

Übung 03: Abstrakte Klassen, dynamische Bindung

Abgabetermin: 10.04.2008, 8:15

Name: _____

Matrikelnummer: _____

Informatik: G1 (Prähofer) G2 (Prähofer) G3 (Wimmer) G4 (Wimmer)

WIN: G1 (Ibrahim) G2 (Ibrahim) G3 (Schwinger)

Aufgabe	Punkte	abzugeben schriftlich	abzugeben elektronisch	korr.	Punkte
Übung 3	24	Prosabeschreibung, Java-Programm, Ausgabe eines Testlaufs	Java-Programm	<input type="checkbox"/>	

Übung 03: Quadtree (24 Punkte)

a) Quadtree

Quadtrees sind eine bekannte Technik bei geographischen Anwendungen, um Punkte in einem Gebiet, die heterogen verteilt sind, effizient verwalten zu können. Das Prinzip ist, dass man rechteckige Bereiche solange in 4 gleichgroße Quadranten aufteilt, bis die Menge von Punkten in einem Quadranten klein genug wird. Dadurch werden Bereiche, in denen viele Punkte liegen, feiner zerteilt als Bereiche mit wenig Punkten. Die Abbildung 1 (a) zeigt eine solche Aufteilung. Bei dieser Aufteilung befinden sich in jedem Quadranten auf unterster Ebene maximal 2 Punkte.

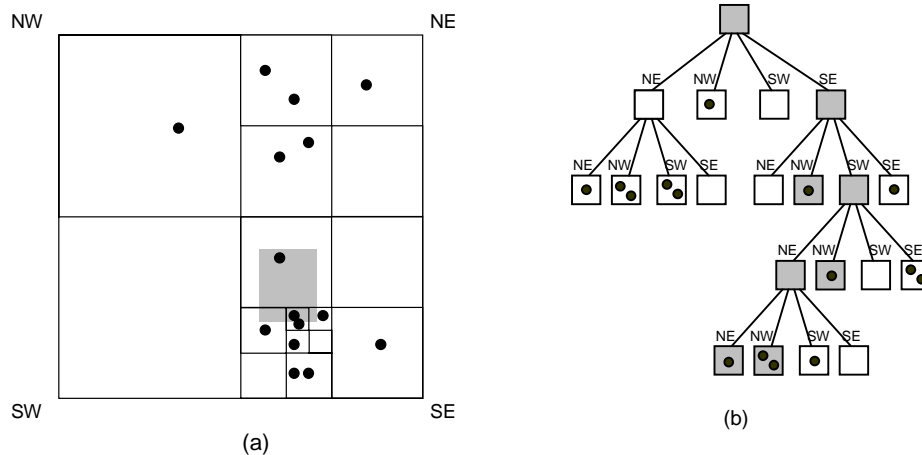


Abbildung 1: Geometrische Aufteilung (a) und Quadtree-Speicherstruktur (b)

Die Speicherform für Quadtrees ist ähnlich binären Bäumen, mit dem Unterschied, dass immer eine Aufteilung in 4 statt 2 Unterknoten erfolgt (Abbildung 1 (b)). Bei jedem Knoten wird sein rechteckiger Bereich gespeichert. Die inneren Knoten haben 4 Unterknoten, jeweils für den nordwestlichen (NW), nordöstlichen (NE), südöstlichen (SE), und südwestlichen Teilquadranten (SW). Die Blätter im Baum speichern schließlich die Punkte.

Mit Quadtrees lässt sich nun die Bereichssuche effizient realisieren. Sollen alle Punkte, die sich in einem gegebenen (rechteckigen) Gebiet befinden, bestimmt werden, geht man folgend vor:

- Beginnend bei der Wurzel im Quadtree untersucht man bei jedem inneren Knoten, welche der untergeordneten Quadranten sich mit dem Suchgebiet schneiden. Für diese Quadranten ruft man rekursiv die Suchmethode auf.
- Gelangt man zu einem Blatt, wird für jeden der beim Blatt gespeicherten Knoten einzeln geprüft, ob er sich innerhalb des Suchgebiets befindet. Ist das der Fall, wird der Punkt in die Ergebnismenge aufgenommen.

Mit diesem Verfahren lässt sich die Suche sehr effizient einschränken. In Abbildung 1 ist in grauer Farbe ein Suchgebiet und die Suche im Baum dargestellt. Man sieht, dass durch die Baumsuche schließlich nur 5 Punkte auf Enthaltensein im Suchgebiet geprüft werden müssen, wovon dann schließlich 2 Punkte als innerhalb des Suchgebiets liegend gefunden werden. Bei großen geographischen Szenen mit vielen Punkten (und kleinen Suchgebieten) führt dieses Verfahren zu einer dramatischen Reduktion der Komplexität.

Sie sollen nun in dieser Übung QuadTrees zur Speicherung von Punktmengen und zur Suche von Punkten für ein rechteckiges Suchgebiet realisieren. Eine Klasse `QuadTree` soll für ein bestimmtes Gesamtgebiet (gegeben als Rechteck) erzeugt werden können und folgende Operationen unterstützen:

- `void insert(Point p)` – Einfügen eines Punktes in den Quadtree
- `Point[] findPoints(Rect r)` – Suchen nach allen Punkten, die innerhalb des Rechtecks liegen

Des Weiteren soll die Ausgabe des QuadTrees mit der Klasse `Window` (siehe Hinweise) möglich sein.

