ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
Τμήμα Πληροφορικής

ΕΠΛ 133 – Αντικειμενοστρεφής Προγραμματισμός
ΑΣΚΗΣΗ 3

Διδάσκων: Καθ. Μάριος Δικαιάκος

Υπεύθυνος Άσκησης: Πύρρος Μπράτσκας

Ημερομηνία Ανάθεσης: 20 Μαρτίου 2023
Ημερομηνία Παράδοσης: 3 Απριλίου 2023 13:00

(ο κώδικας να υποβληθεί μέσω του Moodle)
http://www.cs.ucy.ac.cy/courses/EPL133

1. Στόχος

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

2. Μέρος ΄Α – Νευρώνες

2.1 Περιγραφή

Ο στόχος του ΄Α μέρους της άσκησης είναι να προσομοιώσει, με πολύ βασικό τρόπο προφανώς, έναν ανθρώπινο εγκέφαλο. Ο εγκέφαλος αποτελείται από ένα σύνολο νευρώνων με τα ακόλουθα βασικά χαρακτηριστικά:

- οι νευρώνες μπορούν να συνδεθούν μεταξύ τους
- ένας νευρώνας είναι ευαίσθητος στη λήψη ενός εξωτερικού σήματος, στο οποίο εκπέμπει όλοι με τους οποίους συνδέεται.
- ορισμένοι νευρώνες μπορεί να είναι εξειδικευμένοι.

2.2 Ζητούμενα ΄Α Μέρους

Για αυτήν την άσκηση σας παρέχεται η ακόλουθη κλάση:

- `NeuronSimulator` – μια κλάση η οποία σας παρέχει τη μέθοδο `main()` για να δοκιμάσετε τον κώδικα. Δεν πρέπει να κάνετε αλλαγές στη συγκεκριμένη κλάση.

Εσείς πρέπει να δημιουργήσετε τις ακόλουθες κλάσεις:

- `Position` - Αυτή η κλάση χρησιμοποιείται για τη μοντελοποίηση της θέσης ενός νευρώνα (σε δύο διαστάσεις για απλότητα). Θα χαρακτηρίζεται από δύο `double` που αντιπροσωπεύουν τις συντεταγμένες x και y αντίστοιχα. Οι δημόσιες μέθοδοι της κλάσης θα είναι:
    - ένας κατασκευαστής που αρχικοποιεί τα χαρακτηριστικά χρησιμοποιώντας δυο παραμέτρους `double`.
    - ένας `default` κατασκευαστής που αρχικοποιεί τις δύο συντεταγμένες με μηδέν.
    - `getX()` και `getY()`.
    - Η μέθοδος `toString()`.

- `Neuron` - Η αναπαράσταση ενός νευρώνα του εγκεφάλου. Ένας νευρώνας χαρακτηρίζεται από:
    - μια θέση (τύπου `Position`).
    - ένα εσωτερικό σήμα, ονόματι `signal`, που αντιστοιχεί στην απόκριση σε ένα ερέθισμα που λαμβάνεται από το εξωτερικό του εγκεφάλου (ένα `double`)
    - ένας συντελεστής εξασθένησης, ονόματι `attenuation`, του σήματος (ένας `double`).
    - το σύνολο των νευρώνων, ονόματι `connections`, με τα οποία συνδέεται (μια `ArrayList` of `Neuron`).

  Πρέπει να υλοποιήσετε την κλάση `Neuron`, με τρόπο που να τηρούνται οι ακόλουθοι περιορισμοί:
    - Ο κατασκευαστής της κλάσης θα πρέπει να αρχικοποιήσει τη θέση και τον συντελεστή εξασθένησης χρησιμοποιώντας τιμές που μεταβιβάζονται ως παραμέτρους και το εσωτερικό σήμα με 0. Ο κατασκευαστής θα πρέπει να είναι συμβατός με την μέθοδο main() που σας παρέχεται.

