分析項目中的 Java 套件

根據你的查詢，你需要一個腳本，如果 `level` 參數為 0 或未提供，則使用整個套件名稱（所有部分），實際上將 `level = 0` 視為「無限」或「使用所有部分」，而不是限制為特定數量的部分（例如，當 `level = 1` 時僅使用第一部分）。以下，我將解釋如何在分析 Java 文件以提取套件名稱的 Python 腳本中實現這一點，並提供一個針對你的需求量身定做的完整解決方案。

理解需求

- **當 `level = 0` 或未提供**：使用從 Java `import` 語句中提取的完整套件名稱（例如 `java.util` 或 `com.google.common.eventbus`），而不截斷它。
- **當 `level > 0`**：將套件名稱截斷為前 `level` 部分（例如，對於 `com.google.common.eventbus`，`level = 2` 將給出 `com.google`）。
- 腳本應該繼續處理 Java 文件，從 `import` 語句中提取套件名稱，並適當處理邊界情況。

解決方案方法

1. **參數解析**：
   - 如果僅提供根目錄（例如 `python script.py /path/to/dir`），則設置 `level = 0`，這意味著使用完整的套件名稱。
   - 如果提供了 `level`（例如 `python script.py /path/to/dir 2`），則使用它來截斷套件名稱，確保它是非負整數。
   - 如果參數無效，則退出並顯示錯誤。
2. **套件提取**：
   - 從 `import` 語句中提取套件名稱（例如 `import java.util.ArrayList;` 產生 `java.util`）。
   - 使用 Java 命名約定：套件名稱通常是小寫，類名以大寫字母開頭。
3. **套件截斷邏輯**：
   - 如果 `level = 0`，將完整的套件名稱添加到結果集中。
   - 如果 `level > 0`，按點（`.`）分割套件名稱，並取前 `level` 部分。
4. **記錄和輸出**：
   - 清晰地指出是使用完整的套件名稱還是截斷的套件名稱。
   - 列出所有找到的唯一套件。

實現

以下是滿足你需求的完整 Python 腳本：

```
import os
import sys

def find_java_files(root_dir):
    """
    遞歸查找給定目錄及其子目錄中的所有 .java 文件。

    Args:
        root_dir (str): 開始搜索的根目錄。

    Yields:
        str: 每個 .java 文件的完整路徑。
    """
    for dirpath, dirnames, filenames in os.walk(root_dir):
        print(f"[INFO] 進入目錄: {dirpath}")
        for filename in filenames:
            if filename.endswith('.java'):
                yield os.path.join(dirpath, filename)

def extract_package(import_statement):
    """
    從 import 語句中提取套件名稱。

    假設套件名稱是小寫，並在類名（大寫）或通配符（*）處停止。

    Args:
        import_statement (str): Java 文件中的 import 語句行。

    Returns:
        str: 套件名稱，如果未確定則為空字符串。
    """
    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__':
    # 解析命令行參數
    if len(sys.argv) == 2:
        root_dir = sys.argv[1]
        level = 0  # 未提供 level，使用完整套件名稱
    elif len(sys.argv) == 3:
        root_dir = sys.argv[1]
        try:
            level = int(sys.argv[2])
            if level < 0:
                raise ValueError
        except ValueError:
            print("[ERROR] Level 必須是非負整數。")
            sys.exit(1)
    else:
        print(" 用法: python script.py <root_directory> [level]")
        sys.exit(1)

    # 驗證目錄是否存在
    if not os.path.isdir(root_dir):
        print(f"[ERROR] 指定的路徑不是目錄: {root_dir}")
        sys.exit(1)

    # 記錄分析開始
    if level == 0:
        print(f"[INFO] 開始分析目錄: {root_dir} 使用完整套件名稱。")
    else:
        print(f"[INFO] 開始分析目錄: {root_dir} 級別: {level}")

    # 初始化變量
    packages = set()
    total_files = 0
    error_files = 0
```

```
# 處理 Java 文件
for java_file in find_java_files(root_dir):
    print(f"[INFO] 正在處理文件: {java_file}")
    try:
        with open(java_file, 'r', encoding='utf-8') as f:
            for line in f:
                line = line.strip()
                if line.startswith('import'):
                    package = extract_package(line)
                    if package:
                        if level == 0:
                            # 使用完整套件名稱
                            packages.add(package)
                        else:
                            # 截斷到前 'level' 部分
                            parts = package.split('.')
                            truncated_package = '.'.join(parts[:level])
                            packages.add(truncated_package)
        total_files += 1
    except Exception as e:
        print(f"[ERROR] 無法讀取文件 {java_file}: {e}")
        error_files += 1
        continue

# 打印摘要
print(f"[INFO] 總共嘗試的 Java 文件: {total_files + error_files}")
print(f"[INFO] 成功處理: {total_files}")
print(f"[INFO] 文件出錯: {error_files}")
if level == 0:
    print(f"[INFO] 總共唯一的完整套件: {len(packages)}")
else:
    print(f"[INFO] 級別 {level} 的總共唯一套件: {len(packages)}")

# 打印結果
if packages:
    if level == 0:
        print("[INFO] 分析完成。打印唯一的完整套件:")
    else:
        print(f"[INFO] 分析完成。打印級別 {level} 的唯一套件:")
```

```
        for package in sorted(packages):
            print(package)
    else:
        print("[INFO] 未找到套件。")
```

