ΗΥ-252 Αντικειμενοστραφής Προγραμματισμός

Χειμερινό Εξάμηνο 2011
Διδάσκων: Χριστοφίδης Βασίλης

2η Σειρά Ασκήσεων

Ημερομηνία Παράδοσης: 25/11/2011

Παρακάτω σας δίνονται οι ορισμοί τεσσάρων διαφορετικών Αφαιρετικών Τύπων Δεδομένων (ΑΤΔ). Για όλες τις λειτουργίες (operations) αυτών των ΑΤΔ, δώστε τις υπογραφές (signatures), το είδος τους ανάλογα με την επίδραση που έχουν στην κατάσταση των αντικειμένων (**constructors, accessors, transformers**), και τις εκ των προτέρων, τις εκ των υστέρων και τις αμετάβλητες συνθήκες (**preconditions**, **postconditions, invariants)**. Στη συνέχεια, υλοποιήστε τους ΑΤΔ χρησιμοποιώντας κλάσεις **Java**. Χρησιμοποιήστε τους μηχανισμούς των **JUnit Tests** της Java για να ελέγξετε την ορθότητα του κώδικά σας. Τεκμηριώστε τον κώδικα σας με την βοήθεια των **Javadocs**. Για κάθε άσκηση, πέρα από τον πηγαίο κώδικα που υλοποιεί τους ΑΤΔ που σας ζητούνται, θα πρέπει να συμπεριλάβετε και όλα τα .html αρχεία που παράγει η Javadoc, τα JUnit tests καθώς και μικρά προγράμματα πελάτες που χρησιμοποιούν τις λειτουργίες των ΑΤΔ που υλοποιήσατε.

Παραδοτέα:

`Complex/Complex.java`
`Pic&RGB/Pic.java`
`Pic&RGB/RGB.java`
`Graph/Graph.java`
`MetricSpaces/PointInAMetricSpace.java`
`MetricSpaces/RealNumberMetric.java`
`MetricSpaces/ComplexNumberMetric.java`
`MetricSpaces/PlaneMetric.java`

Επίσης, πρέπει να παραδώσετε τα παρακάτω αρχεία με JUnit Tests:
`complex/ComplexTest.java`
`pic&rgb/Pic&RgbTest.java`
`graph/GraphTest.java`
`metric/MetricSpacesTest.java`

ΠΡΟΣΟΧΗ!!! Σε καμία περίπτωση δε πρέπει να αλλάξετε τα ονόματα των μεθόδων των συμβολαίων, τα ονόματα των κλάσεων ή των πακέτων που σας δίνονται.

Άσκηση 1 – Μιγαδικοί Αριθμοί (25%)
Όνομα πακέτου: `Complex`
Όνομα αρχείου: `Complex.java`

Σχεδιάστε και υλοποιήστε τον αφαιρετικό τύπο δεδομένων (ΑΤΔ) ενός μιγαδικού αριθμού (με όνομα **Complex**). Ένας μιγαδικός αριθμός αποτελείται από ένα πραγματικό (**real**) και ένα φανταστικό (**imaginary**) μέρος. Θεωρούμε ένα συμβόλαιο για τον ΑΤΔ που περιλαμβάνει τις εξής λειτουργίες:

- Δημιουργία ενός νέου μιγαδικού αριθμού

```
public Complex();
```

- Αλλαγή του πραγματικού μέρους του μιγαδικού αριθμού

```
public void setReal(Float re);
```

- Αλλαγή του φανταστικού μέρους

```
public void setImaginary(Float im);
```

- Επιστροφή του πραγματικού μέρους

```
public Float getReal();
```

- Επιστροφή του φανταστικού μέρους

```
public Float getImaginary();
```

- Έλεγχος εάν ένας άλλος μιγαδικός αριθμός είναι ίσος με τον αρχικό

```
public boolean isEqual(Complex other);
```

- Πρόσθεση ενός άλλου μιγαδικού αριθμού στον αρχικό

```
public Complex add(Complex other);
```

- Επιστροφή μιας κειμενικής αναπαράστασης του μιγαδικού αριθμού

```
public String toString();
```

Σημασιολογία:

- Αρχικά, ένας μιγαδικός αριθμός έχει πραγματικό και φανταστικό μέρος ίσο με το 0.0, εκτός και αν δοθούν διαφορετικές τιμές ως παράμετροι κατά την δημιουργία του ΑΤΔ.
- Το πραγματικό και το φανταστικό μέρος ενός μιγαδικού είναι και οι δύο πραγματικοί αριθμοί. Το φανταστικό μέρος ακολουθεί ο φανταστικός αριθμός **i**, που έχει την ιδιότητα $\mathbf{i}^2 = -1$.

