρμογή θα τερματίζει την ανάγνωση των ακεραίων και θα εμφανίζει τα ζητούμενα αποτελέσματα, όταν ο χρήστης πληκτρολογήσει το μηδέν.

Αν τα δοκιμαστικά δεδομένα (είσοδος) της εφαρμογής είναι οι αριθμοί:

```
Give a number: -125
Give a number: 125
Give a number: 10
Give a number: 11
Give a number: 0
```

τότε τα αναμενόμενα αποτελέσματα (έξοδος) θα πρέπει να εμφανίζονται με την παρακάτω μορφή[1]:

```
----------------------------------------
Items   :      4
Average :      5,250
Negative:      1
Positive:      3
Max     :    125
Min     :   -125
----------------------------------------
```

Συμπληρώστε τον κώδικα java στο αρχείο **App2.java**.

[1] Θεωρείστε ότι το μέγιστο πλάτος των ακεραίων αριθμών είναι 7 ψηφία και των πραγματικών 7 ψηφία ακέραιο μέρος συν 3 δεκαδικά ψηφία έτσι ώστε όλοι οι αριθμοί να στοιχίζονται στις μονάδες.

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

Η βασική δευτεροβάθμια εξίσωση:

$$ax^2 + bx + c = 0$$

έχει δύο λύσεις που δίνονται από τη σχέση:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Η πρώτη λύση προκύπτει από τη χρήση του $+$ στη θέση του $\pm$ και η δεύτερη από τη χρήση του $-$.

Γράψτε μια εφαρμογή η οποία να δέχεται τιμές για τα a, b και c (`floating-point numbers`) και στη συνέχεια να υπολογίζει τις δύο λύσεις. Αν η υπόρριζη ποσότητα είναι αρνητικός αριθμός, η εξίσωση δεν έχει πραγματικές ρίζες και το πρόγραμμα σας θα πρέπει να εμφανίζει ένα σχετικό μήνυμα. Μπορείτε να υποθέσετε ότι η τιμή του a είναι μη μηδενική.

Έτσι, αν τα δοκιμαστικά δεδομένα (είσοδος) της εφαρμογής είναι οι αριθμοί:

1,1 10,5 2,8

τότε τα αναμενόμενα αποτελέσματα (έξοδος) θα πρέπει να εμφανίζονται με την παρακάτω μορφή[2]:

```
Enter the first number: 1,1
Enter the second number: 10,5
Enter the third number: 2,8
The first solution is :      -0,275
The second solution is:      -9,271
```

Αν τα δοκιμαστικά δεδομένα (είσοδος) της εφαρμογής είναι οι αριθμοί:

1,0 1,0 10,0

τότε τα αναμενόμενα αποτελέσματα (έξοδος) θα πρέπει να είναι:

```
This program reads three numbers and returns the solution
of the quadratic equation with these numbers as coefficients.
Enter the first number: 1,0
Enter the second number: 1,0
Enter the third number: 10,0
There are no real values for the quadratic equation.
```

Συμπληρώστε τον κώδικα java στο αρχείο **App3.java**.

[2] Θεωρείστε ότι το μέγιστο πλάτος πραγματικών αριθμών που αποτελεί την έξοδο του προγράμματος είναι 7 ψηφία ακέραιο μέρος συν 3 δεκαδικά ψηφία.

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

Γράψτε μια εφαρμογή η οποία να δέχεται ως **παράμετρο κατά την κλήση** της έναν ακέραιο αριθμό και ελέγχει αν ο αριθμός αυτός ανήκει ή όχι στην ακολουθία `Fibonacci`.

Εξ ορισμού, οι πρώτοι δύο αριθμοί `Fibonacci` είναι το 0 και το 1, και κάθε επόμενος αριθμός είναι το άθροισμα των δύο προηγούμενων.

$$F_n = F_{n-1} + F_{n-2}$$
με $F_0 = 0$ και $F_1 = 1$

Δίνεται το παρακάτω παράδειγμα εκτέλεσης της εφαρμογής με δεδομένο εισόδου τον αριθμό 121 – το δεδομένο δίνεται ως όρισμα (argument) της κύριας μεθόδου (main) κατά την κλήση της:

```
C:\Users>java App4 121
Fibonacci number = 1
Fibonacci number = 2
Fibonacci number = 3
Fibonacci number = 5
Fibonacci number = 8
Fibonacci number = 13
Fibonacci number = 21
Fibonacci number = 34
Fibonacci number = 55
Fibonacci number = 89
Fibonacci number = 144
121 is not a fibonacci number
```

Δίνεται το παρακάτω παράδειγμα εκτέλεσης της εφαρμογής με δεδομένο εισόδου τον αριθμό 8:

```
C:\Users>java App4 8
Fibonacci number = 1
Fibonacci number = 2
Fibonacci number = 3
Fibonacci number = 5
Fibonacci number = 8
8 is a fibonacci number
```

Συμπληρώστε τον κώδικα java στο αρχείο **App4.java**.

