Προγραμματιστικές Ασκήσεις, Εργαστήριο 1
Να ολοκληρωθούν μέχρι τη λήξη του εργαστηρίου

1. Γράψτε, μεταγλωττίστε και τρέξτε το παρακάτω πρόγραμμα το οποίο λύνει την πρωτοβάθμια εξίσωση $2x + 4 = 0$:

```
class LinearEquation1 {
      public static void main( String args [ ] ) {
           int alpha = 2;
           int beta = 4;
           double x = -beta/alpha;
          System.out.println ( "The solution  to equation 2x+4 = 0 is x = " + x ) ;
      }
}
```

2. Τροποποιήστε το πρόγραμμα LinearEquation1 από το πρώτο ερώτημα ώστε να λύνει μια οποιαδήποτε πρωτοβάθμια εξίσωση $\alpha x + \beta = 0$, με ακέραιους συντελεστές. Θα ονομάσετε το νέο πρόγραμμα LinearEquation2. Το πρόγραμμα σας θα διαβάζει τους αριθμούς alpha και beta από την είσοδο. Επειδή πλέον μπορούμε να έχουμε οποιεσδήποτε τιμές στα alpha και beta, πρέπει να ελέγξουμε τις ειδικές περιπτώσεις όπου το alpha ή/και το beta είναι μηδέν. Αν και το alpha και το beta είναι μηδενικά θα πρέπει να τυπώσει *Infinite Solutions*. Αν το alpha είναι μηδέν και το beta είναι μη μηδενικό, θα τυπώσει *No Solution*. Αλλιώς το πρόγραμμα θα υπολογίζει και θα εκτυπώνει την ρίζα της εξίσωσης.

Παρακάτω είναι τέσσερα παραδείγματα για το πώς πρέπει να συμπεριφέρεται το πρόγραμμα σας:

```
>java LinearEquation2
Give alpha and beta
2 4
The solution to the equation 2x+4=0 is x=-2
```

```
>java LinearEquation2
Give alpha and beta
2 5
The solution to the equation 2x+5=0 is x=-2.5
```

```
>java LinearEquation2
Give alpha and beta
0 5
No Solution
```

```
>java LinearEquation2
Give alpha and beta
0 0
Infinite solutions
```

3. Τροποποιήστε το πρόγραμμα LinearEquation2 από το δεύτερο ερώτημα ώστε να μπορούμε να λύσουμε πολλαπλές πρωτοβάθμιες εξισώσεις. Θα ονομάσετε το νέο πρόγραμμα LinearEquation3. Το πρόγραμμα θα ρωτάει τον χρήστη αν θέλει να λύσει μια πρωτοβάθμια εξίσωση και αν απαντήσει yes θα συνεχίζει όπως στην LinearEquation2. Η διαδικασία αυτή θα συνεχίζεται μέχρι ο χρήστης να απαντήσει no.

Παρακάτω είναι ένα δείγμα για το πώς πρέπει να συμπεριφέρεται το πρόγραμμα σας:

```
>java LinearEquation3
Do you want to solve a linear equation?
yes
Give alpha and beta
5 2
The solution to the equation 5x+4=0 is x=-0.4
Do you want to solve a linear equation?
no
```

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

- **Στοίχιση κώδικα:** Ο κώδικας πρέπει να είναι σωστά στοιχισμένος αλλιώς αφαιρείται ένα 20% του βαθμού.
- **Turnin**: Κάντε turnin τα προγράμματα σας στο lab1@ply212.

π.χ. turnin lab1@ply212 <όνομα αρχείου>

Στον κώδικα να αναγράφονται σε σχόλια το όνομα, το login και ο ΑΜ σας. Χρησιμοποιείτε **μόνο** λατινικούς χαρακτήρες αλλιώς δεν λειτουργεί το turnin.

Βαθμολόγηση:

Άσκηση 1: 2μ (1,5μ αν τρέχει και 0,5 μ για τη στοίχιση. Αν δεν τρέχει 0μ.)

Άσκηση 2: 5μ (1,5μ για κάθε περίπτωση που περιγράφεται στην οθόνη (για την τρίτη περίπτωση 0.5 αν δουλεύει για ακέραιο αποτέλεσμα, +1 αν έχουν το casting), και 0,5μ για τη στοίχιση.)

Άσκηση 3: 3μ (1.5μ αν δουλεύει η διαδικασία ερώτηση-απάντηση έστω και μία φορά, 1,μ αν γίνεται σωστά η επανάληψη και 0,5μ για τη στοίχιση.)

Προγραμματιστικές Ασκήσεις, Εργαστήριο 2
Να ολοκληρωθούν μέχρι τη λήξη του εργαστηρίου

1. Γράψετε ένα πρόγραμμα **UpperDiagonal** το οποίο να δημιουργεί και να δίνει τιμές σε έναν άνω τριγωνικό πίνακα. Το πρόγραμμά σας θα ζητάει από τον χρήστη το μέγεθος του πίνακα, θα δημιουργεί τον άνω τριγωνικό πίνακα και θα του δίνει τιμές ξεκινώντας από το 0 από αριστερά προς τα δεξιά και πάνω προς τα κάτω. Τέλος, θα διατρέχει και θα τυπώνει τις τιμές του πίνακα.

