

4

Binaire bomen en zoekbomen - Verdieping

In dit hoofdstuk vind je verdiepingsoefeningen bij de les 'Bomen' enerzijds en de les 'Binaire zoekbomen' anderzijds. Er zijn zowel theorie- als praktijkoefeningen. De praktijkoefeningen zijn vaak extra methodes die je aan de domain-klassen van de vorige hoofdstukken kan toevoegen.

Oefening 4.1



(Examenvraag augustus 2019) Je weet wat een complete binaire boom is. Een *strikt* binaire boom wordt gedefinieerd als een binaire boom waar elke knoop, met uitzondering van de bladeren, *exact twee* kinderen heeft. Je krijgt nu 5 stellingen over beide bomen. Welke van de 5 stellingen is juist? Geef bij elke stelling die je af- of goedkeurt een duidelijke uitleg mbv een figuur.

1. Elke binaire boom is ofwel compleet ofwel strikt.
2. Elke complete binaire boom is ook een strikt binaire boom.
3. Elke strikt binaire boom is ook een complete binaire boom.
4. Een binaire boom kan nooit tegelijk compleet en strikt zijn.
5. Geen enkele van bovenstaande zinnen is juist.

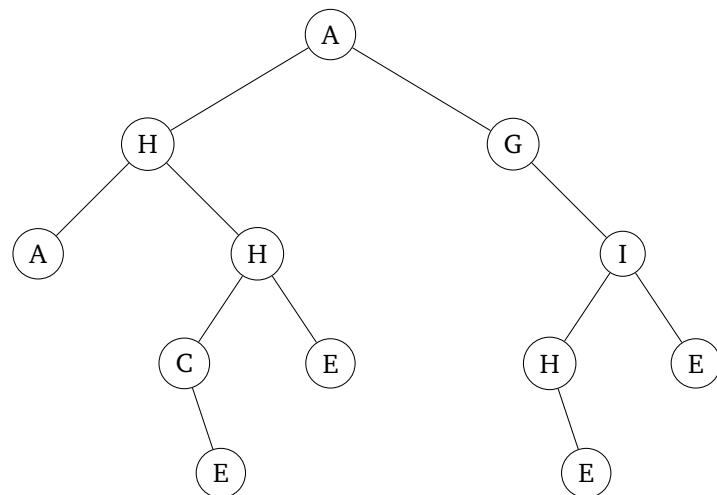
Oefening 4.2



Gegeven de implementatie van een binaire boom zoals in les 2 gegeven. De bedoeling van deze oefening is het aantal voorkomens van een gegeven data-veld in een binaire boom te tellen.

- a) Implementeer een methode `count` in de `BinaryTree` klasse die het aantal voorkomens van een gegeven data-veld telt in een boom.
- b) Maak een nieuwe `BinaryTreeDriver` klasse waarin je de boom overeenkomstig figuur 4.1 maakt.

4 Binaire bomen en zoekbomen - Verdieping




Figuur 4.1 Binaire boom met herhaalde datavelden

- c) Run de main functie in de `BinaryTreeDriver` klasse en controleer hiermee de implementatie van de count methode met respectievelijke parameters: I, A, H, E en Q. Verwachte uitvoer:

Aantal voorkomen van I = 1
Aantal voorkomens van A = 2
Aantal voorkomens van H = 3
Aantal voorkomens van E = 4
Aantal voorkomens van Q = 0

Oefening 4.3

 (Examen augustus 2019) Een preorder wandeling in een binaire zoekboom (BST) met in de knopen gehele getallen levert als resultaat volgende rij: 30, 20, 10, 15, 25, 23, 39, 35, 42. Geef het resultaat van een postorder wandeling in dezelfde BST.



Oefening 4.4

 (Examen augustus 2019) Men zoekt in een BST van gehele getallen naar het getal 43. Men doorloopt zo een aantal knopen en het zesde getal is 43. Welke van volgende rijen van getallen zijn *niet* mogelijk als lijst van opeenvolgende knopen die men bekomt als resultaat van de zoektocht? Geef een duidelijke uitleg waarom je antwoord(en) niet kan (kunnen).

1. 61, 52, 14, 17, 40, 43

2. 2, 3, 50, 40, 60, 43
3. 10, 65, 31, 48, 37, 43
4. 81, 61, 52, 14, 41, 43
5. 17, 77, 27, 66, 18, 43

Oefening 4.5



Zoek de knopen op een bepaalde afstand.