- `void draw()`

Anleitungen:

Für die Realisierung des QuadTrees sind Knoten zu realisieren, wobei zwischen inneren Verzweigungsknoten und Blättern zu unterscheiden ist. Wichtig ist, dass es für beide Arten von Knoten eine gemeinsame Basisklasse gibt, die eine abstrakte Schnittstelle für Knoten definiert.

Blattknoten speichern die Punkte, wobei eine maximale Anzahl von Punkten pro Blatt vorgegeben werden soll.

Soll bei einem Blattknoten ein neuer Punkt eingefügt werden und ist bei diesem die maximale Anzahl von Punkten schon erreicht, muss der Blattknoten durch einen inneren Verzweigungsknoten ersetzt, die derzeit im Blatt gespeicherten Knoten in die neue Knotenstruktur neu eingefügt und dann der neue Knoten hinzu gefügt werden. Folgende Abbildung 2 zeigt das Vorgehen (maximale Anzahl von Knoten pro Blatt ist 2).

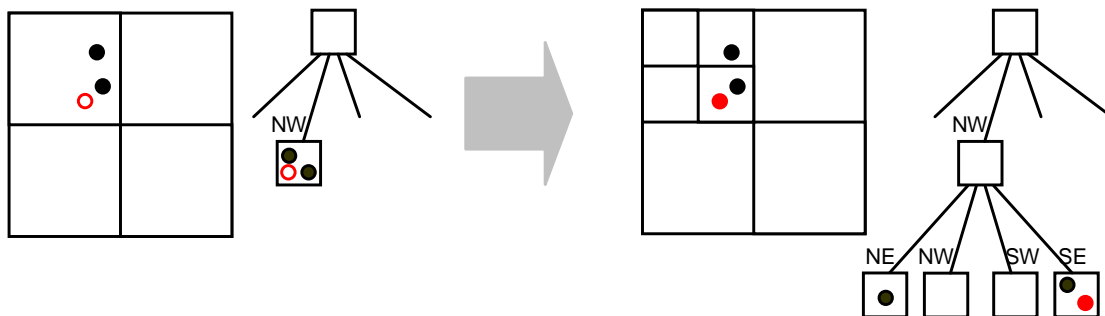


Abbildung 2: Aufspalten eines Quadranten

b) Ausgabe und Test

Es soll möglich sein die geometrische Aufteilung für einen `QuadTree` mit der Klasse `Window` (siehe Hinweise unten) darzustellen. Realisieren Sie dafür in ihren Klassen Methoden `draw()` (analog zu `draw` bei `GraphObject` aus der Übungsstunde).

Testen Sie dann Ihre Implementierung, indem Sie eine Reihe von (zufällig erzeugten) Punkten, in einen `QuadTree` einfügen. Testen Sie damit die Suche nach Punkten in Suchgebieten. Stellen Sie den `QuadTree` und die Suche graphisch dar. Weiter unten ist beispielhaft ein `QuadTree` und eine Suche dargestellt.

Hinweise:

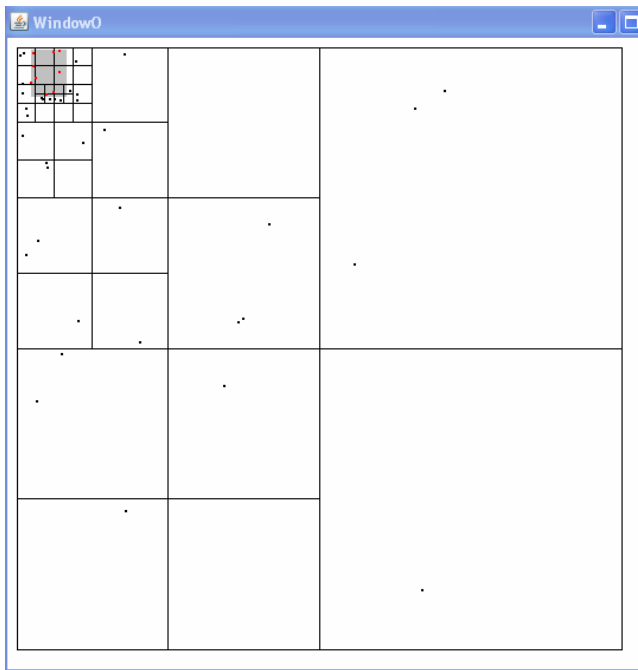
Für die graphische Ausgabe der geometrischen Figuren steht analog zu `Out` eine Klasse `Window` zur Verfügung. Zur Verwendung der Ausgabe muss zuerst die Methode

```
Window.open()
```

aufgerufen werden, mit der ein Fenster geöffnet wird (sonst gibt es beim Ausführen einen Fehler `NullPointerException`). Mit Methoden `Window.drawPoint`, `Window.drawLine`, `Window.drawRectangle` und `Window.drawCircle` kann auf dieses Fenster gezeichnet werden. Analog zu `draw` gibt es bei `Window` `fill`-Operationen, die gefüllte Flächen darstellen.

Mit der Methode `Window.clear()` kann man den Fensterinhalt löschen.

Beispiel einer Programmausgabe:



```

Bitte Suchgebiet eingeben (x y w h): 20 10 30 40
Punkte gefunden:
( 22/ 13)
( 24/ 34)
( 22/ 24)
( 20/ 38)
( 44/ 11)
( 39/ 12)
( 44/ 29)
( 39/ 46)
( 33/ 48)
Bitte Suchgebiet eingeben (x y w h):
    
```