ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
Τμήμα Πληροφορικής

ΕΠΛ 133 – Αντικειμενοστρεφής Προγραμματισμός
ΑΣΚΗΣΗ 3 – Square your mind

Διδάσκων: Καθ. Μάριος Δικαιάκος

Υπεύθυνος Άσκησης: Πύρρος Μπράτσκας

Ημερομηνία Ανάθεσης: 22 Μαρτίου 2024
Ημερομηνία Παράδοσης: 05 Απριλίου 2024 13:00

1. Στόχος

Σε αυτήν την εργασία θα ασχοληθούμε με αντικειμενοστρεφή σχεδίαση προβλημάτων, σχεδίαση διεπαφών και αφαιρετικών κλάσεων, κληρονομικότητα και πολυμορφισμό. Η άσκηση αποτελείται από δυο μέρη. Υλοποιήστε όλα τα μέρη σε αρχεία Java, όπου κάθε κλάση θα αντιπροσωπεύει ένα ξεχωριστό αντικείμενο. Και για τα δυο μέρη, αναπτύξτε την αφαιρετικότητα όσο περισσότερο γίνεται!

2. Μέρος ΄Α - Squarelotron

2.1 Περιγραφή

Το Squarelotron είναι ένα παιχνίδι το οποίο αποτελείται βασικά από ένα δισδιάστατο πίνακα ακέραιων. Αυτός ο πίνακας μπορεί να αποσυντεθεί σε τετράγωνους δακτυλίους, οι οποίοι μπορούν να περιστρέφονται ανεξάρτητα με 4 διαφορετικούς τρόπους: Πάνω-Κάτω (↕), Αριστερά-Δεξιά (↔), μέσω της κύριας αντίθετης διαγωνίου (↙) και μέσω της κύριας διαγωνίου (↘).

Για παράδειγμα, θεωρείτε τα ακόλουθα Squarelotrons:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

a

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	23	21
25	22	19	24	20

b

Το Squarelotron a) έχει 2 δακτυλίους. Ο εξωτερικός δακτύλιος περιέχει τους αριθμούς 1, 2, 3, 4, 5, 8, 9, 12, 13, 14, 15, 16, ενώ ο εσωτερικός δακτύλιος περιέχει τους αριθμούς 6, 7, 10, 11.

Το Squarelotron b) έχει δύο δακτυλίους και ένα κεντρικό κομμάτι. Ο εξωτερικός δακτύλιος περιέχει τους αριθμούς 1, 2, 3, 4, 5, 6, 10, 11, 15, 16, 21, 25, 22, 19, 24, 20, ενώ ο εσωτερικός δακτύλιος περιέχει 7, 8, 9, 12 , 14, 17, 18, 23. Ο αριθμός 13 είναι στο κέντρο.

Μια περιστροφή Πάνω-Κάτω (Upside-Down Flip) (↕) του εξωτερικού δακτυλίου του Squarelotron a) το αλλάζει ως ακολούθως:

13	14	15	16
9	6	7	12
5	10	11	8
1	2	3	4

Μια περιστροφή Αριστερά-Δεξιά (Left-Right Flip) (↔) του εσωτερικού δακτυλίου του Squarelotron b) το αλλάζει ως ακολούθως:

1	2	3	4	5
6	9	8	7	10
11	14	13	12	15
16	23	18	17	21
25	22	19	24	20

Μια περιστροφή μέσω της κύριας αντίθετης διαγωνίου (Main Inverse Diagonal) (↙) του εσωτερικού δακτυλίου του Squarelotron α) το αλλάζει ως ακολούθως:

1	2	3	4
5	11	7	8
9	10	6	12
13	14	15	16

Μια περιστροφή μέσω της κύριας διαγώνιου (Main Diagonal) (↘) του εξωτερικού δακτυλίου του Squarelotron b) το αλλάζει ως ακολούθως:

1	6	11	16	25
2	7	8	9	22
3	12	13	14	19
4	17	18	23	24
5	10	15	21	20

Δεδομένου ότι το παιχνίδι Squarelotron είναι ένα φυσικό αντικείμενο, μπορεί να περιστραφεί απλά σαν αντικείμενο στο χέρι. Για παράδειγμα, αν η πρώτη σειρά (γραμμή του πίνακα) ενός Squarelotron μεγέθους 4x4 περιέχει τους αριθμούς (1, 2, 3, 4) και το Squarelotron περιστρέφεται προς τα δεξιά κατά 90 μοίρες, τότε η δεξιά στήλη του Squarelotron θα περιέχει (1, 2, 3, 4). Αυτό δεν θεωρείται "περιστροφή".