- a) Schrijf een methode `getNodesAtDistance(k)` in de `BinaryTree` klasse die een lijst teruggeeft van de datavelden die op een afstand k van de wortel van de binaire boom verwijderd zijn. In de boom uit figuur 4.1 zijn A, H en I de datavelden van knopen die op een afstand 2 van de root verwijderd zijn.
- b) Test je implementatie uit door in de klasse `BBDriver` de methode `getNodesAtDistance` op te roepen met parameters 0, 1, 2, 3 en 4. Verwachte uitvoer:
 Datavelden van knopen verwijderd op een afstand van 0 van de root = [A]
 Datavelden van knopen verwijderd op een afstand van 1 van de root = [H, G]
 Datavelden van knopen verwijderd op een afstand van 2 van de root = [A, H, I]
 Datavelden van knopen verwijderd op een afstand van 3 van de root = [C, E, H, E]
 Datavelden van knopen verwijderd op een afstand van 4 van de root = [E, E]

Oefening 4.6



Gegeven volgende code:






Listing 1 Mystery methode

```
public ArrayList<E> mystery() {
    ArrayList<E> lijst = new ArrayList<>();
    if (this.leftTree != null) lijst.add(this.leftTree.data);
    if (this.rightTree != null) lijst.add(this.rightTree.data);
    return lijst;
}

public ArrayList<E> mystery(int g) {
    if (g == 1) {
        return this.mystery();
    } else {
        ArrayList<E> links = new ArrayList<>();
        if (this.leftTree != null) links = this.leftTree.mystery(g - 1);
        ArrayList<E> rechts = new ArrayList<>();
        if (this.rightTree != null) rechts = this.rightTree.mystery(g - 1);
        links.addAll(rechts);
    }
}
```

```
    return links;
  }
}
```

Geef nu de uitvoer van volgende oproepen van deze methode voor de boom uit figuur 4.1:

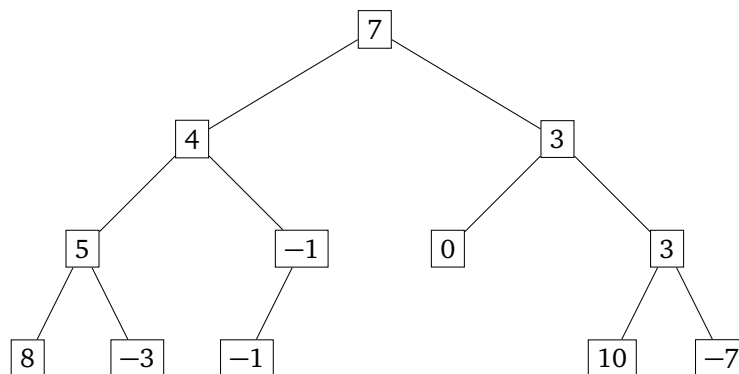
- a) boom.mystery(1) 
- b) boom.mystery(2) 
- c) boom.mystery(3) 
- d) boom.mystery(4) 
- e) boom.mystery(5) 

Oefening 4.7




(Examen juni 2017) Schrijf een methode `kinderSom()`. Het resultaat is een boolean. De methode geeft `true` terug als de boom waarop ze toegepast wordt voldoet aan de eigenschap dat elke knoop als waarde de som van zijn kinderen heeft (dat aantal kinderen kan natuurlijk 2, 1 of 0 zijn). Figuur 4.2 op pagina 20 toont een boom die aan deze eigenschap voldoet.

Je zal voor deze methode een nieuwe klasse moeten schrijven. Je kan immers niet zomaar de waarde van twee knopen van een nog te bepalen type `E` optellen, bvb. als je voor `E` het type `String` kiest! Beperk je dus voor deze oefening tot een nieuw soort binaire boom, nl. één waarbij het veld `data` een `int` is. Test je methode uit door de boom van figuur 4.2 te implementeren in een `main`methode en het resultaat van de toepassing van deze methode op deze boom naar de console te schrijven.



Figuur 4.2 Binaire boom voldoet aan `kinderSom`

Oefening 4.8


 (Examen juni 2019) Een binaire boom heeft een letter als waarde in elke knoop. Als je de boom *in-order* doorwandelt krijg je 'FHCADEGB'. Een *post-order* wandeling levert voor dezelfde boom 'FCHAGEBD'. Teken hieronder deze boom.