  Η κλάση `Neuron` περιλαμβάνει επίσης:

    - τους getters `Position getPosition()`, `int getNumberOfConnections()`, `Neuron getConnection(int index)`, `getAttenuation()` και `getSignal()` επιστρέφοντας αντίστοιχα:
        - τη θέση του νευρώνα,
        - το μέγεθος της λίστας `connections`,
        - το στοιχείο στη θέση `index` της λίστας `connections`,
        - το συντελεστή εξασθένησης `attenuation`,
        - το `signal`.
    - μια μέθοδο `void connection(Neuron n)` που προσθέτει τον νευρώνα `n` στο σύνολο των συνδέσεων. Ο προστιθέμενος νευρώνας πρέπει να προστεθεί στο τέλος του συνόλου.
    - μια μέθοδο `void receivesStimulus(double stimulus)` επιτρέποντας την προσομοίωση ενός νευρώνα. Αυτή η προσομοίωση έχει ως αποτέλεσμα:
        - την αποθήκευση στο εσωτερικό σήμα (`signal`) του νευρώνα της τιμής του ερεθίσματος (η παράμετρος της μεθόδου `stimulus`) πολλαπλασιασμένη με τον συντελεστή εξασθένησης `attenuation`,
        - διαδίδοντας το εσωτερικό σήμα στους συνδεδεμένους νευρώνες με κλήση της μεθόδου `receivesStimulus` σε όλα τα στοιχεία της λίστας `connections`. Αυτή η κλήση της μεθόδου `receivesStimulus` θα γίνει με παράμετρο το εσωτερικό σήμα, `signal`, του εξεταζόμενου νευρώνα (που υπολογίστηκε στο προηγούμενο bullet).
    - μια μέθοδο `toString` η οποία επιτρέπει να εμφανίσουμε έναν νευρώνα στην ακόλουθη μορφή:

```
The neuron in position <position> with attenuation <att> in connection with
   - a neuron in position <position 0>
   ...
   - a neuron in position <position n>
```

όπου <position> είναι η αναπαράσταση σε μορφή συμβολοσειράς της θέσης του νευρώνα, <att> είναι ο συντελεστής εξασθένησης και <position i> είναι η θέση του i-οστού στοιχείου του συνόλου των συνδέσεων των νευρώνων (there are two spaces at the beginning of every line related to the set connections). Στην περίπτωση που ο νευρώνας δεν είναι συνδεδεμένος με

κανέναν άλλο, η παραγόμενη αναπαράσταση θα είναι: `" without connection\n".` Αυτό το μέρος του προγράμματος μπορεί να δοκιμαστεί χρησιμοποιώντας το τμήμα της main() που σας δίνεται, μεταξύ των γραμμών: `// TEST PART 1` και `//END TEST PART1`.

- `CumulativeNeuron` - Ορισμένοι νευρώνες είναι εξειδικευμένοι και αντιμετωπίζουν το λαμβανόμενο ερέθισμα με αθροιστικό τρόπο: `internal signal = internal signal + stimulus * attenuation`

Τώρα πρέπει να υλοποιήσετε μια υποκλάση, `CumulativeNeuron`, η οποία κληρονομεί από την κλάση `Neuron` και που επαναορίζει με κατάλληλο τρόπο τη μέθοδο `receivesStimulus`.
Οι περιορισμοί που πρέπει να ακολουθηθούν είναι:

- ο κατασκευαστής της υποκλάσης θα είναι συμβατός με τη μέθοδο main που σαε παρέχεται,
- η μέθοδος `receivesStimulus` δεν θα αντιγράψει άσκοπα τα τμήματα του κώδικα της επανακαθορισμένης μεθόδου `receivesStimulus`.

Αυτό το μέρος του προγράμματος μπορεί να δοκιμαστεί χρησιμοποιώντας το τμήμα της main() που σας δίνεται, μεταξύ των γραμμών: `// TEST PART2 and // END TEST PART 2.`