2.2 Ζητούμενα Ά Μέρους

Στόχος αυτού του πρώτου μέρους της εργασίας είναι η δημιουργία ενός προγράμματος το οποίο πρέπει να κάνει τα παρακάτω:

- Θα ζητά από το χρήστη τη μέγεθος Squarelotron θα χρησιμοποιηθεί, 4x4 ή 5x5.
- Στη συνέχεια θα εκτυπώσει το αρχικό Squarelotron (με τους αριθμούς σε σειρά, όπως στο (α) παραπάνω. Το (b) είναι ελαφρώς ανακατεμένο).
- Μετά το πρόγραμμα θα ζητά το είδους της περιστροφής που πρέπει να εκτελέσει και θα εκτυπώσει το νέο Squarelotron μετά την περιστροφή.
- Τέλος, το πρόγραμμα πρέπει να επιτρέπει στο χρήστη να ξεκινήσει με ένα νέο Squarelotron ή να σταματήσει.

Για αυτή την εργασία θα πρέπει να υλοποιήσετε διάφορες κλάσεις οι οποίες μοντελοποιούν ένα Squarelotron, καθώς και τη λογική πίσω από τις μεταξύ τους αλληλεπιδράσεις. Για να το πετύχετε αυτό θα χρησιμοποιήσετε τις έννοιες της κληρονομικότητας και του πολυμορφισμού που έχετε διδαχθεί στις διαλέξεις του μαθήματος.

Η εργασία αυτή (όπως και οι προηγούμενες) εστιάζει στη λεπτομέρεια, γι' αυτό βεβαιωθείτε ότι έχετε διαβάσει ολόκληρη την εκφώνηση προσεκτικά και έχετε υλοποιήσει κάθε λεπτομέρεια που περιγράφεται.

Πολλά από τα χαρακτηριστικά των κλάσεων ίσως χρειάζονται μεθόδους getters και setters (αφού όλα τα χαρακτηριστικά θα είναι private). Στο υπόλοιπο της περιγραφής δε θα αναφερόμαστε στις μεθόδους αυτές, αλλά θα πρέπει να υλοποιηθούν όπου κρίνεται απαραίτητο.

Τέλος, στην περιγραφή που ακολουθεί, η έννοια κλάση αναφέρεται τόσο σε διεπαφές (interface), όσο και σε αφαιρετική (abstract) κλάση και κανονική κλάση. Εσείς στην υλοποίησή σας θα πρέπει να μετατρέψετε την έννοια κλάση στην καταλληλότερη από τις τρεις πιο πάνω μορφές.

2.3 Απαραίτητες κλάσεις

`public interface SquarelotronMethods`

Το πρόγραμμα πρέπει να έχει μια διεπαφή η οποία σας δίνεται έτοιμη. Εσείς πρέπει να αποφασίσετε για τη σχέση αυτής της διεπαφής με τις άλλες κλάσεις.

`public class Squarelotron`

Σε αυτήν την κλάση υπάρχουν μεταβλητές και μέθοδοι που είναι κοινές και στις δύο κλάσεις `SmallSquarelotron` και `LargeSquarelotron`. Αυτή η κλάση πρέπει να έχει τη μεταβλητή στιγμιότυπου … `int[][] squarelotron`. Εσείς πρέπει να αποφασίσετε για τη σχέση αυτής της κλάσης με τις άλλες κλάσεις.

`public class SmallSquarelotron`

Αυτή η κλάση υλοποιεί ένα 4x4 squarelotron. Εσείς πρέπει να αποφασίσετε για τη σχέση αυτής της κλάσης με τις άλλες κλάσεις.