Oefening 4.9

 De preorder methode van een BST geeft volgende uitvoer: 30, 20, 10, 15, 25, 23, 39, 35, 42. Geef de postorder uitvoer.




Oefening 4.10

 We wensen een BST te bouwen bestaande uit de gehele getallen 1 tot en met 10. In welke volgorde moeten de getallen worden toegevoegd opdat de resulterende boom compleet is?




Oefening 4.11

 Stel dat 7, 5, 1, 8, 3, 6, 0, 9, 4 en 2 worden toegevoegd aan een lege BST. Wat is de diepte van de resulterende boom?




Oefening 4.12

 Stel dat we een BST maken door vertrekkende van een lege BST vervolgens de getallen 71, 65, 84, 69, 67 en 83 toe te voegen. Welke zijn de data-velden van de bladeren van de boom?




Oefening 4.13

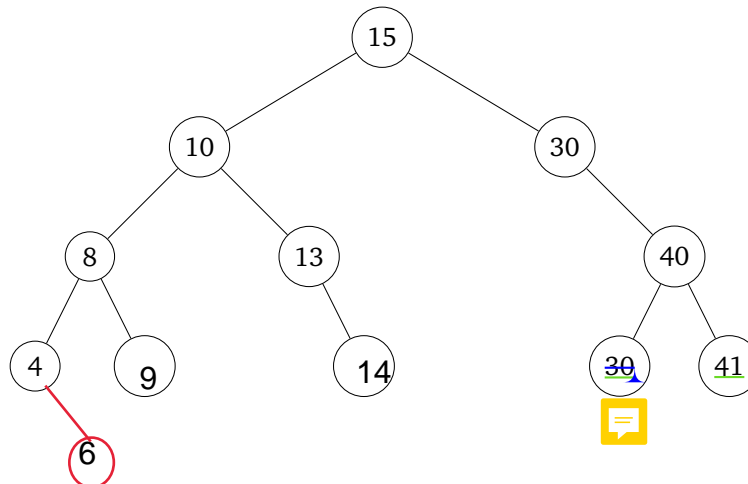
 Stel dat boom een BST is bestaande uit gehele getallen groter dan of gelijk aan 1 en kleiner dan of gelijk aan 100. Welke van de volgende paden kan (kunnen) niet?

- a) 10, 75, 64, 43, 60, 57, 55
- b) 90, 12, 68, 34, 62, 45, 55
- c) 9, 85, 47, 68, 43, 57, 55
- d) 79, 14, 72, 56, 16, 53, 55

Oefening 4.14

 (Examenvraag juni 2017)


1. Gegeven de binaire zoekboom (BST) in figuur 4.3.
 - Vul een geheel getal in voor de ontbrekende knopen.
 - Verbeter eventuele fouten (door enkel aanpassingen te doen aan de bladeren!), zodat de BST eigenschap voor deze boom volledig klopt.



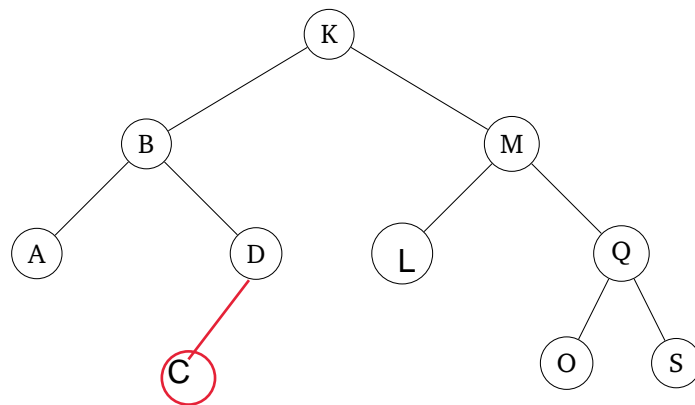
Figuur 4.3

2. Voeg aan de BST van figuur 4.3 een knoop met het getal 6 toe op de juiste plaats.
3. Vertrek nu van de boom die je in het vorige puntje bekwam. Teken hieronder de nieuwe toestand van de BST (teken de volledige boom) na verwijderen van de knoop met het getal 10.

