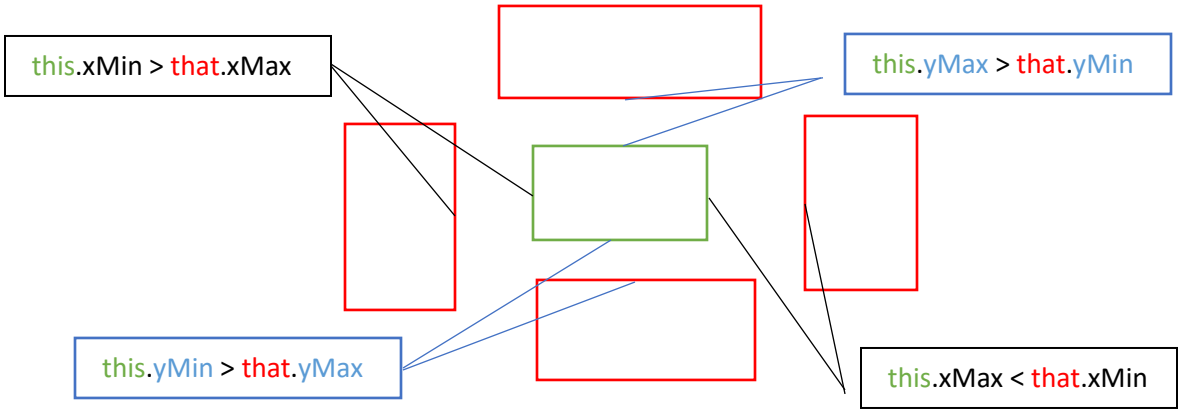
DATA STRUCTURES

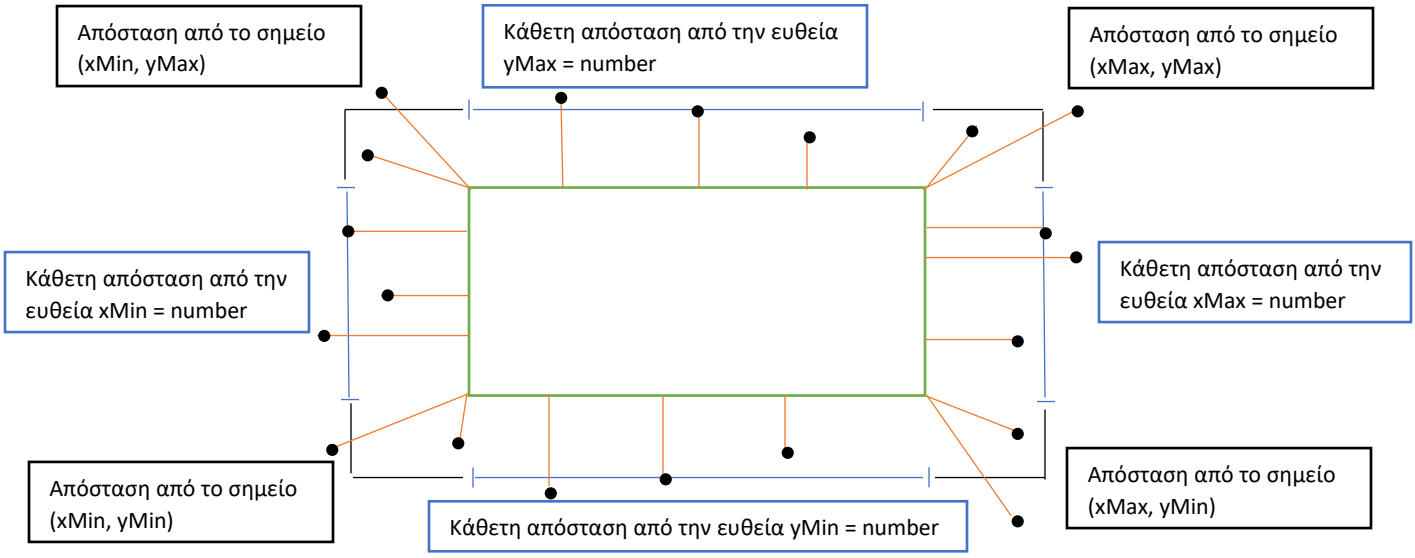
ΜΕΡΟΣ Δ ΑΣΚΗΣΗ 3

α)Rectangle.java:

- **contains(Point p):**Είναι προφανές ότι όταν η συντεταγμένη x του σημείου είναι ανάμεσα από το xMin και το xMax ή ίσο με κάποιο απο τα δύο και επιπλέον η συντεταγμένη y του σημείου είναι ανάμεσα από το yMin και το yMax ή ίσο με κάποιο απο τα δύο του ορθογωνίου παραλληλογράμμου τότε το σημείο αυτό βρίσκεται μέσα η πάνω στα άκρα του παραλληλογράμμου.
- **intersects(Rectangle that):**Για να βρούμε πότε ένα παραλληλόγραμμο έχει κοίνα σημεία με ένα άλλο και να επιστρέψουμε true αρκεί να γυρήσουμε για τις 4 περιπτώσεις που δεν έχουν κοινά σημεία false και για όλες τις υπόλοιπες true. 4 περιπτώσεις να μην έχουν κοινά σημεία



- **quareDistanceTo(Point z):**Η υλοποιήση της distanceTo έγινε στην square και στην distanceTo παίρνουμε απλά την ρίζα του αποτελέσματος της square.Το παρακάτω σχήμα δείχνει της 8 περιπτώσης.Αν βρίσκεται το σημείο μέσα στο παραλληλόγραμμο επιστρέφουμε 0



β) TwoDTree.java:

- **findLeftRectangle(parentNodeRectangle,parentNode),findRightRectangle(parentNodeRectangle,parentNode):**Αρχικά έχουμε αυτές τις δύο μεθόδους τις οποίες θα χρησιμοποιήσουμε για να υλοποιήσουμε τις μεθόδους **rangeSearch** και **nearestNeighbot.**Οι δύο αύτες μέθοδοι παίρνουν ως όρισμα τον πατέρα των δύο παιδιών(αριστερό-δεξί) και βρίσκουν τα παραλληλόγραμμα που αντιστοιχούν στα 2 παιδιά.Προφανώς παίζει ρόλο στον υπολογισμό αν ο πατέρας βρίσκεται σε μονό ή σε ζυγό επίπεδο.Αν το επίπεδο είναι ζυγό η x συντεταγμένη του πατέρα κρίνει τα δύο παραλληλόγραμμα,αν είναι μονό τότε κοιτάμε την y συντεταγμένη του πατέρα
- **rangeSearch(Rectangle userInputRectangle):**Είναι η αρχική δημόσια μέθοδος μέσα στην οποία φτιάχνουμε την ουρά που θα χρησιμοποιήσουμε και καλούμε την βασική **rangeSearch** μέσα στην οποία στέλνουμε την ρίζα του δέντρου από την οποία θα ξεκινήσει η αναδρομή, την ουρά που φτιάξαμε και το παραλληλόγραμμο που έδωσε ο χρήστης και το παραλληλόγραμμο της ρίζας που είναι το [0,100]x[0,100].Στην αρχή τσεκάρουμε την βασική περίπτωση αν είναι ένας κόμβος null να επιστρέφει η μέθοδος δηλαδή αν έχουμε φτάσει σε κάποιο φύλλο.Μετά πολυ απλά θα παράγουμε αναδρομικά τα παιδιά ενός κόμβου αν και μόνο αν ο κόμβος περάσει από το πρώτο if το οποίο ελένχει αν το παραλληλόγραμμο του κόμβου έχει κοινα σημεία με το παραλληλόγραμμο του χρήστη.Αν έχει τότε αρχικά με το δεύτερο if κοιτάμε αν το σημείο που έχει ο κόμβος αυτος περιέχεται στου χρήστη το παραλληλόγραμμο,αν ναι τότε το βάζουμε στην ουρά,αν όχι καλούμε την μέθοδο για τα παιδιά του και στέλνουμε τα παιδιά, το παραλληλόγραμμο του χρήστη ,την ουρά και επίσης καλούμε τις μεθόδους οι οποίες γυρνάνε τα παραλληλόγραμμα των παδιών.Αν δεν μπει στο πρώτο if ο κόμβος ,δεν τον επεκτείνουμε δηλαδή δεν ψάχνουμε τα παιδιά του.Και έτσι ψάχνουμε όσους κόμβους μας επιτρέπει το πρώτο if και βρίσκουμε ολα τα σημεία που περιέχονται στο παραλληλόγραμμο του χρήστη χωρίς να χρείαστει να ψάξουμε όλο το δέντρο στην καλύτερη περίπτωση.
- **nearestNeighbor(Point userInputPoint):**Είναι η αρχική δημόσια μέθοδος με την οποία γυρνάμε το αποτέλεσμα της βασικής μεθόδου μέσα στην οποία στέλνουμε την ρίζα με την οποία θα ξεκινήσει η αναδρομή, το κοντινότερο σημείο το οποίο είναι null στην αρχή, το ορθογώνιο της ρίζας που είναι το [0,100]x[0,100],και το σημείο που μας δίνει ο χρήστης και για το οποίο ψάχνουμε το κοντινότερο σημείο.Βασική περίπτωση παλι είναι όταν ο κόμβος είναι null γυρνάμε απλά το κοντινότερο σημείο.Αρχικοποιούμε σαν 0 τις 2 αποστάσεις που σε κάθε αναδρομή θα κρίνουν αν θα επεκτείνουμε έναν κόμβο και θα καλέσουμε την μέθοδο για τα παιδιά του ή όχι.Η **distanceInputPointFromNearest** υπολογίζει την απόσταση του σημείου που έδωσε ο χρηστης από το τρέχον κοντινότερο σημείο η οποία αν έιναι μικρότερη απο την **distanceInputPointFromNodeRectangle** δηλαδή από την απόσταση που έχει το σημείο του χρηστη απο το παραλληλόγραμμο του κόμβου ή με πιο κατανοητά λογιά την ελάχιστη απόσταση απο το παραλληλόγραμμο στο οποίο βρίσκεται αυτός ο κόμβος άρα ελενχει και τον κόμβο και όλα τα σημεια μέσα σε αυτό το παραλληλόγραμμο δηλαδη το υποδέντρο του κόμβου οπότε και εφόσον αυτή η απόσταση είναι η μικρότερη, αν αυτή η μικρότερη απόσταση είναι μεγαλύτερη από την απόσταση του nearest τότε δεν έχει νόημα να ψάξουμε οτιδήποτε υπάρχει μέσα στο παραλληλόγραμμο αυτο γιατί όλα τα σημεία του δεν θα είναι σίγουρα πιο κοντά από το nearest.Επίσης αυτές τις αποστάσεις τις κοιταμε μόνο όταν το nearest δεν είναι null δηλαδή όταν βρισκόμαστε στην ρίζα δεν τις υπολογίζουμε.Άρα το βασικό if της μεθόδου αυτής που αφήνει έναν κόμβο να επεκταθεί είναι η σύγκριση των δύο αποστάσεων ή την

ρίζα.Αν ο κόμβος περάσει το if κοιτάμε αν αυτός ο κόμβος έχει μικρότερη απόσταση από το σημείο του χρήστη απότι έχει ο nearest.Αν ναι τότε ανανεώνουμε τον nearest κόμβο.Επίσης αν είναι η ρίζα τότε κάνουμε nearest την ρίζα.Μόλις περάσουμε και αυτό το σημείο απλά κοιτάμε αν το επίπεδο του κόμβου είναι μονο η ζυγό για να κοιτάξουμε το x ή y.Αν το x ή y του σημείου του χρήστη είναι μικρότερο από του node που κοιτάμε τότε κοιτάμε πρώτα το αριστερό παιδί στέλνοντας προφανώς το αριστερό παιδί και το παραλληλόγραμμο του, τον μέχρι τώρα κοντινότερο κόμβο και το σημείο του χρήστη αλλιώς κοιταμε πρώτα το δεξί.Και στις δύο περιπτώσεις περνάμε στο δεύτερο παιδί όταν το πρωτο γίνει null η απλά δεν μπει στο if και δεν χρειαστεί να το ψάξουμε.