NodeTree

Uiteindelijk willen we niet een List gebruiken, maar een Tree, die we met een TreeIterator kunnen doorlopen.

Maak een class NodeTree die Tree implementeert. Deze class krijgt een private List variabele van het type ArrayList om alle nodes in op te slaan. Noem deze variabele nodeList.

Maak ook alvast de methode add(E node) aan in NodeTree. Nodes die met add() worden toegevoegd komen in de variabele nodeList terecht.

De voorlopig enige andere methode van NodeTree is de methode die een TreeIterator retourneert. Maak hiervan een anonieme inner class. De signatuur van de methode komt er dus als volgt uit te zien:

```
public TreeIterator<E> iterator() {
  return new TreeIterator<E>(){
     // implementeer hierin de methoden van TreeIterator
  };
}
```

De anonieme inner class bevat uiteindelijk veel code. Hier volgen enkele tips bij het maken van deze class.

De iterator

In de volgende stappen gaan we de method public TreeIterator<T> iterator() vast voor het eerste deel implementeren. Deze moet uiteindelijk in staat zijn om bij het doorlopen van alle nodes in de tree te bepalen wat het level van zo'n node is, of het een leaf node is, et cetera (zie de methods in Iterator en TreeItereator. Een deel van de code kunnen we gelukkig delegeren: nodeList is een List, en heeft zelf een iterator() method die een Iterator terug geeft. Daar gaan we gebruik van maken.

- Maak voor het doorlopen van nodes binnen TreeIterator gebruik van een Iterator op de interne variabele nodeList. Bewaar deze in een instantie variabele met de naam iterator, binnen de anonieme inner class. De basisfunctionaliteit van een Iterator: next(), hasNext() en remove(), kan dan (in ieder geval ten dele) worden gedelegeerd naar deze interne variabele.
- Maak vast voor alle methoden van de interface TreeIterator een lege / default implementatie, zodat de class kan worden gecompileerd.
- Sorteer de variabele nodeList bij het aanmaken van de nieuwe TreeIterator, zodat de nodes op de natuurlijke volgorde worden doorlopen (daartoe implementeert Node de interface Comparable). We kunnen daar geen constructor voor gebruiken, omdat het een anonieme inner class is. Gebruik daarom een instance initializer. “Ververs” daarna de iterator variabele (maak de iterator opnieuw aan) , zodat het doorlopen van de nodes via de iterator overeenkomt met de volgorde van de nodes in nodeList.
- Maak gebruik van de variabele iterator bij het implementeren van de standaard Iterator methoden (TreeIterator is immers een sub-interface van Iterator).
- Vervang in TreeApp de gebruikte List door een Tree van het type NodeTree. Test alle volgende stappen daarna in TreeApp.

Het level

In de volgende stappen kunnen we van de huidige node bepalen wat het level is: hoever is deze node van de root verwijderd?
Om in de level method en in andere methods van de iterator te kunnen verwijzen naar de huidige node, bewaren we deze huidige node als instance variable van de TreeIterator.

- Bewaar in de TreeIterator na elke aanroep van next() de opgehaalde node in een instantie variabele met de naam current.
- De methode level geeft 0 als de variabele current geen parent heeft. Als current wel een parent heeft, tel dan het aantal keer dat van die parent weer de parent opgehaald kan worden.
- Toon ter controle in TreeApp alle nodes met hun level.

De start node

- Als startWith(E node) wordt gebruikt, kunnen we maar een deel van de verzameling doorlopen: alleen het deel dat uiteindelijk onder de opgegeven node valt. Maak daarvoor binnen startWith() een nieuwe List aan, en stop daarin alleen de nodes die meedoen (zie ook volgende tip). Vervang daarna het attribuut iterator door de Iterator van de gevulde List. Bewaar ook de opgegeven start node als instantie variabele.
- Maak binnen startWith() gebruik van een private boolean methode descendantOfStartWith(E node). Deze geeft true als de opgegeven node onder de opgegeven start-node valt, en anders false.
- De methode startWith() mag niet worden aangeroepen na de eerste call van next(). Controleer hierop. Welke exception is voor dit soort situaties bedoeld?
- Breidt de methode level() uit: als startWith() is aangeroepen, wordt de opgegeven start node beschouwd als level 0

remove

- De methode remove mag alleen worden aangeroepen na een aanroep van next(). Lees de API documentatie om te bepalen welke exception anders moet worden opgeworpen.
- Houdt er rekening mee dat de nodes mogelijk via een lokale iterator worden doorlopen (namelijk na een aanroep van startWith()). In dat geval heeft een aanroep van remove() geen invloed op de omliggende class.
- Een node mag alleen worden verwijderd als het een leaf node is. Maak daarvoor gebruik van de methode isLeaf() van de current node (deze geeft nu nog true, en zal later worden geïmplementeerd). Geef ook een duidelijke foutmelding als remove() wordt aangeroepen met een andere node.

Bespreek de opdrachten met de docent.