Oefening 4.15

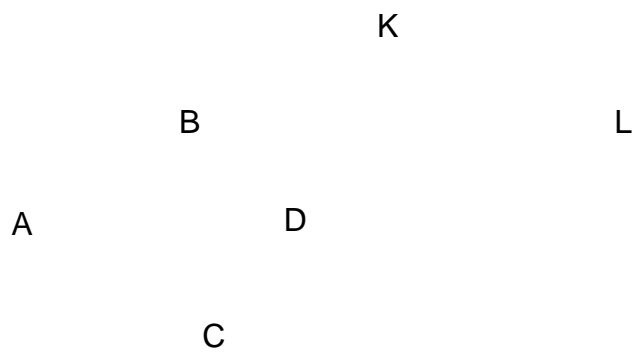
 (Examenvraag augustus 2017)

1. Gegeven de binaire zoekboom (BST) in figuur 4.4. Vul een letter in voor de ontbrekende knoop, zodat de BST eigenschap voor deze boom nog klopt.
2. Voeg in figuur 4.4 aan de BST een knoop met dataveld C toe op de juiste plaats.



Figuur 4.4

3. Teken de nieuwe toestand van de BST (teken de volledige boom, ook met dataveld C) na verwijderen van de knoop met het dataveld M.



Oefening 4.16



(Examenvraag juni 2018) Gegeven de volgende java-klasse:

```
package domain;
public class Persoon implements Comparable<Persoon>{
    private String naam, voornaam;
    private int lengte;
    private double gewicht;
    public Persoon(String voornaam, String naam, int lengte, double gewicht) {
        this.setVoornaam(voornaam);
        this.setNaam(naam);
        this.setLengte(lengte);
        this.setGewicht(gewicht);
    }

    private void setNaam(String naam) {
        if (naam == null || naam.trim().isEmpty()) throw new
            IllegalArgumentException();
        this.naam = naam;
    }

    private void setVoornaam(String voornaam) {
        if (voornaam == null || voornaam.trim().isEmpty()) throw new
            IllegalArgumentException();
        this.voornaam = voornaam;
    }

    private void setLengte(int lengte) {
        if (lengte <= 30) throw new IllegalArgumentException();
        this.lengte = lengte;
    }

    private void setGewicht(double gewicht) {
        if (gewicht <= 25) throw new IllegalArgumentException();
        this.gewicht = gewicht;
    }

    public int getBMI() {
        return (int)(Math.round(gewicht * 10000 / (lengte * lengte)));
    }

    @Override
    public String toString() {
        return voornaam + " " + naam + " BMI = " + this.getBMI();
    }

    @Override
    public int compareTo(Persoon o) {
        if (o == null) return 1;
        else {
            int i = this.getBMI() - o.getBMI();
            if (i != 0) return i;
            else {
```



```

        i = this.naam.compareTo(o.naam);
        if (i != 0) return i;
        else return this.voornaam.compareTo(o.voornaam);
    }
}
}
}

```

1. Volgende objecten worden aan een BST<Persoon> vervolgens toegevoegd:

```

els = new Persoon("Els", "Adams", 176, 86) //bmi = 28
an = new Persoon("Anne", "Janssen", 176, 68) //bmi = 22
tom = new Persoon("Tom", "Frederiks", 185, 105) // bmi = 31
tim = new Persoon("Tim", "Anders", 185, 85) //bmi = 25
joke = new Persoon("Joke", "Alders", 176, 86) //bmi = 28

```

Teken de resulterende boomstructuur

2. Hoeveel objecten zouden aan de boom uit het vorige puntje *minimaal* moeten toegevoegd worden om deze *compleet* te maken? Geef een concreet voorbeeld (zoals in bovenstaande code) voor elk van deze objecten. **Voeg daarna in een andere kleur je nieuwe objecten toe aan bovenstaande boom.**
3. Stel dat de compareTo-methode in de klasse Persoon er als volgt had uitgezien:

```

@Override
public int compareTo(Persoon o) {
    if ( o == null) return 1;
    else return this.getBMI() - o.getBMI();
}

```

Vertrek opnieuw van een lege boom. Teken de boomstructuur na het toevoegen van de objecten els, an, tom, tim en joke gegeven deze compareTo-methode

Oefening 4.17



(Examen augustus 2018)

