# Java パッケージの分析

クエリに基づき、`level` パラメータが 0 または指定されていない場合、パッケージ名の全部分を使用し、`level = 0` を“無限”または“全部分を使用”として扱うスクリプトが必要です。以下に、Java ファイルを解析してパッケージ名を抽出する Python スクリプトの実装方法を説明し、要求に合わせた完全なソリューションを提供します。

### 要件の理解

- **`level = 0` または指定されていない場合**: Java の `import` 文から抽出されたパッケージ名の全体を使用します（例: `java.util` または `com.google.common.eventbus`）、切り詰めません。
- **`level > 0` の場合**: パッケージ名を最初の `level` 部分に切り詰めます（例: `com.google.common.eventbus` で `level = 2` なら `com.google`）。
- スクリプトは、Java ファイルを処理し、`import` 文からパッケージ名を抽出し、エッジケースを適切に処理する必要があります。

### ソリューションのアプローチ

1. **引数解析**:
   - 根ディレクトリのみが指定されている場合（例: `python script.py /path/to/dir`）、`level = 0` として設定し、全パッケージ名を使用します。
   - `level` が指定されている場合（例: `python script.py /path/to/dir 2`）、パッケージ名を切り詰めるために使用し、非負の整数であることを確認します。
   - 引数が無効な場合はエラーで終了します。
2. **パッケージの抽出**:
   - `import` 文からパッケージ名を抽出します（例: `import java.util.ArrayList;` は `java.util` を生成）。
   - Java の命名規則を使用: パッケージ名は通常小文字、クラス名は大文字で始まります。
3. **パッケージの切り詰めロジック**:
   - `level = 0` の場合、結果セットに全パッケージ名を追加します。
   - `level > 0` の場合、パッケージ名をドット（.）で分割し、最初の `level` 部分を取得します。
4. **ログと出力**:
   - 全パッケージ名または切り詰められたものを使用しているかを明示的に示します。
   - 見つかった全ユニークパッケージをリストします。

### 実装

以下は、要求に合った完全な Python スクリプトです:

```
import os
import sys
from collections import Counter

def find_java_files(root_dir):
    """
    指定されたディレクトリとそのサブディレクトリからすべての .java ファイルを再帰的に検索します。

    引数:
        root_dir (str): 検索を開始する根ディレクトリ。

    返り値:
        str: 各 .java ファイルのフルパス。
    """
    for dirpath, dirnames, filenames in os.walk(root_dir):
        for filename in filenames:
            if filename.endswith('.java'):
                yield os.path.join(dirpath, filename)

def extract_package(import_statement):
    """
    import 文からパッケージ名を抽出します。

    パッケージ名は小文字、クラス名は大文字で始まるという規則を使用します。ワイルドカードインポート (*) を

    引数:
        import_statement (str): Java ファイルからの import 文の行。

    返り値:
        str: パッケージ名、または決定できない場合は空文字列。
    """
    parts = import_statement.split()
    if parts[0] == 'import':
        parts = parts[1:]
    if parts[0] == 'static':
        parts = parts[1:]
    import_path = ' '.join(parts).strip(';').strip()
    identifiers = import_path.split('.')
    for i, ident in enumerate(identifiers):
        if ident == '*' or (ident and ident[0].isupper()):
```

```
            package_parts = identifiers[:i]
            break
    else:
        package_parts = []
    package = '.'.join(package_parts)
    return package

if __name__ == '__main__':
    # コマンドライン引数の解析
    if len(sys.argv) < 2 or len(sys.argv) > 4:
        print(" 使用法: python script.py <root_directory> [level] [--count]")
        sys.exit(1)

    root_dir = sys.argv[1]
    level = 0
    count = False

    if len(sys.argv) == 3:
        if sys.argv[2] == "--count":
            count = True
        elif sys.argv[2].isdigit():
            level = int(sys.argv[2])
        else:
            print(f" 無効な引数: {sys.argv[2]}")
            sys.exit(1)
    elif len(sys.argv) == 4:
        if sys.argv[3] == "--count" and sys.argv[2].isdigit():
            level = int(sys.argv[2])
            count = True
        else:
            print(f" 無効な引数: {sys.argv[2]} {sys.argv[3]}")
            sys.exit(1)

    # ディレクトリが存在することを確認
    if not os.path.isdir(root_dir):
        print(f"[ERROR] 指定されたパスはディレクトリではありません: {root_dir}")
        sys.exit(1)

    # 分析の開始をログに記録
    level_str = " 全パッケージ名を使用" if level == 0 else f" レベル {level}"
```

```
count_str = " 出現回数を含む" if count else ""
print(f"[INFO] ディレクトリの分析を開始: {root_dir} {level_str} {count_str}")

# 変数の初期化
package_counter = Counter()
total_files = 0
error_files = 0

# Java ファイルの処理
for java_file in find_java_files(root_dir):
    try:
        with open(java_file, 'r', encoding='utf-8') as f:
            file_packages = set()
            for line in f:
                line = line.strip()
                if line.startswith('import'):
                    package = extract_package(line)
                    if package:
                        if level > 0:
                            parts = package.split('.')
                            truncated_package = '.'.join(parts[:level])
                        else:
                            truncated_package = package
                        file_packages.add(truncated_package)
        for pkg in file_packages:
            package_counter[pkg] += 1
        total_files += 1
    except Exception as e:
        print(f"[ERROR] ファイル {java_file} を読み取れません: {e}")
        error_files += 1
        continue

# サマリーの表示
print(f"[INFO] 試行された Java ファイルの合計: {total_files + error_files}")
print(f"[INFO] 成功した処理: {total_files}")
print(f"[INFO] エラーのあるファイル: {error_files}")
if count:
    print(f"[INFO] ユニークパッケージの合計（出現回数を含む）: {len(package_counter)}")
else:
    print(f"[INFO] ユニークパッケージの合計: {len(package_counter)}")
```

```
    # 適切なソートで結果を表示
    if package_counter:
        if count:
            print("[INFO] 分析完了。ユニークパッケージと出現回数を出力（出現回数の降順、パッケージ名の昇順
            # 出現回数の降順、パッケージ名の昇順でソート
            for pkg, cnt in sorted(package_counter.items(), key=lambda x: (-x[1], x[0])):
                print(f"{pkg}: {cnt}")
        else:
            print("[INFO] 分析完了。ユニークパッケージを出力（パッケージ名の昇順でソート）:")
            # パッケージ名の昇順でソート
            for pkg in sorted(package_counter):
                print(pkg)
    else:
        print("[INFO] パッケージが見つかりませんでした。")
```