`public class LargeSquarelotron`

Αυτή η κλάση υλοποιεί ένα 5x5 squarelotron. Εσείς πρέπει να αποφασίσετε για τη σχέση αυτής της κλάσης με τις άλλες κλάσεις.

2.3.1 Μέθοδοι κλάσεων

Οι παρακάτω μέθοδοι πρέπει να μπουν στις σωστές κλάσεις.

Παρατηρήστε ότι και στο 4x4 και στο 5x5 squarelotron, υπάρχουν δύο δακτύλιοι που μπορούν να περιστραφούν (το κέντρο του δακτυλίου 5x5 είναι ακίνητο). Σε καθεμία από τις ακόλουθες μεθόδους, η παράμετρος *ring* παίρνει μόνο 2 τιμές, `"outer"` ή `"inner"` με μικρά γράμματα, με τις προφανείς έννοιες.

`public SmallSquarelotron(int[] array)`
`public LargeSquarelotron(int[] array)`

Είναι οι κατασκευαστές για τους squarelotrons. Δοθέντος ενός πίνακα `int[]` σωστού μεγέθους (16 ή 25 αριθμοί), αρχικοποιεί ένα squarelotron. Μέσα στην κλάση το squarelotron αντιπροσωπεύεται από τη μεταβλητή στιγμιότυπου `private int[][] squarelotron` που δηλώνεται στην κλάση `Squarelotron`. Οι πρώτοι 4 ή 5 αριθμοί στον πίνακα που περνάμε σαν παράμετρο στους κατασκευαστές αποτελούν την πρώτη σειρά του πίνακα squarelotron και ούτω καθεξής. Η κλήση του κατασκευαστή δεν πρέπει να έχει ως αποτέλεσμα είσοδο/έξοδο δεδομένων.

`public static Squarelotron makeSquarelotron(int[] array) throws IllegalArgumentException`

Κατασκευάζει και επιστρέφει ένα squarelotron εάν πληρούνται οι ακόλουθες συνθήκες: (1) Ο πίνακας που παίρνει σαν παράμετρο είναι μήκους 16 ή 25 και (2) όλοι οι αριθμοί του πίνακα είναι μη αρνητικοί (το μηδέν επιτρέπεται) και μικρότεροι ή ίσοι με 99. Σε αντίθετη περίπτωση (παραβίαση αυτών των όρων) έχουμε `IllegalArgumentException`. Η κλήση αυτής της μεθόδου δεν πρέπει να έχει ως αποτέλεσμα είσοδο/έξοδο δεδομένων.

`public int[] numbers()`

Δοθέντος ενός 4x4 squarelotron η μέθοδος numbers επιστρέφει ένα πίνακα 16 στοιχείων (αριθμών). Δοθέντος ενός 5x5 squarelotron η μέθοδος numbers επιστρέφει ένα πίνακα 25 στοιχείων (αριθμών). Για ένα νέο-δημιουργημένο squarelotron, ο πίνακας που επιστρέφεται με αυτή τη μέθοδο θα πρέπει να έχει τους ίδιους αριθμούς στην ίδια σειρά με τον πίνακα που χρησιμοποιήθηκε για τη δημιουργία του squarelotron. Η κλήση αυτής της μεθόδου δεν πρέπει να έχει ως αποτέλεσμα είσοδο/έξοδο δεδομένων.

`public Squarelotron upsideDownFlip(String ring)`

Αυτή η μέθοδος εκτελεί την περιστροφή Πάνω-Κάτω του squarelotron, όπως περιγράφεται παραπάνω, και επιστρέφει το νέο squarelotron. Το αρχικό squarelotron δεν πρέπει να τροποποιηθεί (θα το ελέγξουμε). Η κλήση αυτής της μεθόδου δεν πρέπει να έχει ως αποτέλεσμα είσοδο/έξοδο δεδομένων.

`public Squarelotron leftRightFlip(String ring)`

Αυτή η μέθοδος εκτελεί την περιστροφή Αριστερά-Δεξιά του squarelotron, όπως περιγράφεται παραπάνω, και επιστρέφει το νέο squarelotron. Το αρχικό squarelotron δεν πρέπει να τροποποιηθεί (θα το ελέγξουμε). Η κλήση αυτής της μεθόδου δεν πρέπει να έχει ως αποτέλεσμα είσοδο/έξοδο δεδομένων.

