Οικονομικό Πανεπιστήμιο Αθηνών
Τμήμα Πληροφορικής

3222: Προγραμματισμός Υπολογιστών με Java

(Εαρινό εξάμηνο 2023 – 2024)

1η Σειρά Ασκήσεων

Ημερομηνία Παράδοσης: 8 Απριλίου 2024 (ώρα 23:59)

Άσκηση 1η (Παραδοτέα: App1.java και App2.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 (κλήση) κατάλληλης **μεθόδου** την οποία πρέπει να γράψετε. Ο αριθμός δίνεται ως παράμετρος της μεθόδου. Η μέθοδος επιστρέφει το παραγοντικό του αριθμού.
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.

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 **App2.java** έτσι ώστε η ***κλήση*** της μεθόδου (στη main) να γίνεται στατικά, **χωρίς τη δημιουργία αντικειμένου.**

Άσκηση 2η (Παραδοτέο: App3.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: 1250
Give a number: 10
Give a number: 11
Give a number: 0
```

τότε τα αποτελέσματα (έξοδος) **θα πρέπει να εμφανίζονται με την παρακάτω μορφή** (με διαχωριστικό χιλιάδων, τρία δεκαδικά ψηφία για τους δεκαδικούς αριθμούς και στοίχιση στις μονάδες), ως εξής:

```
Items   :       4
Average :     286,500
Negative:       1
Positive:       3
Max     :   1.250
Min     :    -125
```

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

Άσκηση 3η (Παραδοτέο: App4.java) – 2 μονάδες.

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 (λύσεις) θα πρέπει να εμφανίζονται τρία δεκαδικά ψηφία, ως εξής:

```
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 αποτελέσματα (έξοδος) θα πρέπει να είναι:

```
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 **App4.java**.

Άσκηση 4η (Παραδοτέο: App5.java) – 2 μονάδες.

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 App5 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 επίσης, το παρακάτω παράδειγμα εκτέλεσης της εφαρμογής με δεδομένο εισόδου τον αριθμό 8:

```
C:\Users>java App5 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 **App5.java**.

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

Ορίστε την κλάση **Account** (τραπεζικός λογαριασμός) ως εξής:

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

2. Η κλάση **Account** πρέπει να διαθέτει έναν **κατασκευαστή** με παραμέτρους το όνομα του πελάτη, τον αριθμό λογαριασμού και το αρχικό υπόλοιπο.

3. Οι **μέθοδοι** της κλάσης **Account** πρέπει να είναι οι εξής:

```
/*      Deposit: Μέθοδος που εκτελεί μια κατάθεση στον τραπεζικό
        λογαριασμό. Το ποσό της κατάθεσης δίνεται ως παράμετρος.
*/

double Deposit (double Amount){
        Εφόσον το ποσό της κατάθεσης είναι μεγαλύτερο από το μηδέν,
        αυξάνει το υπόλοιπο του λογαριασμού κατά το ποσό της κατάθεσης,
        διαφορετικά εμφανίζει ένα κατάλληλο μήνυμα.
        Επιστρέφει το υπόλοιπο του λογαριασμού.
}
```

```
/*      Withdraw: Μέθοδος που εκτελεί μια ανάληψη από τον τραπεζικό
        λογαριασμό. Το ποσό της ανάληψης δίνεται ως παράμετρος.
*/

double Withdraw (double Amount){
        Εφόσον το ποσό της ανάληψης είναι μεγαλύτερο από το μηδέν,
        και εφόσον το υπόλοιπο είναι μεγαλύτερο του ποσού ανάληψης,
        μειώνει το υπόλοιπο του λογαριασμού κατά το ποσό της ανάληψης
        διαφορετικά εμφανίζει ένα κατάλληλο μήνυμα.
        Επιστρέφει το υπόλοιπο του λογαριασμού.
}
```

```
double addInterest (){
```
Προσθέτει τον τόκο στο υπόλοιπο του λογαριασμού
(με βάση το επιτόκιο για μια περίοδο).
Επιστρέφει το υπόλοιπο του λογαριασμού.
```
}
```

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

```
public String toString(){
```
Επιστρέφει μια συμβολοσειρά με τα στοιχεία του λογαριασμού όπως ακριβώς φαίνεται στο παρακάτω παράδειγμα (προσέξτε τη μορφοποίηση του υπολοίπου με δύο δεκαδικά και διαχωριστικό χιλιάδων):

```
Account Number : GR1001
Customer Name  : KALERGIS CHRISTOS
Balance        : 1.500,00

}
```

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

Άσκηση 6η (Παραδοτέο: App6.java) – 2 μονάδες.

