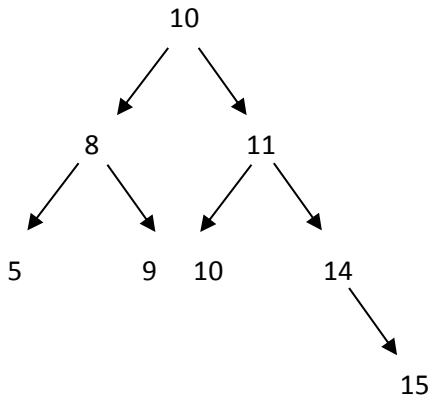
Oefening 4: Binaire bomen

In dit labo ontwikkelen we een pakket dat binaire bomen implementeert. De volgende tekening is een voorbeeld van een binaire boom.



Een binaire boom bestaat uit een aantal elementen, die *knopen* genoemd worden. Er zijn twee soorten knopen: *interne knopen* en *bladeren*.
Op de figuur zijn de interne knopen de knopen van waaruit er pijlen vertrekken, en de bladeren zijn de knopen waarin er enkel maar pijlen aankomen. Vanuit elke interne knoop vertrekken er pijlen naar exact twee andere knopen: samen worden ze de kinderen van deze knoop genoemd; de ene heet het *linkerkind* en de andere het *rechterkind*.
Een binaire boom heeft precies één knoop waarin geen pijlen aankomen. Deze knoop wordt de *wortel* van de boom genoemd.
Elke interne knoop van een boom heeft een *inhoud (*dit is een getal*)* en een *linker- en rechterkind*. Deze kinderen kunnen *'leeg'* zijn (In bovenstaande tekening, heeft de interne knoop met als inhoud 14 bijvoorbeeld geen linkerkind).
Bladeren hebben ook een *inhoud (*een getal*)*.

Implementeer de klassen ***Boom, Knoop, InterneKnoop*** en ***Blad*** die overeenkomen met bovenstaande uitleg. De klasse *Boom* doet niet veel meer dan enkel maar een wortel bijhouden.
Al deze klassen moeten zinvolle constructoren aanbieden en "getters" en "setters" voor hun attributen.
Probeer in een main methode een boom te construeren.

Uitbreiding :
Maak ook een *toString* methode. Het is niet zo eenvoudig om een mooie voorstelling voor zo'n boom te bedenken. Wij stellen de volgende voor:

```
-Knoop 10:
      -Knoop 8:
            -Blad 5
            -Blad 9
      -Knoop 11:
            -Blad 10
            -Knoop 14:
                  -Leeg blad
                  -Blad 15
```

Je kan dit klaarspelen met behulp van een extra functie toStringMetSpaties(int i) die een voorstelling van een knoop produceert, voorafgegaan door i spaties. Een interne knoop doet dan bijvoorbeeld dit:

- Hij produceert i spaties, dan de tekst “-Knoop” gevolgd door zijn inhoud, een dubbelpunt, en een “\n”.

- Dan voegt hij daaraan toe de toStringMetSpaties(i+2)-waarde van zijn linkerkind, gevolgd door de toStringMetSpaties(i+2)-waarde van zijn rechterkind.

Wat een blad moet afprinten, is tamelijk voor de hand liggend. Als bladeren en interne knopen hierbij gemeenschappelijke dingen moeten doen, kunnen deze natuurlijk best in hun superklasse *Knoop* terechtkomen.