   Παρακάτω είναι ένα παράδειγμα για το πώς πρέπει να συμπεριφέρεται το πρόγραμμά σας. Στο παράδειγμα φαίνεται και ποια θα πρέπει να είναι η μορφή του πίνακα που θα δημιουργήσετε και ποιες οι τιμές που θα πρέπει να έχει ο πίνακας.

```
>java UpperDiagonal
Give the matrix size
3
The upper diagonal array of size 3 is:
0 1 2
3 4
5
```

   Προσοχή: Ο στόχος της άσκησης δεν είναι να τυπώσετε αυτές τις τιμές, αλλά να δημιουργήσετε ένα δισδιάστατο πίνακα που θα τις κρατάει. Η απλή εκτύπωση δεν θα πάρει βαθμούς.

2. Δημιουργήστε μια κλάση **Timer** η οποία υλοποιεί ένα χρονόμετρο που μετράει λεπτά και ώρες. Η κλάση θα έχει ένα πεδίο για τα λεπτά και ένα για τις ώρες τα οποία αρχικοποιούνται στο μηδέν. Έχει επίσης δύο μεθόδους: (α) Την μέθοδο **advance** η οποία προχωράει το χρονόμετρο κατά ένα λεπτό, και (β) την μέθοδο **printElapsedTime** η οποία τυπώνει τον χρόνο που έχει περάσει (δείτε το παράδειγμα για το πώς θα πρέπει να είναι η εκτύπωση).

   Δημιουργήστε επίσης μια κλάση **CountTime** η οποία θα έχει την **main**. Μέσα στη main θα δημιουργήσετε ένα αντικείμενο Timer. Ζητήσετε από τον χρήστη να σας δώσει τον αριθμό των λεπτών που θέλει να μετρήσει, προχωρήσετε το χρονόμετρο αντίστοιχα, και θα τυπώσετε στο τέλος τον χρόνο που έχει περάσει. Παρακάτω είναι ένα δείγμα για το πώς πρέπει να συμπεριφέρεται το πρόγραμμα σας:

```
>java CountTime
Give the number of minutes you want to count
90
Elapsed time: 1 hour(s) and 30 minute(s)
```

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

- **Στοίχιση κώδικα:** Ο κώδικας πρέπει να είναι σωστά στοιχισμένος αλλιώς αφαιρείται ένα 20% του βαθμού.
- **Turnin**: Τα Group 1 και 3 θα κάνουν turnin τα προγράμματα στο lab2g13@ply212.
  Τα Group 2 και 4 θα κάνουν turnin τα προγράμματα στο lab2g24@ply212.

  π.χ. turnin **lab2g13**@ply212 UpperDiagonal.java CountTime.java
  turnin **lab2g24**@ply212 UpperDiagonal.java CountTime.java

  Στον κώδικα να αναγράφονται σε σχόλια το όνομα, το login και ο ΑΜ σας. Χρησιμοποιείτε **μόνο** λατινικούς χαρακτήρες αλλιώς δεν λειτουργεί το turnin.

Προγραμματιστικές Ασκήσεις, Εργαστήριο 3
Να ολοκληρωθούν μέχρι τη λήξη του εργαστηρίου

Ένα *ιστόγραμμα* τιμών μετράει για ένα σύνολο από τιμές πόσες φορές εμφανίστηκε η κάθε τιμή. Για παράδειγμα αν έχω τις τιμές: 1,2,1,2,4,5,3,3,3,2,4 το ιστόγραμμα τους είναι 2,3,3,2,1, και είναι ο αριθμός εμφανίσεων των τιμών 1,2,3,4,5 αντίστοιχα (η τιμή 1 εμφανίζεται 2 φορές, η τιμή 2, 3 φορές, κοκ).

Στην άσκηση αυτή θα υλοποιήσετε μια κλάση **GradeHistogram** η οποία κρατάει ένα ιστόγραμμα για τους βαθμούς ενός μαθήματος. Η κλάση σας θα πρέπει να κρατάει το μέγιστο βαθμό για το μάθημα, και ένα πίνακα τον αριθμό εμφανίσεων του κάθε βαθμού. Αν ο μέγιστος βαθμός είναι maxGrade τότε οι πιθανοί βαθμοί θα είναι όλοι οι ακέραιοι στο διάστημα [1,maxGrade]. Η κλάση θα πρέπει να έχει και τις εξής μεθόδους:

1. Ένα **constructor,** ο οποίος θα παίρνει σαν όρισμα τον μέγιστο βαθμό και ένα πίνακα με βαθμούς και δημιουργεί το ιστόγραμμα των βαθμών.
2. Μια μέθοδο **toString**, η οποία θα επιστρέφει ένα String που αναπαριστά το ιστόγραμμα. Για το ιστόγραμμα στο παραπάνω παράδειγμα θα επιστρέφει το String: «1:2 2:3 3:3 4:2 5:1».
3. Την μέθοδο **equals**, η οποία θα συγκρίνει αν δύο ιστογράμματα είναι ίδια.
4. Μια μέθοδο **addHistogram** η οποία παίρνει σαν όρισμα ένα άλλο ιστόγραμμα (ένα αντικείμενο τύπου **GradeHistogram**) και, εφόσον έχουν τον ίδιο μέγιστο βαθμό, το προσθέτει στο υπάρχον ιστόγραμμα.

Σας δίνεται η κλάση **GradeHistogramTest,** για να τεστάρετε την κλάση σας. Όταν υλοποιήσετε τις μεθόδους που καλούνται στην main, βγάλετε τα σχόλια από τις αντίστοιχες εντολές για να τεστάρετε τις μεθόδους. Τεστάρετε τον κώδικα σας σταδιακά όπως φαίνεται στα σχόλια.

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

- **Στοίχιση κώδικα:** Ο κώδικας πρέπει να είναι σωστά στοιχισμένος αλλιώς αφαιρείται ένα 20% του βαθμού.
- Πριν ξεκινήσετε την υλοποίηση σκεφτείτε τι πεδία θέλετε να κρατάει η κλάση σας και πως θα τα χρησιμοποιήσετε.
- **Πεδία:** Όλα τα πεδία θα πρέπει να ορίζονται **private.**
- Κάντε turnin τα προγράμματα σας στο lab3@ply212.

  π.χ. turnin lab3@ply212 Geometric.java

  Στον κώδικα να αναγράφονται σε σχόλια τα ονόματά σας και ο ΑΜ σας.

Προγραμματιστική Άσκηση, Εργαστήριο 4

Στην άσκηση αυτή θα υλοποιήσετε ένα πρόγραμμα που διαχειρίζεται ένα τραπεζικό λογαριασμό και επιταγές. Το πρόγραμμα σας θα έχει δύο βασικές κλάσεις: την κλάση Check για τις επιταγές, και την κλάση Account για τον λογαριασμό της τράπεζας.

Η κλάση **Check**, έχει δύο βασικά πεδία: το όνομα του παραλήπτη και το ποσό της επιταγής. Και τα δύο πεδία αρχικοποιούνται στον **constructor** της μεθόδου. Ορίστε και τις κατάλληλες μεθόδους πρόσβασης (**accessor methods**).

Η κλάση **Account** έχει επίσης δύο βασικά πεδία: το ποσό του λογαριασμού (το οποίο είναι ένας double) και το όνομα του κατόχου του λογαριασμού. Επίσης έχει τις εξής μεθόδους:

1. Τον **constructor** της κλάσης ο οποίος παίρνει σαν όρισμα το όνομα του κατόχου. Το ποσό αρχικοποιείται στο μηδέν.
2. Τη μέθοδο **deposit** η οποία παίρνει σαν όρισμα ένα ποσό και το προσθέτει στο λογαριασμό.
3. Υπερφορτώστε την **deposit** ώστε να παίρνει όρισμα μια επιταγή και να προσθέτει το ποσό της επιταγής στον λογαριασμό, εφόσον η επιταγή έχει παραλήπτη τον κάτοχο του λογαριασμού. Επιστρέφει boolean τιμή αν έγινε σωστά η κατάθεση.
4. Τη μέθοδο **toString** η οποία επιστρέφει ένα String που αποτελείται από το όνομα του κατόχου και το ποσό χωρισμένα με ":".

Για να τεστάρετε την κλάση σας δημιουργείστε μία άλλη κλάση **TestAccount** και συμπληρώστε την main όπως φαίνεται παρακάτω.

```
class TestAccount
{
      public static void main(String[] args){
            // δημιουργήστε τον λογαριασμό myAccount στο όνομα σας
            // καταθέστε 100 ευρώ στον λογαριασμό σας
            // δημιουργήστε μια επιταγή στο όνομα σας για 100 ευρώ
            // καταθέστε την επιταγή στον λογαριασμό σας
            // θα εκτυπώνεται ένα μήνυμα αν έγινε σωστά η κατάθεση
            System.out.println(myAccount);
      }
}
```

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

- **Στοίχιση κώδικα:** Ο κώδικας πρέπει να είναι σωστά στοιχισμένος αλλιώς αφαιρείται ένα 20% του βαθμού.
- **Πεδία:** Όλα τα πεδία θα πρέπει να ορίζονται **private.**
- Κάντε turnin τα προγράμματα σας στο lab4g13@ply212.

  π.χ. turnin lab4g13@ply212 TestAccount.java

  Στον κώδικα να αναγράφονται σε σχόλια τα ονόματά σας και ο ΑΜ σας.