- `Brain` – ο εγκέφαλος είναι ένα σύνολο (`ArrayList`)από `Neuron`. Αυτή η κλάση θα έχει τις ακόλουθες μεθόδους:
  - τους getters `int getNbNeurons()` και `Neuron getNeuron(int index)` επιστρέφοντας αντίστοιχα τον αριθμό των νευρώνων στον εγκέφαλο και τον νευρώνα στη θέση `index` στη λίστα των νευρώνων του εγκεφάλου,
  - τη μέθοδο `void addNeuron(Position pos, double attenuation)` η οποία δημιουργεί έναν νευρώνα χρησιμοποιώντας τις παρεχόμενες παραμέτρους και τον προσθέτει στο σύνολο των νευρώνων του εγκεφάλου;
  - τη μέθοδο `void addCumulativeNeuron(Position pos, double attenuation)` η οποία δημιουργεί ένα εξειδικευμένο νευρώνα (`CumulativeNeuron`) χρησιμοποιώντας τις παρεχόμενες παραμέτρους της μεθόδου και τον προσθέτει στο σύνολο των νευρώνων του εγκεφάλου;
  - τη μέθοδο `void stimulate(int index, double stimulus)` η οποία προσομοιώνει τη διέγερση του νευρώνα στη θέση `index` (stimulating the neuron with index `index`)(καλεί τη μέθοδο `receivesStimulus`);
  - τη μέθοδο `probe(int index)` επιστρέφοντας το σήμα που είναι αποθηκευμένο στον νευρώνα στη θέση `index`;
  - τη μέθοδο `void createConections()`, η οποία δημιουργεί συνδέσεις (connections) μεταξύ των νευρώνων του εγκεφάλου ακολουθώντας τον παρακάτω αλγόριθμο:
    - ο νευρώνας στη θέση 0 συνδέεται με τον νευρώνα στη θεση 1 εάν υπάρχει (*the neuron with index zero is connected to the neuron with index 1 if it exists*),
    - ο νευρώνας στη θέση 0 συνδέεται με τον νευρώνα στη θεση 2 εάν υπάρχει (*the neuron with index zero is connected to the neuron with index 2 if it exists*),
    - για όλες τις περιττές θέσεις `i` μικρότερες από το μέγεθος της λίστας των νευρώνων πλην 2: δημιουργείται μια σύνδεση μεταξύ του νευρώνα i και του νευρώνα i+1 καθώς και μεταξύ του νευρώνα i+1 και του νευρώνα i+2. (*for all odd indexes i inferior to the size of the set of neurons minus two: a connection is created between the neuron i and the neuron i+1 as well as between the neuron i+1 and the neuron i+2*).

- τη μέθοδο `toString` η οποία επιτρέπει να εμφανίσουμε έναν εγκέφαλο στην ακόλουθη μορφή (παράδειγμα εγκεφάλου με 4 νευρώνες):

```
*----------*
The brain contains 4 neuron(s)
The neuron in position (0.0, 0.0) with attenuation 0.5 in conection with
- a neuron in position (0.0, 1.0)
- a neuron in position (1.0, 0.0)
The neuron in position (0.0, 1.0) with attenuation 0.2 in connection with
- a neuron in position (1.0, 0.0)
The neuron in position (1.0, 0.0) with attenuation 1.0 in connection with
- a neuron in position (1.0, 1.0)
The neuron in position (1.0, 1.0) with attenuation 0.8 without connection
*----------*
```

There are two line breaks after every "*----------*".
Αυτό το μέρος του προγράμματος μπορεί να δοκιμαστεί χρησιμοποιώντας το τμήμα της main() μετά το // TEST PART 3.

2.3 Παράδειγμα εκτέλεσης

```
Test part 1:
--------------------
Signals:
5.0
5.0
10.0

First connection of neuron 1
The neuron in position (1.0, 0.0) with attenuation 1.0 in connection with
  - a neuron in position (1.0, 1.0)

Test part 2:
--------------------
Signal of cumulative neuron -> 10.0

Test part 3:
--------------------
Signal of 3rd neuron -> 4.800000000000001
The brain contains 4 neuron(s)
The neuron in position (0.0, 0.0) with attenuation 0.5 in connection with
  - a neuron in position (0.0, 1.0)
  - a neuron in position (1.0, 0.0)

The neuron in position (0.0, 1.0) with attenuation 0.2 in connection with
  - a neuron in position (1.0, 0.0)

The neuron in position (1.0, 0.0) with attenuation 1.0 in connection with
  - a neuron in position (1.0, 1.0)

The neuron in position (1.0, 1.0) with attenuation 0.8 without connection

*----------*
```