`public Squarelotron inverseDiagonalFlip(String ring)`

Αυτή η μέθοδος εκτελεί την περιστροφή μέσω της κύριας αντίθετης διαγωνίου του squarelotron, όπως περιγράφεται παραπάνω, και επιστρέφει το νέο squarelotron. Το αρχικό squarelotron δεν πρέπει να τροποποιηθεί (θα το ελέγξουμε). Η κλήση αυτής της μεθόδου δεν πρέπει να έχει ως αποτέλεσμα είσοδο/έξοδο δεδομένων.

`public Squarelotron mainDiagonalFlip(String` *`ring`*`)`

Αυτή η μέθοδος εκτελεί την περιστροφή μέσω της κύριας διαγωνίου του squarelotron, όπως περιγράφεται παραπάνω, και επιστρέφει το νέο squarelotron. Το αρχικό squarelotron δεν πρέπει να τροποποιηθεί (θα το ελέγξουμε). Η κλήση αυτής της μεθόδου δεν πρέπει να έχει ως αποτέλεσμα είσοδο/έξοδο δεδομένων.

`public Squarelotron sideFlip(String` *`side`*`)`

Η παράμετρος `side` μπορεί να πάρει μια από τις τιμές: `"left"`, `"right"`, `"top"`, και `"bottom"`. Οι δύο υποδεικνυόμενες στήλες (αριστερότερη ή δεξιότερη) ή οι δύο υποδεικνυόμενες σειρές (δύο πάνω σειρές ή δύο κάτω σειρές) πρέπει να περιστραφούν. Το αρχικό squarelotron δεν πρέπει να τροποποιηθεί (θα το ελέγξουμε). Η κλήση αυτής της μεθόδου δεν πρέπει να έχει ως αποτέλεσμα είσοδο/έξοδο δεδομένων.

`public void rotateRight(int` *`numberOfTurns`*`)`

Η παράμετρος *`numberOfTurns`* δείχνει πόσες φορές πρέπει να περιστραφεί το squarelotron κατά `90°` δεξιόστροφα (clockwise). Οποιοσδήποτε ακέραιος αριθμός, συμπεριλαμβανομένων μηδενικών και αρνητικών ακεραίων, είναι επιτρεπτός ως όρισμα. Μια τιμή `-1` δείχνει περιστροφή κατά `90°` αριστερόστροφα (counterclockwise). Αυτή η μέθοδος τροποποιεί την εσωτερική αναπαράσταση του squarelotron και **δεν** δημιουργεί ένα νέο squarelotron. Η κλήση αυτής της μεθόδου δεν πρέπει να έχει ως αποτέλεσμα είσοδο/έξοδο δεδομένων.

```
@Override
public boolean equals(Object object)
```

Επιστρέφει `true` εάν η παράμετρος *`object`* είναι ένα squarelotron το οποίο είναι ίσος με το this squarelotron, και `false` σε διαφορετική περίπτωση. Ένα squarelotron `s1` θεωρείται ίσος με το squarelotron που προκύπτει από *rotate* του `s1`. Τα squarelotrons διαφορετικών μεγεθών δεν είναι ποτέ ίσοι. Η κλήση αυτής της μεθόδου δεν πρέπει να έχει ως αποτέλεσμα είσοδο/έξοδο δεδομένων.

```
@Override
public String toString()
```

Επιστρέφει μια εκτυπώσιμη έκδοση του `this` squarelotron. Η επιστρεφόμενη συμβολοσειρά πρέπει να περιέχει χαρακτήρες νέας γραμμής, `'\n'`, μεταξύ των γραμμών, έτσι ώστε η εκτύπωση της επιστρεφόμενης συμβολοσειράς να παράγει καθαρή, ωραία εμφάνιση. Η κλήση αυτής της μεθόδου δεν πρέπει να έχει ως αποτέλεσμα είσοδο/έξοδο δεδομένων. Δεν χρειάζεται να δημιουργήσετε unit test για αυτή τη μέθοδο.

