Bruch

Schreibe eine Klasse `Bruch` zum Verarbeiten von Bruchzahlen. Jedes Objekt dieser Klasse kapselt zwei Instanzvariable vom Typ `long` für Zähler und Nenner. Es sind mindestens folgende Konstruktoren und Methoden zu implementieren:

```
public Bruch()                    // 0/1
public Bruch(long z)              // z/1
public Bruch(long zähler, long nenner)
public Bruch(String bruch)
public Bruch(double wert)
private void vereinfache()        // kürzt und macht Nenner positiv
public static long ggT(long a, long b)
public static long kgV(long a, long b)
public Bruch add(Bruch b)         // liefert this + b
public Bruch sub(Bruch b)         // liefert this - b
public Bruch mult(Bruch b)        // liefert this * b
public Bruch div(Bruch b)         // liefert this / b
public Bruch hoch(int e)          // liefert this ^ e
public void addThis(Bruch b)      // this += b
public void subThis(Bruch b)      // this -= b
public void multThis(Bruch b)     // this *= b
public void divThis(Bruch b)      // this /= b
public void hochThis(int e)       // this ^= e
public String toString()
```

Implementierungsdetails:

- `toString()` liefert je nach Zustand einer privaten boolschen Klassenvariablen eine Bruchdarstellung oder eine Dezimalzahl. Für diese Variable existieren Getter und Setter.
- Ungültige Bruchobjekte werden nicht erzeugt. Sollte eine Operation zu einem ungültigen Bruchobjekt führen, so wird eine `IllegalArgumentException` geworfen.
- Wird den `xxxThis`-Methoden `null` übergeben, so wird das `this`-Objekt nicht verändert. Jene Rechenmethoden, welche einen neuen Bruch retournieren, werfen eine `IllegalArgumentException`.

Teste deine Klasse mit den beiliegenden Tests.