1. AST を構成するノードはすべて Node 型とみなせる理由を説明せよ。

構文木は Node の結合ともみなせる（Monkey 言語の場合）

▶ Go 言語のインターフェースの利用

▶ AST に沿って Node を辿ることで評価を行う

すべての要素が Node インターフェースを実装するように設計

1. eval() 関数の戻り値の型に Object インタフェースを利用する理由を説明せよ。

▶ ノードが中置演算子の場合

▶ eval() の再帰呼び出しにより左右の項の評価

▶ 返却型はどのような型となるのか？

▶ 整数値、真偽値、null など様々な種類

▶ 統一的な表現を可能としたい

▶ オブジェクト (インターフェース) の定義

▶ 異なるノードが表現する値を表現するためのデータ構造

▶ オブジェクト指向言語とは無関係

1. 評価器で、"(", ")" に対する処理を記述しなくても、 "(...)" による優先順位の変更が正しく計算されるのは何故か？

評価器で括弧に対する特別な処理を記述しなくても、"(...)" による優先順位の変更が正しく計算されるのは、以下の理由が考えられます：

1. \*\*構文木の構造\*\*:

式を解析して構文木を構築する際に、括弧が適切にノードとして表現されます。つまり、括弧を含む式は、構文木上で適切な位置にノードとして挿入されます。このため、構文木自体が式の優先順位を正しく表現しており、括弧がない場合と同様に適切な優先順位で式が評価されます。

2. \*\*再帰的な評価\*\*:

評価器は構文木を再帰的に評価します。この過程で、括弧を含む式が適切に評価され、その結果が親の式の一部として組み込まれます。括弧内の式が優先的に評価されるため、優先順位の変更が正しく反映されます。

3. \*\*演算子の処理\*\*:

演算子の評価は、構文木の構造に基づいて行われます。演算子の優先順位は、構文木の深さやノードの配置によって決定されるため、括弧が適切に配置されている場合、それに基づいて適切な演算子が優先的に評価されます。

以上の理由から、評価器は括弧に対する特別な処理を記述せずとも、構文木の構造と再帰的な評価に基づいて、括弧を含む式の優先順位を正しく計算することができます。