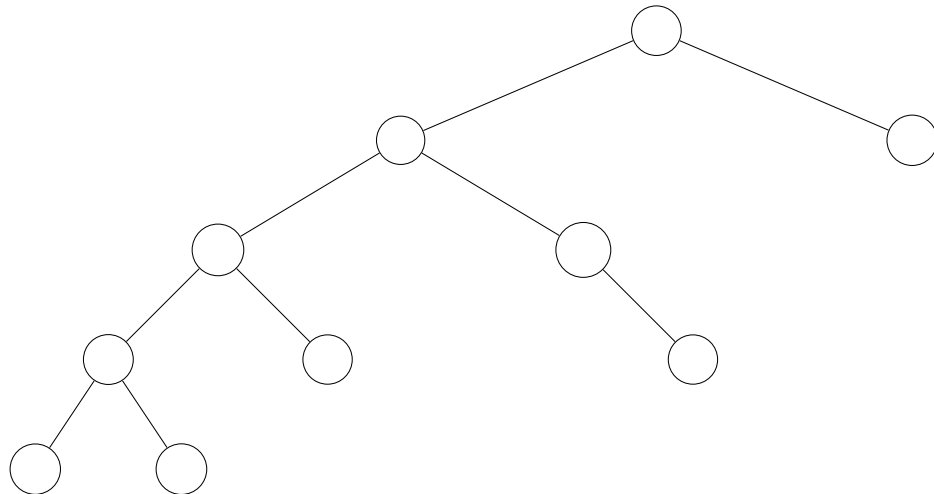
1. Gegeven een complete binaire boom met als datavelden gehele getallen. Een postorderwandeling doorheen de boom levert als opeenvolgende getallen: -5, 4, 7, 9, 8, 2, 0, 11, 3, 5. Teken deze boom.
2. Gegeven de getallen 3, 9, -4, 6, 10, -3, -8, 5. In welke volgorde moet je deze 8 getallen toevoegen aan een lege boom zodat het resultaat een complete binaire zoekboom is? Geef de getallen in de juiste volgorde en teken de BST. Is deze volgorde uniek of zijn er meerdere mogelijke volgordes die dezelfde complete BST geven + leg uit?

Oefening 4.18

 (Examen juni 2019)

In onderstaande binaire zoekboom met diepte 5 worden studentendossiers opgeslagen. De dossiers zijn sorteerbaar op familienaam. Er is een dossier voor volgende studenten: Aerts, Bériot, Colins, Decoster, Eerdeken, Lints, Mellaerts, Smolders, Tobback, Vansina. Stel deze namen verkort voor met hun eerste letter: A, B, ... , V.

1. Vul de eerste letter van de 10 namen op de juiste plaats in de BST van figuur 4.5.



Figuur 4.5 BST voor studentendossiers

2. Geef de/een (zie volgende vraag) volgorde waarin deze 10 dossiers moeten toegevoegd worden.
3. Is er een andere invoervolgorde mogelijk? Zo ja, geef er één. Zo nee, leg uit waarom er maar één volgorde kan zijn om deze BST te vullen.
4. (Deze en de volgende vragen kan je pas beantwoorden als de leerstof van heaps gezien is) Stel dat we deze 10 studentendossiers niet in een BST maar in een binaire max-heap bijhouden. Teken deze binaire max-heap.
5. Geef de arrayvoorstelling van deze max-heap.
6. Stel dat de binaire max-heap diepte 13 moet hebben. Reken uit hoeveel studentendossiers je dan *minimaal* zou moeten toevoegen bij de boom uit vraag (d). Je mag je berekeningen altijd aan de achterzijde van een blad maken. Verwijs er dan wel duidelijk naar ("zie achterzijde pg ...").

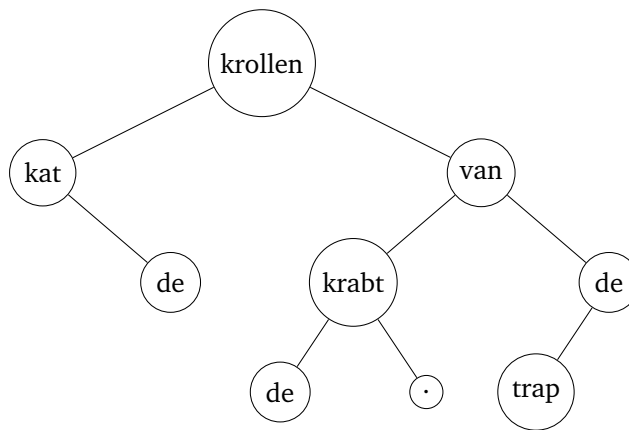
Oefening 4.19

 (Examen augustus 2018) Wandelen door een boom en BST-oefening.