Άσκηση 5η (Παραδοτέο: Clock.java) – 2 μονάδες

Στην άσκηση αυτή θα κατασκευάσετε ένα ψηφιακό ρολόι. Το ρολόι θα αναπαρίσταται στο πρόγραμμά σας ως ένα αντικείμενο της τάξης **`Clock`** την οποία θα πρέπει πρώτα να ορίσετε και υλοποιήσετε σύμφωνα με τις παρακάτω προδιαγραφές:

1. Τα **δεδομένα** της τάξης **`Clock`** είναι τρεις ακέραιοι αριθμοί που παριστάνουν την ώρα, τα λεπτά και τα δευτερόλεπτα. Η πρόσβαση στα δεδομένα πρέπει να γίνεται μέσω των κατάλληλων μεθόδων. Αναλυτικότερα:

2. Οι **μέθοδοι** της τάξης **`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 τάξη και επιπλέον μεθόδους.

Συμπληρώστε τον κώδικα java στο αρχείο **Clock.java.**

Άσκηση 6η (Παραδοτέο: clockApp.java) – 1 μονάδα

Θα πρέπει να υλοποιήσετε ένα κυρίως πρόγραμμα το οποίο θα δημιουργεί ένα ψηφιακό ρολόι ως ένα αντικείμενο της τάξη Clock 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.

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.

Έτσι το πρόγραμμα θα εμφανίζει:

```
16:28:58
16:28:59
16:29:00
16:29:01
```

κλπ

Συμπληρώστε τον κώδικα java στο αρχείο clockApp.java.

Άσκηση 7η (Παραδοτέο: Account.java) – 2 μονάδες

Στην άσκηση αυτή θα κατασκευάσετε μια τραπεζική εφαρμογή.

Για το λόγο αυτό πρέπει, αρχικά, να ορίσετε την τάξη **`Account`** (τραπεζικός λογαριασμός) σύμφωνα με τις παρακάτω προδιαγραφές:

1. Τα **δεδομένα** της τάξης **`Account`** είναι το επιτόκιο **RATE** με βάση το οποίο υπολογίζεται ο τόπος της περιόδου, το όνομα **Name** του πελάτη, ο αριθμός λογαριασμού **acctNumber** (τύπου String) και το υπόλοιπο (**Balance**) του λογαριασμού σε ευρώ. Υποθέστε ότι το επιτόκιο RATE είναι σταθερό 1,5% για όλους τους λογαριασμούς και για όλη τη χρονική περίοδο. Για το λόγο αυτό ορίστε το RATE ως τελική (final) μεταβλητή.

2. Η τάξη **`Account`** πρέπει να διαθέτει **δύο μεθόδους κατασκευαστές**:

Ο πρώτος κατασκευάζει-αρχικοποιεί αντικείμενα με παράμετρο το όνομα του πελάτη, τον αριθμό λογαριασμού και το αρχικό υπόλοιπο.

Ο δεύτερος κατασκευάζει-αρχικοποιεί αντικείμενα με παράμετρο το όνομα του πελάτη, τον αριθμό λογαριασμού. Στην περίπτωση αυτή ο κατασκευαστής θα πρέπει να θέτει στο υπόλοιπο την τιμή μηδέν.