- Για να είναι ένας μιγαδικός αριθμός ίσος με έναν άλλον, θα πρέπει τα πραγματικά μέρη τους να είναι ίσα και τα φανταστικά τους μέρη ίσα αντίστοιχα.
- Για να προσθέσουμε δύο μιγαδικούς αριθμούς, προσθέτουμε τα πραγματικά μέρη τους μαζί και μετά τα φανταστικά μέρη τους.
- Η text αναπαράσταση πρέπει να είναι της μορφής: **Re() + Im() * i**, όπου **Re()** είναι το πραγματικό μέρος και **Im()** το φανταστικό.

Άσκηση 2 – Εικόνες και Κουκίδες (25%)

Όνομα πακέτου: `Pic&RGB`
Ονόματα αρχείων: `Pic.java`
`RGB.java`

Σχεδιάστε και υλοποιήστε τον αφαιρετικό τύπο δεδομένων (ΑΤΔ) μιας εικόνας με κουκίδες (pixels) (με όνομα **Pic)** και έναν ΑΤΔ χρωμάτων RGB (με όνομα **RGBColor**). Θεωρούμε ένα συμβόλαιο για τον ΑΤΔ **Pic** που περιλαμβάνει τις εξής λειτουργίες:

- Δημιουργία μιας κενής εικόνας.

```
public Pic();
```

- Δημιουργία μιας εικόνας με μια μόνο κουκίδα χρώματος **c**.

```
public Pic(RGBColor c);
```

- Τοποθέτηση μιας εικόνας κάτω από μία άλλη και επιστροφή της νέας εικόνας που δημιουργήθηκε. Οι δύο εικόνες που ενώνονται θα πρέπει να έχουν το ίδιο μήκος, όχι όμως αναγκαστικά και το ίδιο ύψος.

```
public Pic addBellow(Pic other);
```

- Περιστροφή της εικόνας κατά 90 μοίρες.

```
public Pic rotate();
```

- Αναστροφή της εικόνας.

```
public Pic flip();
```

- Επιστροφή του ύψους της εικόνας.

```
public int height();
```

- Επιστροφή του μήκους της εικόνας.

```
public int length();
```

- Επιστροφή του χρώματος μιας κουκίδας που βρίσκεται στην θέση x,y.

```
public RGBColor pixelAt(int x, int y);
```

Σημασιολογία:

- Για την δημιουργία μιας εικόνας αρχικά δημιουργούμε την πρώτη κουκίδα χρώματος **c**. Έπειτα δημιουργούμε την επόμενη κουκίδα και ενώνουμε τις δύο κουκίδα την μια κάτω από το άλλη και παίρνουμε μια νέα εικόνα που τώρα αποτελείται από δύο κουκίδες. Αυτή η διαδικασία συνεχίζεται αναδρομικά με αποτέλεσμα να "χτίζουμε" την εικόνα από άλλες εικόνες. Παρατηρήστε ότι εάν έχω κατασκευάσει μια εικόνα με ένα τετράγωνο 2x2 χρώματος μαύρου και μια εικόνα με ένα παραλληλόγραμμο 4x2 χρώματος λευκού, μπορώ να ενώσω τις δύο εικόνες και να πάρω μια νέα εικόνα που αναπαριστά ένα παραλληλόγραμμο 6x2 με μαύρο χρώμα στην κορυφή και λευκό στην βάση.
- Μπορούμε ακόμα να περιστρέψουμε μια εικόνα κατά 90 μοίρες δεξιά. Για παράδειγμα, μετά την περιστροφή του παραπάνω παραλληλογράμμου 6x2, θα προκύψει ένα νέο παραλληλόγραμμο διαστάσεων 2x6.
- Αντίστοιχα, κατά την αναστροφή μιας εικόνας, τα pixels που βρίσκονται δεξιά της εικόνας θα μεταφερθούν αριστερά και τα αριστερά, δεξιά. Το αποτέλεσμα αυτής της λειτουργίας είναι η δημιουργία μιας εικόνας αντίστοιχης με αυτή ενός ειδώλου μέσα από ένα καθρέφτη.