它是如何工作的

- **運行腳本**：
  - `python script.py /path/to/java/project`：分析目錄中的所有 `.java` 文件並使用完整套件名稱（`level = 0`）。
  - `python script.py /path/to/java/project 2`：將套件名稱截斷為前 2 部分(例如，`com.google.common.eventbus` 變為 `com.google`)。
- **示例輸出**：假設你有一個 Java 文件，內容如下：

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

  - **當 `level = 0`（或未提供 level）**：

```
[INFO] 開始分析目錄: /path/to/java/project 使用完整套件名稱。
[INFO] 進入目錄: /path/to/java/project
[INFO] 正在處理文件: /path/to/java/project/MyFile.java
[INFO] 總共嘗試的 Java 文件: 1
[INFO] 成功處理: 1
[INFO] 文件出錯: 0
[INFO] 總共唯一的完整套件: 2
[INFO] 分析完成。打印唯一的完整套件:
com.google.common.eventbus
java.util
```

  - **當 `level = 2`**：

```
[INFO] 開始分析目錄: /path/to/java/project 級別: 2
[INFO] 進入目錄: /path/to/java/project
[INFO] 正在處理文件: /path/to/java/project/MyFile.java
[INFO] 總共嘗試的 Java 文件: 1
[INFO] 成功處理: 1
[INFO] 文件出錯: 0
[INFO] 級別 2 的總共唯一套件: 2
[INFO] 分析完成。打印級別 2 的唯一套件:
com.google
java.util
```

- **關鍵功能**：
  - **完整套件名稱**：當 `level = 0` 或未傳遞時，使用從中提取的完整套件名稱（例如 `java.util`，`com.google.common.eventbus`）。
  - **截斷**：當 `level > 0` 時，取前 `level` 部分。
  - **錯誤處理**：檢查有效目錄和非負 `level`。
  - **唯一性**：將套件存儲在 `set` 中以避免重複。

處理的邊界情況

- **缺少 level**：默認為 `level = 0`，使用完整套件名稱。
- **無效 level**：如果 `level` 為負或不是整數，則退出並顯示錯誤。
- **短套件**：如果套件的部分數少於 `level`（例如，`java` 當 `level = 2`），則使用完整套件（`java`）。
- **空 import**：跳過無法提取套件的行。

這個腳本完全滿足你的需求：當 `level = 0` 或未提供時，使用套件名稱的所有部分，實際上將其視為「無限」部分。