**使用方法**

- **スクリプトの実行**:
  - `python script.py /path/to/java/project`: ディレクトリ内のすべての `.java` ファイルを分析し、全パッケージ名を使用します（`level = 0`）。
  - `python script.py /path/to/java/project 2`: パッケージ名を最初の 2 部分に切り詰めます（例: `com.google.common.eventbus` は `com.google` になります）。
- **出力例**: 例えば、以下のような Java ファイルがあるとします:

```
import java.util.ArrayList;
import com.google.common.eventbus.EventBus;
```

  - `level = 0`**（またはレベルが指定されていない）の場合**:

```
[INFO] ディレクトリの分析を開始: /path/to/java/project 全パッケージ名を使用
[INFO] ディレクトリに入る: /path/to/java/project
[INFO] ファイルを処理: /path/to/java/project/MyFile.java
[INFO] 試行された Java ファイルの合計: 1
[INFO] 成功した処理: 1
[INFO] エラーのあるファイル: 0
[INFO] ユニークパッケージの合計: 2
[INFO] 分析完了。ユニークパッケージを出力:
com.google.common.eventbus
java.util
```

- `level = 2` **の場合**:

```
[INFO] ディレクトリの分析を開始: /path/to/java/project レベル 2
[INFO] ディレクトリに入る: /path/to/java/project
[INFO] ファイルを処理: /path/to/java/project/MyFile.java
[INFO] 試行された Java ファイルの合計: 1
[INFO] 成功した処理: 1
[INFO] エラーのあるファイル: 0
[INFO] レベル 2 のユニークパッケージの合計: 2
[INFO] 分析完了。レベル 2 のユニークパッケージを出力:
com.google
java.util
```

- **主要な機能**:
  - **全パッケージ名**: `level = 0` または指定されていない場合、抽出された全パッケージ名を使用します（例: `java.util`、`com.google.common.eventbus`）。
  - **切り詰め**: `level > 0` の場合、最初の `level` 部分を取得します。
  - **エラー処理**: 有効なディレクトリと非負の `level` を確認します。
  - **ユニーク性**: パッケージを `set` に格納して重複を避けます。

**対処したエッジケース**

- **レベルの欠如**: デフォルトで `level = 0` に設定し、全パッケージ名を使用します。
- **無効なレベル**: `level` が負または整数でない場合はエラーで終了します。
- **短いパッケージ**: パッケージの部分が `level` より少ない場合（例: `java` で `level = 2`）、全パッケージ（`java`）を使用します。
- **空のインポート**: パッケージが抽出できない行はスキップします。

このスクリプトは、要求を完全に満たします: `level = 0` または指定されていない場合、パッケージ名の全部分を使用し、部分の数を“無限”として扱います。