Θεωρούμε ένα συμβόλαιο για τον ΑΤΔ **RGBColor** που περιλαμβάνει τις εξής λειτουργίες:

- Δημιουργία ενός νέου χρώματος με προκαθορισμένες τιμές κόκκινο = 0, πράσινο = 0, μπλε = 0.

```
public RGBColor();
```

- Δημιουργία ενός νέου χρώματος με τιμές πράσινο, κόκκινο, μπλε που ορίζονται από τον χρήστη.

```
public RGBColor(int red, int green, int blue);
```

- Αλλαγή οποιουδήποτε από τα τρία χρώματα.

```
public void setRed(int red);

public void setBlue(int blue);

public void setGreen(int green);
```

- Επιστροφή οποιουδήποτε από τα τρία χρώματα.

```
public int getRed();

public int getGreen();

public int getBlue();
```

- Έλεγχος εάν δύο χρώματα είναι ίδια.

```
public boolean isEqual(RGBColor that);
```

Σημασιολογία:

- Ένα χρώμα **RGB** αποτελείται από τρεις ακέραιες τιμές από 0 έως 255 όπου η πρώτη τιμή συμβολίζει το κόκκινο, η δεύτερη το πράσινο και η τρίτη το μπλε
- Για να είναι δύο χρώματα RGB ίδια, θα πρέπει να συμφωνούν και στα τρία διαφορετικά πεδία.

Άσκηση 3 – Γράφος (25%)

Όνομα πακέτου: `Graph`
Ονόματα αρχείων: `Graph.java`

Σχεδιάστε και υλοποιήστε τον Αφαιρετικό Τύπο Δεδομένων (ΑΤΔ) ενός μη κατευθυνόμενου γράφου (με όνομα **Graph**). Ένας γράφος αποτελείται από ένα σύνολο κόμβων **V** και ένα σύνολο ακμών **E**. Κάθε ακμή συνδέει δύο κόμβους από το σύνολο **V**. Επίσης κάθε κόμβος έχει μία ετικέτα (label). Θεωρούμε ένα συμβόλαιο για τον ΑΤΔ **Graph** που περιλαμβάνει τις παρακάτω λειτουργίες:

- Δημιουργία ενός κενού γράφου.

```
public Graph();
```

- Εισαγωγή ενός νέου κόμβου με μια συγκεκριμένη ετικέτα.

```
public void insertVertex(String label);
```

- Εισαγωγή μίας νέας ακμής που συνδέει δύο κόμβους που ήδη υπάρχουν μέσα στο γράφο.

```
public boolean insertEdge(String vertex1, String vertex2);
```

- Διαγραφή ενός κόμβου και όλων των ακμών που πρόσκεινται στον υπό διαγραφή κόμβο.

```
public boolean removeVertex(String label);
```

- Διαγραφή μίας ακμής.

```
public boolean removeEdge(String vertex1, String vertex2);
```

- Έλεγχος εάν δύο κόμβοι ενώνονται με μία ακμή.

```
public boolean checkEdge(String vertex1, String vertex2);
```

- Επιστροφή όλων των κόμβων του γράφου.

```
public Set getNodes();
```

- Επιστροφή όλων των γειτονικών κόμβων ενός κόμβου.