2.4 Unit tests

Γράψτε ***unit test*** για τις μεθόδους που περιγράφονται στην ενότητα 2.3.1. Για αυτό μπορείτε να χρησιμοποιήσετε την ακόλουθη τεχνική TDD (Test-Driven Design):

1. Ξεκινήστε με μια μέθοδο που δεν κάνει τίποτα (κενή μέθοδο).
2. Γράψτε μια Junit τεστ για τη μέθοδο.
3. Εκτελέστε το τεστ και βεβαιωθείτε ότι **αποτυγχάνει**.
4. Προσθέστε κώδικα στη μέθοδο προσπαθώντας να κάνετε το τεστ να επιτύχει.
5. Εκτελέστε ξανά το τεστ.
    a. Εάν το τεστ αποτύχει, κάντε debug τη μέθοδο (ή το τεστ) μέχρι να περάσει το τεστ.
    b. Εάν επέτυχε το τεστ, continue.
6. Είναι ολοκληρωμένη η μέθοδος (κάνει ό, τι πρέπει να κάνει);

a. Εάν η μέθοδος δεν είναι πλήρης, προσθέστε κώδικα στο τεστ για να ελέγξετε τη νέα μέθοδο. Πηγαίνετε στο βήμα 3.
b. Εάν η μέθοδος είναι πλήρης, έχετε ολοκληρώσει με αυτήν τη μέθοδο.

Ακολουθήστε ακριβώς τις προδιαγραφές που περιγράφονται πιο πάνω, γιατί θα χρησιμοποιήσουμε επίσης τα δικά μας unit tests για να ελέγξουμε την εργασία σας.

Ο σχεδιασμός του περιβάλλοντος χρήστη να γίνει σε μια κλάση ονόματι SquarelotronTest η οποία να περιέχει τη μέθοδο main() του προγράμματος. Αυτή η κλάση δεν είναι μια junit test κλάση!

3. Μέρος ΄Β - Chess Classes

3.1 Ζητούμενα ΄Β Μέρους

Σε αυτό το μέρος της εργασίας θα πρέπει να γράψετε κλάσεις που μοντελοποιούν μια σκακιέρα. Γράψτε τις ακόλουθες κλάσεις και enums:

`Color` – ένα `enum` με δυο τιμές: `WHITE` και `BLACK` που αντιπροσωπεύουν τα χρώματα των κομματιών.

`Square` – μια κλάση που αντιπροσωπεύει τα τετράγωνα στη σκακιέρα. Η κλάση πρέπει να έχει τους ακόλουθους κατασκευαστές και μεθόδους:

> `public Square(char row, char col)`
> Δημιουργεί ένα αντικείμενο σε συγκεκριμένη γραμμή και στήλη. Η γραμμή παίρνει τιμές όπως `'a'` και η στήλη τιμές όπως `'1'`.
>
> `public Square(String name)`
> Χρησιμοποίει το όνομα ενός τετραγώνου όπως `"a1"` για να αρχικοποιεί ένα αντικείμενο `Square`.
>
> `public String toString()`
> Επιστρέφει το όνομα του τετραγώνου όπως `"a1"`.
>
> `public boolean equals(Object object)`

`Piece` – μια κλάση που αντιπροσωπεύει τα πιόνια. Η κλάση πρέπει να είναι abstract και να έχει τους ακόλουθους κατασκευαστές και μεθόδους:

> Ένα κατασκευαστή που παίρνει σαν παράμετρο ένα `Color`
>
> `public getColor()` που επιστρέφει το `Color` του κομματιού.
>
> `public abstract algebraicName()` η οποία επιστρέφει ένα `String` που είναι το όνομα του κομματιού: `""` για πιόνια, ή ένα από τα `"K", "Q", "B", "N","R"` για τα άλλα κομμάτια.
>
> `public abstract movesFrom(Square square)` η οποία επιστρέφει ένα `Square[]` που περιέχει όλα τα τετράγωνα στα οποία μπορεί να μετακινηθεί το κομμάτι ξεκινώντας από το τετράγωνο που παίρνει σαν παράμετρο. Θεωρούμε ότι η σκακιέρα περιλαμβάνει μόνο αυτό το κομμάτι.