Προγραμματιστική Άσκηση, Εργαστήριο 5

Ο στόχος της άσκησης είναι να εξασκηθείτε με την δημιουργία **copy constructors** και την **φωλιασμένη κλήση** μεθόδων. Για την άσκηση αυτή θα δημιουργήσετε ένα πρόγραμμα που θα χειρίζεται τις επαφές ενός κινητού. Το πρόγραμμα σας θα έχει 2 βασικές κλάσεις:

1. Η πρώτη είναι η κλάση **Contact** η οποία κρατάει πληροφορίες για μία επαφή: το παρατσούκλι (**nickname**) με το οποίο είναι καταχωρημένη η επαφή, το όνομα (**name**) και το νούμερο (**number**).
2. Η δεύτερη είναι η κλάση **ContactList** που κρατάει πληροφορίες για μια λίστα από επαφές. Οι επαφές αποθηκεύονται σε ένα πίνακα. Το μέγεθος του πίνακα δίνεται στον constructor.

Σας δίνεται η κλάση **ListTest** η οποία περιέχει την μέθοδο main και η οποία προϋποθέτει τις παραπάνω δύο κλάσεις και κάποιες μεθόδους γι αυτές τις κλάσεις. Υλοποιήστε τις κλάσεις ώστε η main να λειτουργεί σωστά. Η έξοδος της θα πρέπει να είναι:

```
King's Slayer: Jamie Lanister - 999
Tywin: Tywin Lanister - 888
Tyrion: Tyrion Lanister - 777

King's Slayer: Jamie Lanister - 999
Tywin: Tywin Lanister - 888
The Imp: Tyrion Lanister - 777

The two lists are different
```

Υποδείξεις: Για την άσκηση αυτή θα πρέπει να σκεφτείτε μόνοι σας τι πεδία και τι μεθόδους θα ορίσετε. Προσέξτε τι σας ζητάει η main και σκεφτείτε ποια θα είναι τα απαραίτητα πεδία και μέθοδοι για την υλοποίηση σας. Μην υλοποιήσετε μεθόδους που δεν χρειάζονται. Υλοποιήστε φωλιασμένη κλήση μεθόδων όπου γίνεται, αλλιώς η λύση σας δεν θα θεωρηθεί σωστή.

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

- **Στοίχιση κώδικα:** Ο κώδικας πρέπει να είναι σωστά στοιχισμένος αλλιώς αφαιρείται ένα 20% του βαθμού.
- **Πεδία:** Όλα τα πεδία θα πρέπει να ορίζονται **private.**
- Κάντε turnin τα προγράμματα σας στο lab5g24@ply212.

  π.χ. turnin lab5g24@ply212 ContactList.java

  Στον κώδικα να αναγράφονται σε σχόλια τα ονόματά σας και ο ΑΜ σας.

Προγραμματιστική Άσκηση, Εργαστήριο 6

Στην άσκηση αυτή θα υλοποιήσετε ένα πρόγραμμα που διαχειρίζεται τις τραπεζικές συναλλαγές τριών ατόμων. Η υλοποίηση σας είναι μια πιο σύνθετη εκδοχή της υλοποίησης που κάνατε για το Εργαστήριο 4. Μπορείτε να χρησιμοποιήσετε κάποιο από τον κώδικα που κάνατε σε εκείνο το εργαστήριο, αλλά με προσοχή γιατί υπάρχουν αρκετές και σημαντικές αλλαγές.

Το πρόγραμμα σας θα έχει τρεις βασικές κλάσεις: την κλάση Check για τις επιταγές, και την κλάση Account για τον λογαριασμό της τράπεζας, και την κλάση Person για ένα άτομο.

Η κλάση **Check**, έχει τρία πεδία: το όνομα του παραλήπτη, το ποσό της επιταγής, και τον τραπεζικό λογαριασμό που εξέδωσε την επιταγή. Όλα τα πεδία αρχικοποιούνται στον **constructor** της μεθόδου. Ορίστε μεθόδους πρόσβασης (**accessor methods**) για όλα τα πεδία.

Η κλάση **Account** έχει επίσης τρία πεδία: το νούμερο του λογαριασμού, το ποσό των χρημάτων στον λογαριασμό (το οποίο είναι ένας double), και το άτομο Person που είναι ο ιδιοκτήτης του λογαριασμού. Επίσης έχει τις εξής μεθόδους:

5. Τον **constructor** της κλάσης ο οποίος παίρνει σαν ορίσματα τιμές και για τα τρία πεδία.
6. Μία μέθοδο πρόσβασης για το ποσό του λογαριασμού.
7. Την μέθοδο **withdraw** η οποία παίρνει σαν όρισμα ένα ποσό και το αφαιρεί από τον λογαριασμό. Επιστρέφει μια boolean τιμή αν η ανάληψη των χρημάτων έγινε επιτυχώς.
8. Την μέθοδο **issueCheck** η οποία παίρνει σαν όρισμα το όνομα του παραλήπτη και το ποσό και δημιουργεί επιστρέφει μια επιταγή από τον λογαριασμό για αυτό τον παραλήπτη.
9. Τη μέθοδο **deposit** η οποία παίρνει σαν όρισμα μια επιταγή και εφόσον η επιταγή έχει παραλήπτη τον κάτοχο του λογαριασμού, και η επιταγή καλύπτεται, εξαργυρώνει την επιταγή στον λογαριασμό. Επιστρέφει boolean τιμή αν έγινε σωστά η κατάθεση.