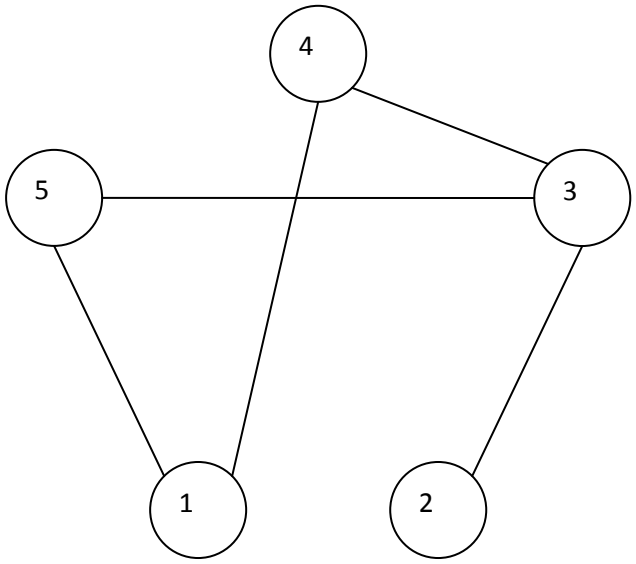
```
public Vector getNeighbours(String vertex);
```

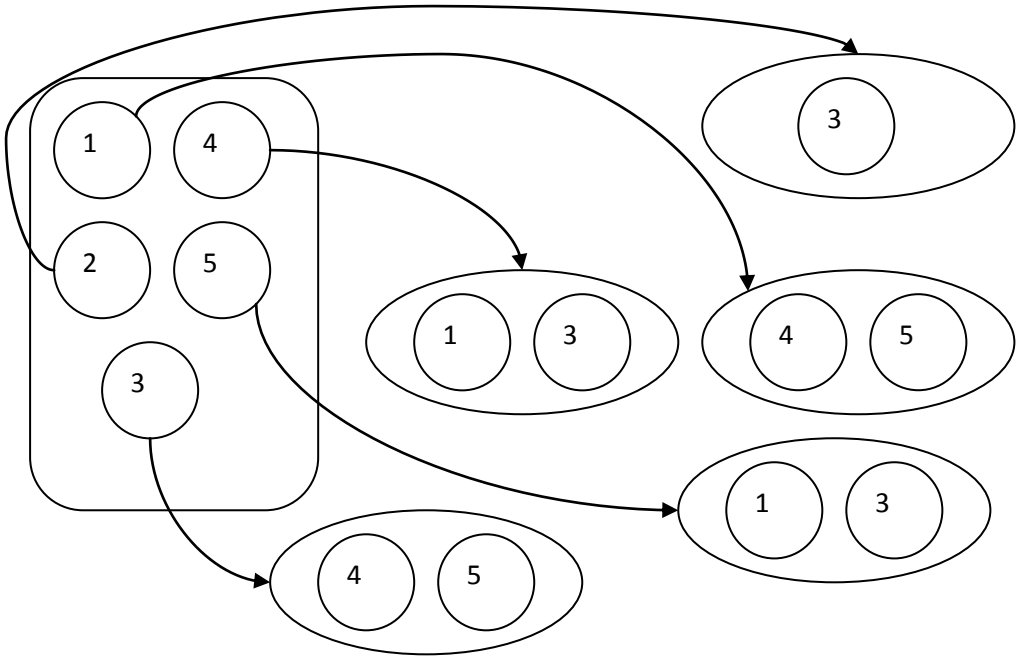
Σημασιολογία:

- Κάθε κόμβος έχει μια μοναδική ετικέτα και κατά την εισαγωγή ενός νέου κόμβου θα πρέπει να γίνεται έλεγχος της μοναδικότητας των ετικετών.
- Κάθε ακμή συνδέει ακριβώς δύο κόμβους χωρίς να έχει σημασία ο προσανατολισμός της ακμής.
- Κατά την εισαγωγή μίας νέας ακμής θα πρέπει να δίνεται σαν παράμετρος η ετικέτα των δύο κόμβων που ενώνει αυτή η ακμή και να γίνεται έλεγχος εάν όντως οι κόμβοι αυτοί υπάρχουν.
- Δύο κόμβοι μπορούν να συνδέονται μόνο με μία ακμή.
- Στην σάρωση όλων των κόμβων του γράφου δεν έχει σημασία η σειρά που θα επιστρέφονται αρκεί να επιστρέφονται ακριβώς μία φορά
- Γειτονικοί κόμβοι ενός δοθέντος κόμβου είναι όλοι οι κόμβοι που συνδέονται με τον δοθέντα κόμβο με μία μόνο ακμή.

