Träd

Tema: Träd, binära träd

U 1. Jämför följande två förslag (i pseudokod) att räkna antal noder i ett binärt träd med roten n. Är de korrekta eller ej. Kan de förenklas?

a)
```
count(n)
    om n == null return 0
    annars
        om n.left == null && n.right == null return 1
        annars
            om n.left == null return 1 + count(n.right)
            annars
                om n.right == null return 1 + count(n.left)
                annars return 1 + count(n.left) + count(n.right)
```

b)
```
count(n)
    om n == null return 0
    annars return 1 + count(n.left) + count(n.right)
```

U 2. Antag att vi har följande klass som representerar ett binärt träd:

```
public class BinaryTree<E> {
    private Node<E> root;

    // övriga metoder

    /** Skriver ut trädets noder i inorder. */
    public void printInorder() {
        printInorder(root);
    }

    private void printInorder(Node<E> n) {
        ...
    }

    private static class Node<E> {
        private E element;
        private Node<E> left;
        private Node<E> right;

        private Node(E data) {
            this.element = data;
            left = right = null;
        }
    }
}
```

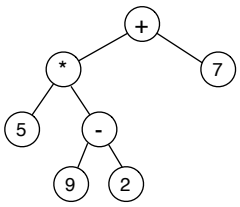
a) Implementera färdigt den privata, rekursiva metoden `printInorder`.

b) Hur förändras lösningen till uppgift a om man istället vill skriva ut elementen i preorder respektive postorder?

U 3. a) Lägg till en metod `nbrLeaves()` i klassen `BinaryTree` som returnerar antal löv i trädet. Låt den publika metoden `nbrLeaves` anropa en privat rekursiv metod `nbrLeaves(Node<E> n)` som returnerar antal löv i det träd där n är rot.

b) Lös samma uppgift som i deluppgift a men placera den rekursiva metoden i nodklassen.

U 4. Ett aritmetiskt uttryck som består av tal och de fyra binära operatorerna +, -, * och / kan representeras av ett binärt träd där löven innehåller talen och övriga noder innehåller operatorer. Exempel:

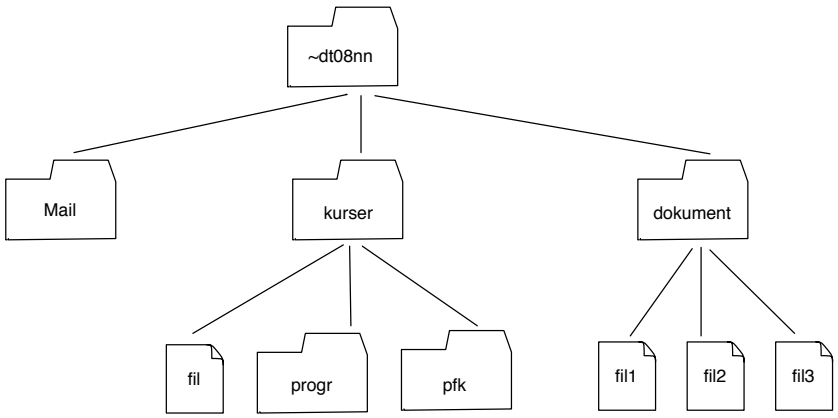


Följande klasser representerar ett binärt uttryckträd. För enkelhets skull har vi i den här uppgiften låtit både operatorer och heltal representeras av en teckensträng.

```
public class ExprTree {
    private ExprNode root; // refererar till roten i trädet

    // konstruktor och övriga metoder

    /** Returnerar en teckensträng som representerar uttrycket. Teckensträngen
        innehåller parenteser runt alla deluttryck, utom runt talen. */
    public String fullParen();

    private static class ExprNode {
        private String element;     // nodens innehåll
        private ExprNode left;      // refererar till vänster barn
        private ExprNode right;     // refererar till höger barn

        // konstruktor och övriga metoder
    }
}
```

Implementera metoden `fullParen` med rekursiv teknik. Om uttryckträdet ser ut som i exemplet ska följande teckensträng returneras: `((5*(9-2))+7)`

U 5. Här är ett annat exempel på ett träd. En katalog (eng: directory) är en fil som kan innehålla andra filer och kataloger. Se exempel:



Din uppgift är att implementera metoden

```
/** Returnerar en lista med alla vanliga filer i file som är större än
    size bytes. */
```

```
public static List<File> biggerThan(File file, int size) {
```

`File` är en klass i Java som representerar en fil eller en katalog. Klassen har bland annat följande metoder:

```
/** Returnerar true om filen är en vanlig fil. */
boolean isFile();

/** Returnerar true om filen är en katalog. */
boolean isDirectory();

/** Returnerar längden (i bytes) om aktuell fil är en vanlig fil. I annat
    fall returneras ett ospecificerat värde.
long length();

/** Returnerar en vektor med filer om aktuell fil är en katalog. Om aktuell
    fil är en vanlig fil returneras null.
File[] listFiles();
```

U 6. Beskriv hur man kan gå tillväga för att behandla noderna i ett binärt träd nivå- för nivå (dvs. först roten, sedan rotens barn, barnbarn ...)? Tips! Använd en kö.