Γράψτε μια **εφαρμογή** που **αρχικά δημιουργεί έναν τραπεζικό λογαριασμό**, ως ένα αντικείμενο της κλάσης **Account** (άσκηση 5). Τα στοιχεία του λογαριασμού (αριθμό λογαριασμού, όνομα πελάτη και αρχικό υπόλοιπο) τα δίνει ο χρήστης, για παράδειγμα:

```
Creating an account
Account Number  : GR100
Customer Name   : TOGANTZH MARIA
Initial balance : 1000
```

Στη συνέχεια εμφανίζει μια **λίστα επιλογών** (menu) μέσω της οποίας ο χρήστης επιλέγει μια από τις παρακάτω λειτουργίες (κατάθεση, ανάληψη, εμφάνιση στοιχείων λογαριασμού, προσθήκη τόκου). Όλες οι λειτουργίες αφορούν τον τραπεζικό λογαριασμό που έχετε ήδη δημιουργήσει:

```
---------------
1. Deposit
2. Withdraw
3. Account Details
4. Add Interest
0. Exit
---------------
>
```

Η **υλοποίηση** των παραπάνω λειτουργιών πρέπει να γίνεται από το πρόγραμμα με **χρήση (κλήση)** των αντίστοιχων **μεθόδων** της κλάσης **Account** (άσκηση 5).

Ακολουθούν παραδείγματα χρήσης του προγράμματος (η είσοδος του χρήστη φαίνεται κάθε φορά με έντονη γραμματοσειρά):

Περίπτωση 1: ο χρήστης επιλέγει κατάθεση και δίνει ένα θετικό ποσό, για παράδειγμα:

```
---------------
1. Deposit
2. Withdraw
3. Account Details
4. Add Interest
0. Exit
---------------
> 1
Deposit ...
Account Number : GR100
Customer Name  : TOGANTZH MARIA
Balance        : 1.000,00
Amount: 100
New Balance    : 1.100,00
```

Περίπτωση 2: ο χρήστης επιλέγει κατάθεση και δίνει ένα αρνητικό ποσό ή μηδέν, για παράδειγμα:

```
---------------
1. Deposit
2. Withdraw
3. Account Details
4. Add Interest
0. Exit
---------------
> 1
Deposit ...
Account Number : GR100
Customer Name  : TOGANTZH MARIA
Balance        : 1.100,00
Amount: -100
Error: Deposit Amount is invalid
New Balance    : 1.100,00
```

Περίπτωση 3: ο χρήστης επιλέγει ανάληψη και δίνει ένα θετικό ποσό, για παράδειγμα:

```
---------------
1. Deposit
2. Withdraw
3. Account Details
4. Add Interest
0. Exit
---------------
> 2
Withdraw ...
Account Number : GR100
Customer Name  : TOGANTZH MARIA
Balance        : 1.100,00
Amount: 80
New Balance    : 1.020,00
```

Περίπτωση 4: ο χρήστης επιλέγει ανάληψη και δίνει ένα αρνητικό ποσό ή μηδέν, για παράδειγμα:

```
---------------
1. Deposit
2. Withdraw
3. Account Details
4. Add Interest
0. Exit
---------------
> 2
Withdraw ...
Account Number : GR100
Customer Name  : TOGANTZH MARIA
Balance        : 1.020,00
Amount: -100
Error: Withdraw Amount is invalid
New Balance    : 1.020,00
```

Περίπτωση 5: ο χρήστης επιλέγει ανάληψη και δίνει ποσό ανάληψης που είναι μεγαλύτερο από το διαθέσιμο υπόλοιπο του λογαριασμού, για παράδειγμα:

```
---------------
1. Deposit
2. Withdraw
3. Account Details
4. Add Interest
0. Exit
---------------
> 2
Withdraw ...
Account Number : GR100
Customer Name  : TOGANTZH MARIA
Balance        : 1.020,00
Amount: 1500
Error: Insufficient funds
New Balance    : 1.020,00
```

Περίπτωση 6: ο χρήστης επιλέγει την προβολή των λεπτομερειών (στοιχείων) του λογαριασμού:

```
---------------
1. Deposit
2. Withdraw
3. Account Details
4. Add Interest
0. Exit
---------------
> 3
Account Number : GR100
Customer Name  : TOGANTZH MARIA
Balance        : 1.020,00
```

Περίπτωση 7: ο χρήστης επιλέγει τον υπολογισμό και την προσθήκη του τόκου στο υπόλοιπο του λογαριασμού:

```
---------------
1. Deposit
2. Withdraw
3. Account Details
4. Add Interest
0. Exit
---------------
> 4
Add Interest
Account Number : GR100
Customer Name  : TOGANTZH MARIA
Balance        : 1.020,00
New Balance    : 1.035,30
```

Σημαντική παρατήρηση: Η λειτουργία του προγράμματος και η είσοδος/έξοδος πρέπει να γίνεται όπως εμφανίζεται παραπάνω (περιπτώσεις 1-7). Η μορφή των αποτελεσμάτων (έξοδος) του προγράμματος πρέπει να γίνεται όπως εμφανίζεται στα παραδείγματα παραπάνω (προσέξτε ότι το υπόλοιπο του λογαριασμού εμφανίζεται με δύο δεκαδικά ψηφία και διαχωριστικό χιλιάδων).

Συμπληρώστε τον κώδικα java στο αρχείο **App6.java**

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. Κάθε πρόγραμμα σας πρέπει να μεταγλωττίζεται και να εκτελείται στη **γραμμή εντολών** (όχι μέσω κάποιου IDE).
5. 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
6. Για τις τυχόν απορίες σας μπορείτε να επικοινωνήσετε με την κ. Μαρία Τογαντζή (mst@aueb.gr) και τον κ. Χρήστο Καλέργη (xsk@aueb.gr).

Καλή Επιτυχία!