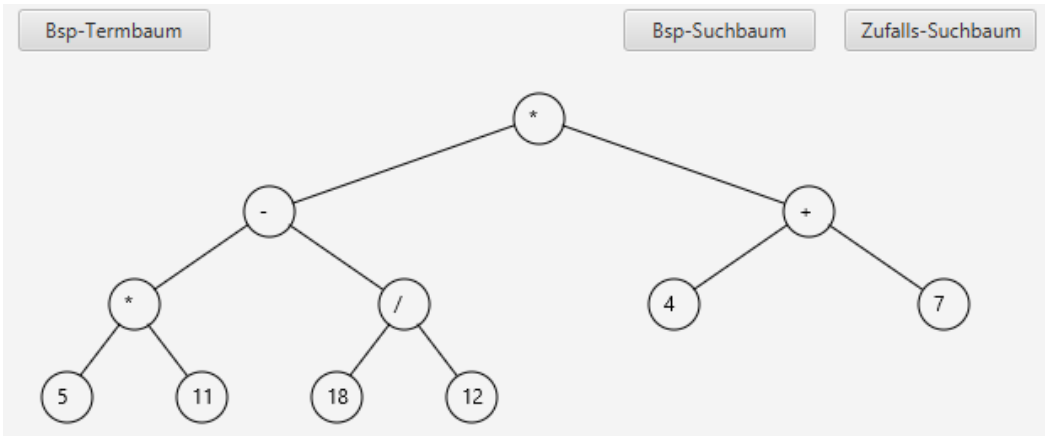
Termbaum - Anwendung BinaryTree (über Vererbung)

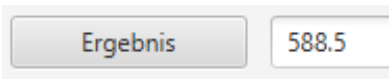
Klasse Termbaum

```
public class Termbaum extends BinaryTree <String> {
 public Termbaum() { super(); }
 public Termbaum(String pContent) { super(pContent); }
 public Termbaum(String pContent, BinaryTree <String> pLeftTree, BinaryTree <String> pRightTree) {
  super(pContent, pLeftTree, pRightTree);
 }
 public String infix() {
  if (!isEmpty()) {
   if (getContent().equals("+") || getContent().equals("-") ||
       getContent().equals("*") || getContent().equals("/")) {
     return "(" + ((Termbaum) getLeftTree()).infix() + getContent() + ((Termbaum) getRightTree()).infix() + ")";
   } else return "" + getContent();
  } else return "";
 }
 public double Ergebnis() {
  if (getContent().equals("+"))
  return ((Termbaum) getLeftTree()).Ergebnis() + ((Termbaum) getRightTree()).Ergebnis();
  else if (getContent().equals("-"))
  return ((Termbaum) getLeftTree()).Ergebnis() - ((Termbaum) getRightTree()).Ergebnis();
  else if (getContent().equals("*"))
  return ((Termbaum) getLeftTree()).Ergebnis() * ((Termbaum) getRightTree()).Ergebnis();
  else if (getContent().equals("/"))
  return ((Termbaum) getLeftTree()).Ergebnis() / ((Termbaum) getRightTree()).Ergebnis();
  else return Double.parseDouble(getContent());
 }
}
```



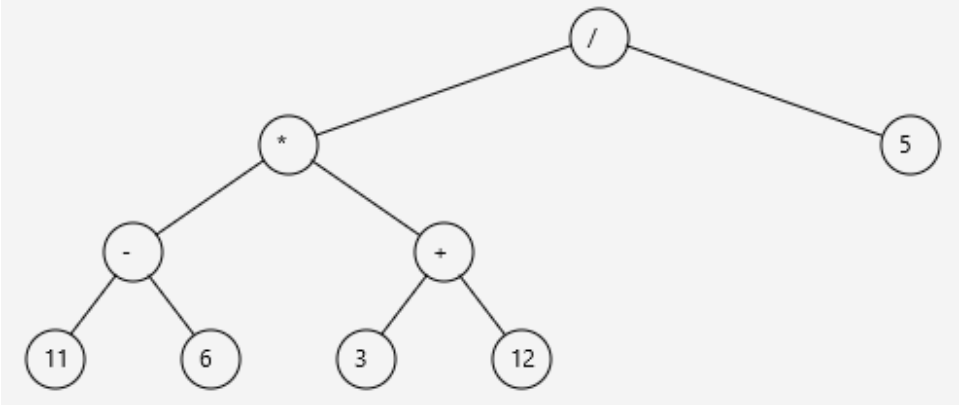
Extras in Controller-Klasse

```
public void btInfix_onClick() {
 if (Baum instanceof Termbaum) tfInfix.setText(((Termbaum) Baum).infix());
 else tfInfix.setText("nur Termbäume!");
}
public void btErgebnis_onClick() {
 if (Baum instanceof Termbaum)
   tfErgebnis.setText("" + Math.round(100.0*((Termbaum) Baum).Ergebnis())/100.0);
 else tfErgebnis.setText("nur TB");
}
```



```
public Termbaum InfixZuTermbaum() {
//funktioniert nicht, wenn nur eine Zahl eingegeben wird (nicht so wichtig, weil kein "richtiger" Term)
//alle (Teil-)Rechnungen müssen eingeklammert werden!
 int i,j,k,l,m,n;
 String t;
 Termbaum b;
 if (Term.charAt(0) == '(') {
  b = new Termbaum(""); Term = Term.substring(1);
  b.setLeftTree(InfixZuTermbaum()); b.setContent(Term.substring(0,1));
  Term = Term.substring(1); b.setRightTree(InfixZuTermbaum());
  Term = Term.substring(1);
 }
 else {
  i = 100;
  j = Term.indexOf('+'); if ((j > 0) && (j < i)) i = j; k = Term.indexOf('-'); if ((k > 0) && (k < i)) i = k;
  l = Term.indexOf('*'); if ((l > 0) && (l < i)) i = l; m = Term.indexOf('/'); if ((m > 0) && (m < i)) i = m;
  n = Term.indexOf(')'); if ((n > 0) && (n < i)) i = n;
  t = Term.substring(0,i); Term = Term.substring(i); b = new Termbaum(t);
 }
 return b;
}
```


```
public void btInfixBaum_onClick() {
 Term = tfInfixBaum.getText(); Baum = InfixZuTermbaum(); paint(myCanvas.getGraphicsContext2D());
}
```




```
public Termbaum PraefixZuTermbaum() {
 //vgl. GitHub bei Interesse - aus Komplexitätsgründen reicht es hier erstmal nur
 //einen der drei Algorithmen „von Term zu Baum“ darzustellen!
 …
}

…

public Termbaum PostfixZuTermbaum() {
//s.o.
…
}
…
```