3 Μέρος ΄Β – Polynomials

3.1 Περιγραφή

Ένα πολυώνυμο αποτελείται από πολλούς όρους, με κάθε όρο να έχει έναν συντελεστή και μια μεταβλητή σε κάποια δύναμη. Ένα παράδειγμα πολυωνύμου είναι το ακόλουθο: $f(x)=3x^4-5x^3+2x-4$. Αυτό το πολυώνυμο έχει τέσσερις όρους.
Ο βαθμός ενός πολυωνύμου ορίζεται η μέγιστη δύναμη της μεταβλητής με μη μηδενικό συντελεστή. Στο παραπάνω παράδειγμα, ο βαθμός του πολυωνύμου είναι 4. Τα πολυώνυμα με τα οποία θα ασχοληθούμε σε αυτήν την άσκηση, έχουν μόνο θετικές δυνάμεις. Οι συντελεστές των πολυωνύμων μας είναι επίσης ακέραιοι αριθμοί. Θεωρούμε ότι δύο πολυώνυμα είναι ίδια αν περιέχουν τους ίδιους όρους.
Πολλές αλγεβρικές πράξεις είναι δυνατές με πολυώνυμα. Η απλούστερη είναι η αξιολόγηση του πολυωνύμου για μια συγκεκριμένη τιμή της μεταβλητής. Για παράδειγμα, για το πολυώνυμο: $f(x)=3x^4-5x^3+2x-4$, η αξιολόγηση του για x=2.5 είναι: $f(2.5)=3*(2.5)^4-5*(2.5)^3+2*(2.5)-4 = 40.0625$.
Τα πολυώνυμα μπορούν να προστεθούν για να δημιουργηθεί ένα νέο πολυώνυμο. Η πρόσθεση γίνεται συνδυάζοντας όλους τους όρους και προσθέτοντας τους συντελεστές των όρων με την ίδια ισχύ. Για παράδειγμα: $3x^4-5x^3+2x-4 + 2x^3+2x^2+4 = 3x^4-3x^3+2x^2+2x$. Ο βαθμός του αθροίσματος είναι το μέγιστο των βαθμών των δύο πολυωνύμων.

3.2 Ζητούμενα ΄Β Μέρους

Για αυτό το μέρος της εργασίας χρειάζεται να σχεδιάσετε και να υλοποιήσετε τουλάχιστον τα παρακάτω:

Δημιουργήστε μια διεπαφή, `Polynomial`, η οποία θα δηλώνει της λειτουργίες (πράξεις) ενός πολυώνυμου. Συγκεκριμένα, αυτή η διεπαφή θα πρέπει να έχει τις ακόλουθες μεθόδους:

- Μια μέθοδο `addTerm` που παίρνει σαν παραμέτρους έναν συντελεστή και μια δύναμη (και οι δύο ακέραιοι αριθμοί) και προσθέτει τον όρο που προκύπτει στο πολυώνυμο. (Αυτό θα σας επιτρέψει να δημιουργήσετε ένα πολυώνυμο όρο-προς-όρο.) Θα πρέπει να εγείρεται ένα `IllegalArgumentException` εάν περαστεί μια αρνητική δύναμη σαν παράμετρο.
- Μια μέθοδο `removeTerm` που παίρνει σαν παράμετρο μια δύναμη και αφαιρεί όλους τους όρους στο πολυώνυμο με αυτήν την δύναμη.
- Μια μέθοδο `getDegree` που επιστρέφει το βαθμό του πολυωνύμου.
- Μια μέθοδο `getCoefficient` που παίρνει σαν παράμετρο μια δύναμη και επιστρέφει τον συντελεστή για τον όρο με αυτήν την δύναμη.
- Μια μέθοδο `evaluate` που παίρνει σαν παράμετρο έναν δεκαδικό αριθμό διπλής ακρίβειας και επιστρέφει ένα αποτέλεσμα διπλής ακρίβειας.
- Μια μέθοδο `add` που παίρνει σαν παράμετρο ένα άλλο αντικείμενο Polynomial και επιστρέφει το πολυώνυμο που προκύπτει προσθέτοντας τα δύο πολυώνυμα. Οποιαδήποτε υλοποίηση θα πρέπει να διασφαλίζει ότι αυτή η μέθοδος δεν μεταλλάσσεται κανένα πολυώνυμο. Η υλοποίηση μπορεί να υποθέσει ότι το δεδομένο Πολυώνυμο είναι της ίδιας συγκεκριμένης κλάσης με αυτό το αντικείμενο. Εάν πρόκειται για διαφορετική κλάση, η μέθοδος να εγείρει ένα `IllegalArgumentException`.
- Οποιαδήποτε άλλη μέθοδο κρίνετε απαραίτητη.