Βοήθεια: Η υλοποίησή σας θα πρέπει να χρησιμοποιεί μόνο βασικές δομές από το στάνταρ API της Java. Για παράδειγμα, μπορούμε να κατασκευάσουμε ένα σύνολο (set) που αναπαριστά όλους τους κόμβους του γράφου και για κάθε κόμβο να κρατούμε ένα δεύτερο σύνολο με όλους τους κόμβους που συνδέονται μαζί του με μία ακμή (δηλ. τους γειτονικούς του κόμβους). Αυτή η υλοποίηση είναι γνωστή ως σύνολο γειτνίασης (**adjacency-set).** Στο πρόγραμμα πελάτης μπορείτε να τυπώσετε όλα τα στοιχεία που σας επιστρέφουν οι τελευταίες δύο μέθοδοι του συμβολαίου του ΑΤΔ έτσι ώστε να είστε σίγουροι ότι η δομή σας λειτουργεί σωστά.

Παρακάτω σας δίνεται μια γραφική απεικόνιση αυτής της υλοποίησης. Το πρώτο σχήμα δείχνει ένα παράδειγμα γράφου και το δεύτερο την υλοποίησή του χρησιμοποιώντας ένα σύνολο γειτνίασης. Για την υλοποίηση του ΑΤΔ **Graph** μπορείτε φυσικά να ακολουθήσετε και άλλες τεχνικές υλοποίησης.





Άσκηση 4 – Μετρικοί Χώροι (Metric Spaces) (25%)

Όνομα πακέτου: `MetricSpaces`
Ονόματα αρχείων: `PointInAMetricSpace.java, RealNumberMetric.java, ComplexNumberMetric.java, PlaneMetric.java`

Σχεδιάστε και υλοποιήστε τον Αφαιρετικό Τύπο Δεδομένων (ΑΤΔ) σημείων σε έναν μετρικό χώρο (με όνομα **PointInAMetricSpace**) . Πιο συγκεκριμένα, έστω X ένα σύνολο σημείων . Με τον όρο **μετρική** πάνω στο σύνολο X ουσιαστικά εννοούμε την απόσταση $d(x, y) \in R$ για κάθε ζευγάρι "σημείων" X, Y που ανήκουν στο σύνολο X και ικανοποιούν τις παρακάτω συνθήκες :

1) (Ταυτότητα) Για όλα τα $x, y \in X$, $d(x, y) = 0$ αν και μόνο αν $x = y$.

2) (Συμμετρικότητα) Για όλα τα $x, y \in X$, $d(x, y) = d(y, x)$.

3) (Τριγωνική ανισότητα) Για όλα τα $x, y, z \in X$, $d(x, y) + d(y, z) \geq d(x, z)$.

Με τον όρο **μετρικός χώρος** εννοούμε το σύνολο X μαζί με την κάθε μετρική. **Απόσταση** δύο σημείων x, y ενός μετρικού χώρου, ονομάζεται η τιμή d(x,y). Σε έναν μετρικό χώρο επιπλέον, μπορεί να δείξει κανείς ότι για όλα τα $x, y \in X$, $d(x, y) >= 0$ (δηλ αποτελεί αμετάβλητη συνθήκη). Θεωρούμε ένα συμβόλαιο για τον ΑΤΔ **PointIsnAMetricSpace** που περιλαμβάνει τις παρακάτω λειτουργίες:

- Δημιουργία ενός σημείου σε έναν Μετρικό Χώρο

  ```
  public PointsInAMetricSpace();
  ```

- Υπολογισμός της απόστασης δύο σημείων

  ```
  public abstract double distance();
  ```

- Έλεγχος αν οι αποστάσεις 2 μετρικών είναι ίσες

  ```
  public boolean isEqual(PointInAMetricSpace b);
  ```

- Σύγκριση δύο μετρικών (να συγκρίνονται 2 αποστάσεις μεταξύ τους)

  ```
  public int compareTo(PointInAMetricSpace b);
  ```

- Επιστροφή μιας κειμενικής αναπαράστασης της απόστασης

  ```
  public String toString();
  ```

Χρησιμοποιώντας τώρα το παραπάνω συμβόλαιο σε συνδυασμό με τις λειτουργίες της κληρονομικότητας που μας προσφέρει η Java θα πρέπει να υλοποιήσετε τις κλάσεις RealNumberMetric., ComplexNumberMetric και PlaneMetric για να υπολογίζετε τις αποστάσεις στις περιπτώσεις που έχουμε σημεία στο χώρο των πραγματικών αριθμών , στο χώρο των μιγαδικών αριθμών και στον επίπεδο χώρο. Οι αποστάσεις αυτές υπολογίζονται με βάση του παρακάτω τύπους :

1) **Χώρος πραγματικών αριθμών**
   - $d(x, y) = |x - y|$.

