BooleanExpression Constructor

```
public BooleanExpression(String s)
```

Given a boolean expression as a string, parses it into a BooleanExpression object that can manipulate and evaluate it. For example, the following are valid inputs:

- "(a || !b)"
- "!((a && (!b && c)) || d)

Parameters:

`s` - A string representation of the expression to create.

isVariable

```
public boolean isVariable()
```

Returns true if this boolean expression is a variable and false otherwise. For example:

- "a" is a variable
- "!a" is NOT a variable
- "a && b" is NOT a variable

isOr

```
public boolean isOr()
```

Returns true if the top level of this boolean expression is an **or** and false otherwise. For example:

- "a || b" is an or
- "a || (b && c)" is an or
- "a && (b || c)" is NOT an or
- "a" is NOT an or
- "!(a || b)" is NOT an or

isAnd

```
public boolean isAnd()
```

Returns true if the top level of this boolean expression is an **and** and false otherwise. For example:

- "a && b" is an and
- "a && (b || c)" is an and
- "a || (b && c)" is NOT an and
- "a" is NOT an and
- "!(a && b)" is NOT an and

isNot

```
public boolean isNot()
```

Returns true if the top level of this boolean expression is a **not** and false otherwise. For example:

- "!(a && b)" is a not
- "!(a && (b || c))" is a not
- "!a || !(b && c)" is NOT a not
- "a" is NOT a not
- "a && b" is NOT a not

toString

```
public String toString()
```

Returns a string representation of this boolean expression.

Overrides:

`toString` in class `Object`

getVariables

```
public Set<String> getVariables()
```

Returns a set of the variables in this boolean expression. For example:

- If the boolean expression were "a || b", then this method would return ["a", "b"].
- If the boolean expression were "(!a || (!b || (a && c)))", this method would return ["a", "b", "c"].

evaluate

```
public Boolean evaluate(Map<String,Boolean> assns)
```

Returns true if the given assignments force this boolean expression to be true, false if they force it to be false, and null if it could be either true or false. For example:

- If the boolean expression is "(a || b) && c", and the assignments are: {"a"=true}, then "true || b) && c" is not necessarily true AND it's not necessarily false. So, this method would return null.
- If the boolean expression is "(a || b) && c", and the assignments were {"a"=true, "c"=true}, then we would have "(true || b) && true" which is always true; so, this method would return true.

Parameters:

`assns` - A (partial) map from variables to truth values.