3. Οι **μέθοδοι-πράξεις** της τάξης **Account** πρέπει να είναι οι εξής:

```
double deposit (double amount){
        // Μέθοδος που εκτελεί μια κατάθεση στον τραπεζικό λογαριασμό.
        // Το ποσό της κατάθεσης δίνεται ως παράμετρος.
        Εμφανίζει το ποσό της κατάθεσης.
        Εφόσον το ποσό της κατάθεσης είναι θετικό, αυξάνει
        το υπόλοιπο του λογαριασμού κατά το ποσό της κατάθεσης.
        Επιστρέφει το νέο υπόλοιπο.
}

double withdraw (double amount){
        // Μέθοδος που εκτελεί μια ανάληψη από τον τραπεζικό λογαριασμό.
        // Το ποσό της ανάληψης δίνεται ως παράμετρος.
        Εμφανίζει το ποσό της ανάληψης.
        Εφόσον το ποσό της ανάληψης είναι θετικό,
        και εφόσον το υπόλοιπο είναι μεγαλύτερο του ποσού ανάληψης,
        μειώνει το υπόλοιπο του λογαριασμού κατά το ποσό της ανάληψης.
        Επιστρέφει το νέο υπόλοιπο.
 }

double addInterest (){
        Προσθέτει τον τόκο στο λογαριασμό (αυξάνει το υπόλοιπο),
        με βάση το επιτόκιο (RATE) και επιστρέφει το νέο
        υπόλοιπο του λογαριασμού.
}

double getBalance (){
        Επιστρέφει το τρέχον υπόλοιπο του λογαριασμού.
}

String getAccountNumber (){
        Επιστρέφει τον αριθμό λογαριασμού.
}

public String toString(){
        Επιστρέφει το όνομα πελάτη, τον αριθμό λογαριασμού και το υπόλοιπο.
}
```

Συμπληρώστε τον κώδικα java στο αρχείο **Account.java.**

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

Δημιουργείστε μια εφαρμογή που δημιουργεί τρεις τραπεζικούς λογαριασμούς και εκτελεί τις παρακάτω συναλλαγές με τη βοήθεια της class Acount που αναπτύξατε στην προηγούμενη άσκηση. Δείτε παρακάτω τις πράξεις που πρέπει να κάνει η εφαρμογή σας καθώς και τα αποτελέσματα που πρέπει να εμφανίζει:

1. Δημιουργεί και εμφανίζει τους τραπεζικούς λογαριασμούς:

```
Account Number: 100-00
Nane: Togantzi Maria
Balance: 188,46

Account Number: 100-01
Nane: Kalergis Christos
Balance: 140,21

Account Number: 100-02
Nane: Maras Petros
Balance: 0,00
```

2. Κατάθεση αρνητικού ποσού (-10,00) στο λογαριασμό 100-00:

```
Deposit @ Account 100-00
Balance 188,46
Requested: -10.0
Error: Deposit amount is invalid.
New Balance 188,46
```

3. Κατάθεση θετικού ποσού (140,21) στο λογαριασμό 100-01:

```
Deposit @ Account 100-01
Balance 140,21
Requested: 500.1
New Balance 640,31
```

4. Ανάληψη ποσού (1420,75) που ξεπερνά το υπόλοιπο του λογαριασμού 100-02:

```
withDraw @ Account 100-02
Balance 0,00
Requested: 1420.75
Error: Insufficient funds.
New Balance 0,00
```

5. Ανάληψη αρνητικού ποσού (-10,00) από το λογαριασμό 100-02:

```
withDraw @ Account 100-02
Balance 0,00
Requested: -10.0
Error: Withdraw amount is invalid.
New Balance 0,00
```

6. Ανάληψη ποσού (420.75) από το λογαριασμό 100-02:

```
withDraw @ Account 100-02
Balance 0,00
Requested: 420.75
Error: Insufficient funds.
New Balance 0,00
```

7. Προσθέτει τον τόκο σε όλους τους λογαριασμούς

```
add interest ...
```

8. Εμφανίζει όλους τους λογαριασμούς μετά τον τοκισμό με τα νέα υπόλοιπά τους.

```
Account Number: 100-00
Nane: Togantzi Maria
Balance: 191,29

Account Number: 100-01
Nane: Kalergis Christos
Balance: 649,91

Account Number: 100-02
Nane: Maras Petros
Balance: 0,00
```

Συμπληρώστε τον κώδικα java στο αρχείο **bankApp.java.**

Επιπλέον Γενικές Οδηγίες:

1. Η άσκηση είναι ατομική.
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 (1η Σειρά Ασκήσεων), **αφού πρώτα τα συμπιέσετε σε ένα αρχείο με όνομα τον αριθμό του φοιτητικού σας μητρώου, για παράδειγμα 3210000.zip**

Καλή Επιτυχία!