Υλοποιήστε αυτήν τη διεπαφή μέσω μιας κλάσης, `Polynomial`. Πέρα από την υλοποίηση της διεπαφής Polynomial, αυτή η υλοποίηση θα πρέπει να έχει τα ακόλουθα χαρακτηριστικά (να υπακούει σε αυτούς τους περιορισμούς):

- Η κλάση θα πρέπει να αποθηκεύει το πολυώνυμο χρησιμοποιώντας μια συνδεδεμένη λίστα με κόμβους.
- Η κλάση πρέπει να αποθηκεύει μόνο όρους με μη μηδενικούς συντελεστές.
- Η κλάση θα πρέπει να αποθηκεύει τους πολυωνυμικούς όρους με φθίνουσα σειρά των δυνάμεών τους.
- Η κλάση θα πρέπει να έχει έναν κατασκευαστή χωρίς παραμέτρους που δημιουργεί ένα πολυώνυμο χωρίς όρους, δηλαδή το πολυώνυμο 0.
- Η κλάση θα πρέπει να έχει έναν άλλο κατασκευαστή που παίρνει σαν παράμετρο ένα πολυώνυμο ως συμβολοσειρά, το αναλύει και δημιουργεί το πολυώνυμο ανάλογα. Η συμβολοσειρά περιέχει το πολυώνυμο, με κάθε όρο να χωρίζεται από ένα κενό. Τα ακόλουθα παραδείγματα θα πρέπει να λειτουργούν με τον κατασκευαστή σας:
    - "4x^3 +3x^1 -5"
    - "-3x^4 -2x^5 -5 +11x^1"
- Προσθέστε όποια άλλη μέθοδο κρίνετε απαραίτητη.

Εκτός των παραπάνω κλάσεων πρέπει να υλοποιήσετε και άλλες κλάσεις οι οποίες θα κάνουν την αντικειμενοστρεφή σχεδίαση αυτού του μέρους της εργασίας καλύτερη! Επίσης, πρέπει να δημιουργήσετε και μια κλάση ονόματι `TestPolynomial` η οποία να περιέχει μόνο τη μέθοδο main.

Γράψτε δοκιμές `Junit tests` που δοκιμάζουν διεξοδικά αυτήν την υλοποίηση. Αυτές οι δοκιμές πρέπει να βρίσκονται στις συγκεκριμένες κλάσεις δοκιμής όπως έχετε διδαχτεί στο μάθημα.

4 Γενικές Οδηγίες

4.1 Δομή του προγράμματος

Είναι απαραίτητο να ακολουθήσετε την παρακάτω οργάνωση του προγράμματος:

- Package name: `username.hw3.neurons` και `username.hw3.polynomial` (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.

Το πρόγραμμα σας θα πρέπει να περιλαμβάνει εύστοχα και περιεκτικά σχόλια, να έχει καλή στοίχιση και το όνομα κάθε μεταβλητής, σταθεράς, ή συνάρτησης να είναι ενδεικτικό του ρόλου της. **Να χρησιμοποιήσετε το λογισμικό τεκμηρίωσης javadoc** έτσι ώστε να μπορούμε να μετατρέψουμε τα σχόλια του προγράμματός σας σε HTML αρχεία και να τα δούμε με ένα browser. Η συστηματική αντιμετώπιση της λύσης ενός προβλήματος περιλαμβάνει στο παρόν στάδιο τη διάσπαση του προβλήματος σε μικρότερα ανεξάρτητα προβλήματα που κατά κανόνα κωδικοποιούμε σε ξεχωριστές συναρτήσεις. Για αυτό τον λόγο σας καλούμε να κάνετε χρήση μεθόδων και άλλων τεχνικών αντικειμενοστραφής προγραμματισμού. Επίσης, σας θυμίζουμε ότι κατά την διάρκεια της εκτέλεσης του προγράμματος σας, αυτό θα πρέπει να δίνει τα κατάλληλα μηνύματα σε περίπτωση λάθους.

Επίσης να ακολουθήσετε τα πιο κάτω βήματα όταν υποβάλετε την άσκηση σας στο Moodle:

1. **Ανεβάστε στο Moodle τα αρχεία ένα προς ένα.**

Καλή επιτυχία.