Μια υποκλάση της κλάσης `Piece` ονόματι `King` η οποία αναιρεί (overrides) σωστά τις μεθόδους της κλάσης `Piece`

Μια υποκλάση της κλάσης `Piece` ονόματι `Queen` η οποία αναιρεί (overrides) σωστά τις μεθόδους της κλάσης `Piece`

Μια υποκλάση της κλάσης `Piece` ονόματι `Bishop` η οποία αναιρεί (overrides) σωστά τις μεθόδους της κλάσης `Piece`

Μια υποκλάση της κλάσης `Piece` ονόματι `Knight` η οποία αναιρεί (overrides) σωστά τις μεθόδους της κλάσης `Piece`

Μια υποκλάση της κλάσης `Piece` ονόματι `Rook` η οποία αναιρεί (overrides) σωστά τις μεθόδους της κλάσης `Piece`

Μια υποκλάση της κλάσης `Piece` ονόματι `Pawn` η οποία αναιρεί (overrides) σωστά τις μεθόδους της κλάσης `Piece`

Δοκιμάστε τις κλάσεις σας απλά χρησιμοποιώντας τες, όπως παρακάτω:

```
Piece knight = new Knight(Color.BLACK);
assert knight.algebraicName().equals("N");
assert knight.fenName().equals("n");
Square[] attackedSquares = knight.movesFrom(new Square("f6"));
// test that attackedSquares contains e8, g8, etc.
Square a1 = new Square("a1");
Square otherA1 = new Square('a', '1');
Square h8 = new Square("h8");
assert a1.equals(otherA1);
assert !a1.equals(h8);
```

3.2 Άλλες κλάσεις ΄Β Μέρους

Προσθέστε στο πρόγραμμα σας τις ακόλουθες κλάσεις:

- μια κλάση exception ονόματι `InvalidSquareException`. Το μήνυμα (String) που επιστρέφει η μέθοδος `getMessage` της κλάσης, πρέπει να είναι το τετράγωνο που δεν υπάρχει στη σκακιέρα π.χ. `a9` ή `i1` .
- Αποφασίστε εάν η κλάση `InvalidSquareException` θα είναι checked ή unchecked exception και υλοποιήστε την κατάλληλα. Στα σχόλια JavaDoc της κλάσης `InvalidSquareException` εξηγήστε την επιλογή σας (checked ή unchecked exception).
- Τροποποιήστε την κλάση `Square` ώστε η μέθοδοι της να χρησιμοποιούν την κλάση `InvalidSquareException`.
- Μια κλάση ChessTest η οποία θα περιέχει τη μέθοδο main() του προγράμματος. Κατά την εκτέλεσή της, να παρουσιάζει μια σκακιέρα και να επιτρέπει στον χρήστη τουλάχιστον μια κίνηση για κάθε πιόνι.

4. Γενικές Οδηγίες

4.1 Δομή του προγράμματος

Είναι απαραίτητο να ακολουθήσετε την παρακάτω οργάνωση του προγράμματος:

- Project name: `Homeworks`
- Package name: `hw3.squarelotron` και `hw3.chess` (κάτω από το src)
- Class names: Όπως σας ζητείται από την εκφώνηση

- Method signatures Όπως σας ζητείται από την εκφώνηση.
- Javadoc documentation για όλες τις κλάσεις και μεθόδους.
- Σωστή εφαρμογή των Java Naming Conventions.
- Να ανεβάσετε στο Moodle όλα τα αρχεία java, ένα προς ένα, **όχι σε zip!**

4.2 Αξιολόγηση του προγράμματος

Το πρόγραμμα σας θα ελεγχθεί με δικό μας πρόγραμμα δοκιμής. Για αυτό είναι σημαντικό να ακολουθήσετε πιστά τις οδηγίες της εκφώνησης. Για να βεβαιωθείτε ότι το πρόγραμμα σας τρέχει κανονικά δημιουργήστε Junit test για την κάθε κλάση.

Καλή επιτυχία