1. Gegeven een complete binaire boom met als datavelden gehele getallen. Een postorder-wandeling doorheen de boom levert als opeenvolgende getallen: $-5, 4, 7, 9, 8, 2, 0, 11, 3, 5$. Teken deze boom.
2. Gegeven de getallen $3, 9, -4, 6, 10, -3, -8, 5$. In welke volgorde moet je deze 8 getallen toevoegen aan een lege boom zodat het resultaat een complete binaire zoekboom is? Geef de getallen in de juiste volgorde en teken de BST. Is deze volgorde uniek of zijn er meerdere mogelijke volgordes die dezelfde complete BST geven + leg uit?

Oefening 4.20

 (Examen augustus 2017) Gegeven de boom uit figuur 4.6.



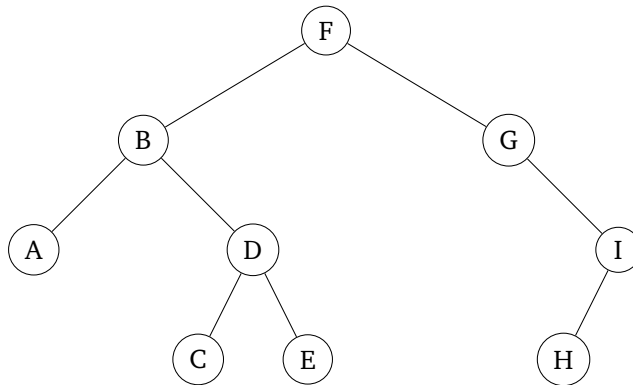
Figuur 4.6 Boom 1

1. Geef de resulterende zin na een inorder-wandeling door de boom van figuur 4.6.
2. Gegeven de zin in de vorm van een array van Strings: `String[] {"Als", "de", "kat", "van", "huis", "is", "dansen", "de", "muizen", "altijd", "op", "tafel"}`. We kunnen deze zin omzetten naar een binaire boom bestaande uit knopen met als dataveld de individuele elementen van deze String-array waarbij een *postorder*-wandeling door deze boom de oorspronkelijke zin teruggeeft. Teken deze boom gegeven dat de boom *compleet* moet zijn.

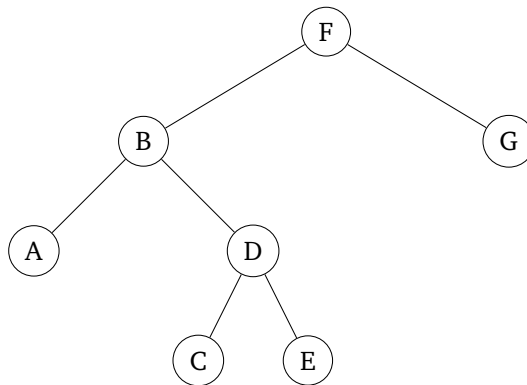
Oefening 4.21



(Examen augustus 2017) Herneem de domainklasse van Binary Tree. Schrijf in deze klasse een methode `deelZonder(E wortelInfo)` die een nieuwe `BinaryTree<E>` teruggeeft die gelijk is aan deze boom zonder de deelboom met als wortel de knoop met dataveld `wortelInfo`. Deze methode geeft `null` terug indien de parameter niet voorkomt in deze binary tree.



wordt na het oproepen van de methode `deelZonder(I)`:



UI-klasse:

```
public class BinaryTreeDriver {  
  
    public static void main(String[] args) {  
        BinaryTree<String> nodeA = new BinaryTree<>("A");  
        BinaryTree<String> nodeC = new BinaryTree<>("C");  
        BinaryTree<String> nodeE = new BinaryTree<>("E");  
        BinaryTree<String> nodeH = new BinaryTree<>("H");  
        BinaryTree<String> nodeD = new BinaryTree<>("D", nodeC, nodeE);  
        BinaryTree<String> nodeB = new BinaryTree<>("B", nodeA, nodeD);  
        BinaryTree<String> nodeI = new BinaryTree<>("I", nodeH, null);  
        BinaryTree<String> nodeG = new BinaryTree<>("G", null, nodeI);  
        BinaryTree<String> boom = new BinaryTree<>("F", nodeB, nodeG);  
    }  
}
```

```

        System.out.println("\nVolledige boom preorder:");
        boom.printPreorder();
        System.out.println("\nVolledige boom inorder:");
        boom.printInOrder();

        BinaryTree<String> boomZonderI = boom.deelZonder("I");
        System.out.println("\nBoom zonder I preorder: ");
        boomZonderI.printPreorder();
        System.out.println("\nBoom zonder I inorder: ");
        boomZonderI.printInOrder();

        BinaryTree<String> boomZonderB = boom.deelZonder("B");
        System.out.println("\nBoom zonder B preorder: ");
        boomZonderB.printPreorder();
        System.out.println("\nBoom zonder B inorder: ");
        boomZonderB.printInOrder();
    }
}