Η κλάση **Person** έχει τέσσερα πεδία: το όνομα και το ΑΦΜ του κατόχου, τον λογαριασμό (Account) του ατόμου, και ένα ArrayList checkList με όλες τις επιταγές (Check) που έχουν εκδοθεί για το άτομο αυτό. Επίσης έχει τις εξής μεθόδους:

1. Τον **constructor** της κλάσης ο οποίος παίρνει σαν ορίσματα το όνομα και το ΑΦΜ.
2. Μία μέθοδο πρόσβασης για το όνομα.
3. Την μέθοδο **openAccount** η οποία παίρνει σαν όρισμα ένα ποσό δημιουργεί τον λογαριασμό για το άτομο με νούμερο ίδιο με το ΑΦΜ, και αρχικό ποσό το όρισμα.
4. Την μέθοδο **issueCheck** η οποία παίρνει σαν όρισμα ένα άλλο άτομο και ένα ποσό και εκδίδει μια επιταγή για αυτό και το ενημερώνει κατάλληλα.
5. Την μέθοδο **depositCheck** η οποία παίρνει σαν όρισμα μια επιταγή την καταθέτει στον λογαριασμό του ατόμου. Επιστρέφει boolean τιμή αν έγινε σωστά η κατάθεση.
6. Την μέθοδο **printAssets** η οποία εκτυπώνει τα περιουσιακά στοιχεία του ατόμου. Συγκεκριμένα τυπώνει το όνομα και ΑΦΜ, το ποσό που υπάρχει στον λογαριασμό, και το (συνολικό) ποσό που έχει το άτομο σε επιταγές.

Σας δίνεται επίσης και η υλοποίηση της μεθόδου **depositAllChecks** (μέσα στο αρχείο **Person.java**) η οποία προσπαθεί να καταθέσει τις επιταγές στο λογαριασμό του πελάτη, και αφαιρεί όποια επιταγή κατατίθεται

επιτυχώς. Η μέθοδος υποθέτει ότι η λίστα σας λέγεται **checkList**. Χρησιμοποιεί ένα iterator για να αφαιρεί στοιχεία το οποίο δεν έχετε μάθει ακόμη.

Τέλος σας δίνεται ακόμη το αρχείο **Transactions.java** το οποίο περιέχει την main και μοντελοποιεί τις συναλλαγές τριών ατόμων, και η σωστή έξοδος στο **Output.txt**. Αυτό θα χρησιμοποιήσετε για να τεστάρετε τον κώδικα σας.

Υποδείξεις:

- Δημιουργήστε ένα ξεχωριστό αρχείο για κάθε κλάση.
- Δημιουργήστε πρώτα τα πεδία, τους constructors και τις μεθόδους πρόσβασης για κάθε κλάση.
- Προχωρήστε στην συνέχεια με την σειρά που δίνει τις κλάσεις η άσκηση (Check – Account – Person).
- Κάθε φορά που ολοκληρώνετε μια κλάση μπορείτε να την κάνετε compile για να βλέπετε αν έχετε συντακτικά λάθη.
- Μπορείτε να βάλετε σε σχόλια κομμάτια της main για να τεστάρετε αυτά που έχετε υλοποιήσει.

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

- **Στοίχιση κώδικα:** Ο κώδικας πρέπει να είναι σωστά στοιχισμένος αλλιώς αφαιρείται ένα 20% του βαθμού.
- **Πεδία:** Όλα τα πεδία θα πρέπει να ορίζονται **private.**
- Κάντε turnin τα προγράμματα σας στο lab6g13@ply212.

  π.χ. turnin lab6g13@ply212 Transactions.java

  Στον κώδικα να αναγράφονται σε σχόλια τα ονόματά σας και ο ΑΜ σας.

Προγραμματιστική Άσκηση, Εργαστήριο 7

ΕΡΩΤΗΣΗ 1

Σας δίνεται το αρχείο **Example13.java** το οποίο περιέχει τις κλάσεις **Person**, **IdPerson**, και **Example13**. Ο κώδικας όπως είναι χτυπάει λάθος όταν κάνουμε compile. Διορθώστε τα λάθη **αλλάζοντας μόνο την κλάση IdPerson**. Παραδώστε τον διορθωμένο κώδικα.