2) **Χώρος μιγαδικών αριθμών**
   - $d((x_1, y_1), (x_2, y_2)) = \sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$.

3) **Επίπεδος Χώρος**
   - $d_\infty((x_1, y_1), (x_2, y_2)) = \max\{x_1 - x_2|, |y_1 - y_2|\}$.

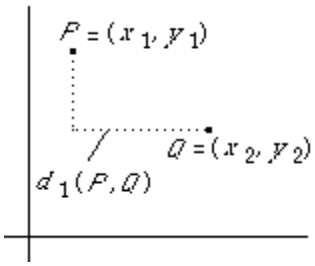
Βοήθεια: Ουσιαστικά οι τρείς παραπάνω κλάσεις θα είναι υποκλάσεις της κύριας κλάσης που θα υπάρχει στο πρόγραμμα σας και η καθεμία ανάλογα σε ποιο χώρο αναφερόμαστε θα χρησιμοποιείται για τον υπολογισμό της αντίστοιχης απόστασης των δοθέντων σημείων και κατ'επέκταση και της αντίστοιχης μετρικής . Επίσης η αναπαράσταση ενός σημείου στο χώρο γίνεται με βάση την οριζόντια συντεταγμένη Χ καθώς και με την κάθετη συντεταγμένη Υ στη περίπτωση που έχουμε κάποιο σημείο στο χώρο των 2

διαστάσεων . Για τους μιγαδικούς αριθμούς έχουμε έναν αριθμό για το πραγματικό μέρος και ένα για το φανταστικό μέρος. Σε περίπτωση που μιλάμε για χώρο περισσότερων διαστάσεων έχουμε και τις αντίστοιχες X,Y,Z κτλ συντεταγμένες .

Hint: Για την εκτέλεση των πράξεων, χρησιμοποιείστε τις συναρτήσεις της έτοιμης κλάση Math της java:
http://download.oracle.com/javase/1.4.2/docs/api/java/lang/Math.html.
Για παράδειγμα, αν επιθυμείτε να βρείτε το μέγιστο μεταξύ δύο αριθμών καλείτε τη συνάρτηση max της κλάσης Math, ως εξής: Math.max(3,2);

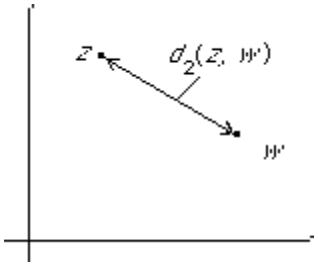
Παραδείγματα υπολογισμού αποστάσεων

Α) Παράδειγμα υπολογισμού της απόστασης στην περίπτωση που βρισκόμαστε στο R^2



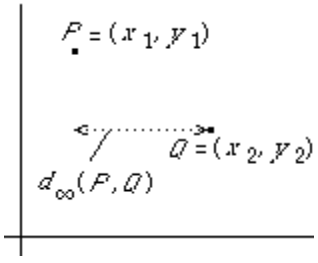
$$d_1((x_1, y_1), (x_2, y_2)) = |x_1 - x_2| + |y_1 - y_2|$$

Β) Παράδειγμα υπολογισμού της απόστασης στην περίπτωση που βρισκόμαστε στο χώρο των μιγαδικών



$$d_2(z, w) = |z - w|$$

Γ) Παράδειγμα υπολογισμού της απόστασης στην περίπτωση που βρισκόμαστε στο R^2 και η απόσταση τείνει στο άπειρο



$$d_\infty((x_1, y_1), (x_2, y_2)) = \max\{x_1 - x_2|, |y_1 - y_2|\}$$