```

geeft volgende uitvoer:

```

Volledige boom preorder:
F B A H C E G I H
Volledige boom inorder:
A B C H E F G H I
Boom zonder I preorder:
F B A H C E G
Boom zonder I inorder:
A B C H E F G
Boom zonder B preorder:
F G I H
Boom zonder B inorder:
F G H I

```

Oefening 4.22



(Examen juni 2018) We noemen een binaire boom *strikt* als alle knopen ofwel 0 ofwel 2 (dus niet 1!) kinderen hebben. Schrijf een methode `isStrikt(): boolean` die van een gegeven binaire boom nakijkt of hij strikt is.

Vertrek van de klasse `BinaryTree<E>` die we definieerden in de les:

```

package domain;

import java.util.ArrayList;

public class BinaryTree<E> {

```

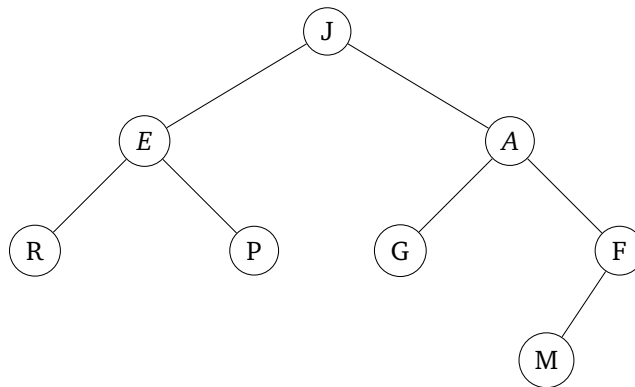
4 Binaire bomen en zoekbomen - Verdieping

```
private E data;
private BinaryTree<E> leftTree, rightTree;

public BinaryTree(E data){
    this(data,null,null);
}

public BinaryTree(E data, BinaryTree<E> leftTree, BinaryTree<E> rightTree){
    this.data= data;
    this.leftTree= leftTree;
    this.rightTree= rightTree;
}
}
```

Maak in het package ui een main methode in het bestand `BinaryTreeDriver.java` om de boom uit figuur 4.7 te maken.



Figuur 4.7 Eenvoudige binaire boom van gehele getallen

Zorg dat je main methode volgend resultaat in de console toont:

```
volledige binaire boom strikt? -> false
binaire boom met enkel node R strikt? -> true
binaire boom met node E en kinderen R en P strikt? -> true
```

Oefening 4.23



(Examen juni 2019) Gegeven een BST zoals we die implementeerden in de les. Schrijf een methode `geefKnopenBinnenInterval(min:E, max:E): ArrayList<E>` die een arraylist geeft van alle knopen in stijgende volgorde en gelegen binnen het interval `[min,max]`.

UI klasse:

```

public static void main(String[] args) {
    BinarySearchTree<Integer> boom = new BinarySearchTree<>();
    boom.addNode(6);
    boom.addNode(4);
    boom.addNode(8);
    boom.addNode(3);
    boom.addNode(5);
    boom.addNode(7);
    boom.addNode(9);

    printBoomInfo(boom);

    System.out.println("\nKnopen tussen 5 en 8: "+boom.geefKnopenBinnenInterval(5,8));
    System.out.println("\nKnopen tussen 3 en 5: "+boom.geefKnopenBinnenInterval(3,5));
    System.out.println("\nKnopen tussen 8 en 9: "+boom.geefKnopenBinnenInterval(8,9));
    System.out.println("\nKnopen tussen -10 en 0: "+boom.geefKnopenBinnenInterval(-10,0));
    System.out.println("\nKnopen tussen 100 en 110: "+boom.geefKnopenBinnenInterval(100,110));
}

```

geeft volgende uitvoer:

Knopen tussen 5 en 8: [5, 6, 7, 8]

Knopen tussen 3 en 5: [3, 4, 5]

Knopen tussen 8 en 9: [8, 9]

Knopen tussen -10 en 0: []

Knopen tussen 100 en 110: []