ΕΡΩΤΗΣΗ 2

Ο στόχος της άσκησης είναι να εξασκηθείτε με την κληρονομικότητα. Ορίστε την κλάση **Document** η οποία κρατάει πληροφορία για ένα κείμενο. Η κλάση έχει ένα πεδίο **text** τύπου String το οποίο κρατάει το περιεχόμενο του κειμένου. Ο constructor παίρνει σαν όρισμα ένα String με το περιεχόμενο του κειμένου. Δημιουργήστε και μία μέθοδο **print()** η οποία τυπώνει το κείμενο.

Ορίστε μια κλάση **TextFile** η οποία παράγεται (κληρονομεί) από την κλάση **Document** και κρατάει πληροφορία για ένα αρχείο κειμένου. Η κλάση έχει το πεδίο **pathname** τύπου String το οποίο κρατάει το path του αρχείου στον υπολογιστή. Ο constructor παίρνει σαν ορίσματα το pathname και το κείμενο του αρχείου. Υπερβείτε την μέθοδο **print()** ώστε να τυπώνει πρώτα το pathname και μετά (σε ξεχωριστή γραμμή) το κείμενο του αρχείου.

Ορίστε μια κλάση **Email** η οποία να παράγεται (κληρονομεί) από την κλάση **Document** και κρατάει πληροφορία για ένα email. Η κλάση έχει τα πεδία **sender** και **recipient** τύπου String τα οποία κρατάνε τον αποστολέα και παραλήπτη του email αντίστοιχα. Ο constructor παίρνει σαν ορίσματα το όνομα του αποστολέα, το όνομα του παραλήπτη, και το κείμενο του email. Υπερβείτε την μέθοδο **print()** ώστε να τυπώνει πρώτα From: και το όνομα του αποστολέα, μετά (σε ξεχωριστή γραμμή) To: και το όνομα του παραλήπτη, και τέλος (σε ξεχωριστή γραμμή) το κείμενο του αρχείου. Ορίστε επίσης την μέθοδο **attach** η οποία παίρνει σαν όρισμα ένα αντικείμενο Document και συνενώνει το κείμενο του Document σε αυτό του Email, χωρίζοντας τα με το String "----------".

Ορίστε επίσης την κλάση **DocumentTest** η οποία θα περιέχει την main καθώς και την μέθοδο **printDocuments** η οποία παίρνει σαν όρισμα ένα πίνακα από Documents και τα τυπώνει. Η **main** θα δημιουργεί ένα πίνακα από τρία Documents. Το πρώτο θα είναι ένα απλό Document, το δεύτερο ένα Email και το τρίτο ένα αρχείο. Καλέσετε την μέθοδο printDocuments για να τα τυπώσετε τα τρία Documents. Μετά κάνετε attach το αρχείο κειμένου στο email, και καλέστε ξανά την printDocuments.

Όλα τα πεδία σε όλες τις κλάσεις θα πρέπει να ορίζονται **private.**

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

- **Στοίχιση κώδικα:** Ο κώδικας πρέπει να είναι σωστά στοιχισμένος αλλιώς αφαιρείται ένα 20% του βαθμού.
- **Πεδία:** Όλα τα πεδία θα πρέπει να ορίζονται **private.**
- Κάντε turnin τα προγράμματα σας στο lab7g13@ply212.

  π.χ. turnin lab7g13@ply212 Example13.java

  Στον κώδικα να αναγράφονται σε σχόλια τα ονόματά σας και ο ΑΜ σας.

Προγραμματιστική Άσκηση, Εργαστήριο 8
Να ολοκληρωθούν μέχρι τη λήξη του εργαστηρίου

Για την άσκηση αυτή θα δημιουργήσετε κλάσεις που χειρίζονται σχήματα (παραλληλόγραμμα, κύκλους, τρίγωνα, κλπ). Για οποιοδήποτε σχήμα θέλουμε να υπολογίζουμε την περίμετρο του (**perimeter**) και το εμβαδό του (**area**). Επίσης θέλουμε να συγκρίνουμε τα σχήματα με βάση το **λόγο** του εμβαδού προς την περίμετρο.

Η υλοποίηση σας θα πρέπει να είναι αρκετά γενική ώστε να μπορεί να επεκταθεί για οποιοδήποτε σχήμα. Για το λόγο αυτό θα χρησιμοποιήσετε κληρονομικότητα. Θα ακολουθήσετε τα εξής βήματα:

- Ορίστε ένα **interface ComparableShape** το οποίο θα ορίζει την μέθοδο **compareAreaToPerimeterRatio** που θέλουμε να υλοποιούν όλα τα σχήματα. Η μέθοδος παίρνει σαν όρισμα ένα άλλο ComparableShape και αν έχουν τον ίδιο λόγο επιστρέφει 0, αν το αντικείμενο που καλεί έχει μεγαλύτερο λόγο επιστρέφει 1, αν το αντικείμενο που δίνετε σαν όρισμα έχει μεγαλύτερο λόγο επιστρέφει -1.
- Ορίστε μια **αφηρημένη** κλάση **Shape** η οποία υλοποιεί το interface ComparableShape. Η κλάση ορίζει **αφηρημένες** τις μεθόδους **perimeter** και **area** και υλοποιεί την μέθοδο **compareAreaToPerimeterRatio**. Στην κλάση ορίζουμε και ένα πεδίο String **name**, το οποίο αρχικοποιούμε στον constructor και κρατάει ένα όνομα που δίνουμε για να ξεχωρίζουμε το σχήμα. Η κλάση έχει και μία μέθοδο **print** η οποία τυπώνει το όνομα, περίμετρο, εμβαδόν.
- Ορίστε δύο ενυπόστατες (**concrete**) κλάσεις **Rectangle** και **Circle** που παράγονται από την Shape και ορίζουν το σχήμα παραλληλόγραμμο και κύκλος. Το παραλληλόγραμμο αρχικοποιείται με τις δύο πλευρές του και ο κύκλος με την ακτίνα.
- Ορίστε την κλάση **ShapeTest** η οποία έχει την main, στην οποία θα δημιουργήσετε ένα πίνακα από 4 αντικείμενα ComparableShape. Τα μισά θα τα ορίσετε κύκλους και τα άλλα μισά παραλληλόγραμμα. Διατρέξτε τον πίνακα, τυπώστε τις πληροφορίες για κάθε σχήμα, και τυπώστε το σχήμα με το μεγαλύτερο λόγο εμβαδού προς περίμετρο.

Υποδείξεις:

- Όλα τα πεδία σε όλες τις κλάσεις θα πρέπει να ορίζονται **private.** Ορίστε όλα τα μεγέθη double.
- Η περίμετρος του κύκλου με ακτίνα r είναι $2\pi r$, και το εμβαδόν πr^2.
- Για τις μαθηματικές πράξεις χρησιμοποιείστε τις (στατικές) μεθόδους της κλάσης **Math.** Π.χ., **Math.PI** μας δίνει την τιμή του π, και **Math.sqrt** είναι η μέθοδος για την τετραγωνική ρίζα. Διαβάστε περισσότερα για την Math εδώ.

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

- **Στοίχιση κώδικα:** Ο κώδικας πρέπει να είναι σωστά στοιχισμένος αλλιώς αφαιρείται ένα 20% του βαθμού.
- Κάντε turnin τα προγράμματα σας στο lab8@ply212.

  π.χ. turnin lab8@ply212 ShapeTest.java

  Στον κώδικα να αναγράφονται σε σχόλια τα ονόματά σας και ο ΑΜ σας.

Προγραμματιστική Άσκηση, Εργαστήριο 9
Να ολοκληρωθούν μέχρι τη λήξη του εργαστηρίου

Για την άσκηση αυτή θα δημιουργήσετε κλάσεις που χειρίζονται το χαρτοφυλάκιο ενός χρηματιστή. Για την άσκηση θα χρησιμοποιήσετε κληρονομικότητα και την δομή HashMap.

Ορίστε μια κλάση **Stock** η οποία κρατάει πληροφορία για κάποια μετοχή. Η κλάση έχει το σύμβολο της μετοχής, και την τιμή της μετοχής (double) τα οποία αρχικοποιούνται στον constructor. Ορίστε accessor και mutator μεθόδους όπου χρειάζεται, και μια μέθοδο **toString** η οποία επιστρέφει String με τα πεδία της κλάσης. Ορίστε επίσης μια μέθοδο **updatePrice** η οποία αλλάζει την τιμή της μετοχής κατά r% όπου το r είναι ένας τυχαίος πραγματικός αριθμός μεταξύ -0.5 και 0.5. Όλα τα πεδία πρέπει να οριστούν **private**.

Ορίστε μια κλάση **FixedRateStock** η οποία να κληρονομεί από την κλάση **Stock** και κρατάει πληροφορία για μετοχές που δίνουν σταθερή απόδοση. Η κλάση έχει επιπλέον το πεδίο **rate** το οποίο κρατάει την απόδοση της μετοχής και αρχικοποιείται στον constructor. Η κλάση υπερβαίνει την μέθοδο **updatePrice** ώστε να αυξάνεται κάθε φορά κατά ποσοστό rate. Επίσης υπερβαίνει και την μέθοδο **toString** ώστε να επιστρέφει πληροφορία και για το rate.

Ορίστε μια κλάση **Investment** η οποία κρατάει πληροφορία για μια επένδυση. Κρατάει πληροφορία για την μετοχή στην οποία έχει γίνει η επένδυση και τον αριθμό (int) των μετοχών που έχουν αγοραστεί. Έχει επίσης τις μεθόδους: **computeValue** η οποία μας επιστρέφει την αξία της επένδυσης; **buy** η οποία παίρνει σαν όρισμα ένα αριθμό και προσθέτει τις ανάλογες μετοχές; **sell** η οποία παίρνει σαν όρισμα ένα αριθμό, αφαιρεί τις αντίστοιχες μετοχές και επιστρέφει τα έσοδα από την πώληση. Αν δεν είναι εφικτή η πώληση απλά επιστρέφει μηδέν; **allSold** η οποία επιστρέφει boolean τιμή αν όλες οι μετοχές της επένδυσης έχουν πουληθεί; **toString** η οποία επιστρέφει String με όλες τις πληροφορίες της επένδυσης.

Ορίστε μια κλάση **Portfolio** η οποία κρατάει πληροφορία για το χαρτοφυλάκιο. Κρατάει ένα **HashMap** με τις επενδύσεις (Investment) που έχει το χαρτοφυλάκιο, με κλειδί το σύμβολο της μετοχής. Έχει επίσης το πεδίο **name** με το όνομα του κατόχου του χαρτοφυλακίου, και το πεδίο **cash** με το ποσό των χρημάτων που έχουν τοποθετηθεί στο χαρτοφυλάκιο, τα οποία αρχικοποιούνται στον constructor. Η κλάση έχει τις μεθόδους: **buy** που παίρνει ένα αντικείμενο Stock και ένα αριθμό μετοχών και εφόσον είναι δυνατόν αγοράζει τις μετοχές και τις προσθέτει στο χαρτοφυλάκιο; **sell** που παίρνει ένα αντικείμενο Stock και ένα αριθμό μετοχών και εφόσον είναι δυνατόν πουλάει τις μετοχές και τις αφαιρεί από το χαρτοφυλάκιο. Αν η έχουν πουληθεί όλες οι μετοχές μιας επένδυσης, τότε αφαιρείται από το χαρτοφυλάκιο; **computeValue**, η οποία υπολογίζει την συνολική αξία του χαρτοφυλακίου (συμπεριλαμβάνοντας και τα μετρητά); **toString** η οποία επιστρέφει String με τις πληροφορίες για τα περιεχόμενα του χαρτοφυλακίου (μετρητά και μετοχές).

Σας δίνεται η κλάση StockMarket η οποία χρησιμοποιεί τις παραπάνω κλάσεις για να τεστάρετε τον κώδικα σας.

Υποδείξεις:

- Όλα τα πεδία σε όλες τις κλάσεις θα πρέπει να ορίζονται **private.** Ορίστε όλα τα μεγέθη double.
- Οι μέθοδοι toString πρέπει να υλοποιηθούν με **φωλιασμένες** κλήσεις της toString.

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

- **Στοίχιση κώδικα:** Ο κώδικας πρέπει να είναι σωστά στοιχισμένος αλλιώς αφαιρείται ένα 20% του βαθμού.
- Κάντε turnin τα προγράμματα σας στο lab9@ply212.

    π.χ. turnin lab9@ply212 Portfolio.java

    Στον κώδικα να αναγράφονται σε σχόλια τα ονόματά σας και ο ΑΜ σας.

Προγραμματιστική Άσκηση, Εργαστήριο 10
Να ολοκληρωθούν μέχρι τη λήξη του εργαστηρίου

Στην άσκηση αυτή θα εξασκηθείτε στην επεξεργασία αρχείων και την χρήση των δομών της Java.

Σας δίνεται ένα αρχείο input.txt με φιλίες μεταξύ χρηστών του Facebook. Κάθε γραμμή του αρχείου έχει ένα ζευγάρι από IDs χρηστών (integer) χωρισμένα με tab, που σημαίνει ότι αυτοί οι δύο χρήστες είναι φίλοι. Διαβάστε το αρχείο και δημιουργείστε ένα καινούριο αρχείο output.txt όπου σε κάθε γραμμή θα έχετε ένα χρήστη και την λίστα με τα IDs των φίλων του. Έτσι για παράδειγμα αν στο αρχείο εισόδου έχουμε τα ζευγάρια:

1 2
1 3
4 1

Για τον χρήστη με ID 1, το αρχείο θα έχει την γραμμή:

1: 2 3 4

Για την υλοποίηση σας θα πρέπει να χρησιμοποιήσετε ένα HashMap από ArrayList. Δημιουργείστε μια κλάση FriendList η οποία θα έχει την main μέσα στην οποία θα κάνετε όλη την υλοποίηση.

Εκτός από την είσοδο σας δίνεται και το αρχείο με το σωστό αποτέλεσμα της εξόδου.

ΓΕΝΙΚΕΣ ΟΔΗΓΙΕΣ

- **Στοίχιση κώδικα:** Ο κώδικας πρέπει να είναι σωστά στοιχισμένος αλλιώς αφαιρείται ένα 20% του βαθμού.
- Κάντε turnin τα προγράμματα σας στο lab10@ply212.

  π.χ. turnin lab10@ply212 FriendList.java

  Στον κώδικα να αναγράφονται σε σχόλια τα ονόματά σας και ο ΑΜ σας.