Analizando Paquetes Java

Basado en su consulta, desea un script en el que, si el parámetro `level` es 0 o no se proporciona, utilice el nombre completo del paquete (todas sus partes), tratando `level = 0` como "infinito"o "usar todas las partes", en lugar de limitarlo a un número específico de partes (como solo la primera parte cuando `level = 1`). A continuación, explicaré cómo se puede implementar esto en un script de Python que analiza archivos Java para extraer nombres de paquetes, y proporcionaré una solución completa adaptada a su solicitud.

Comprensión del Requisito

- **Cuando `level = 0` o no se proporciona**: Use el nombre completo del paquete extraído de las declaraciones `import` de Java (por ejemplo, `java.util` o `com.google.common.eventbus`), sin truncarlo.
- **Cuando `level > 0`**: Truncar el nombre del paquete a las primeras `level` partes (por ejemplo, para `com.google.common.eventbus`, `level = 2` daría `com.google`).
- El script debe seguir procesando archivos Java, extraer nombres de paquetes de las declaraciones `import` y manejar casos límite adecuadamente.

Enfoque de la Solución

1. **Análisis de Argumentos**:
   - Si solo se proporciona el directorio raíz (por ejemplo, `python script.py /ruta/a/dir`), establezca `level = 0`, lo que significa usar el nombre completo del paquete.
   - Si se proporciona `level` (por ejemplo, `python script.py /ruta/a/dir 2`), úselo para truncar el nombre del paquete, asegurándose de que sea un entero no negativo.
   - Salga con un error si los argumentos no son válidos.
2. **Extracción de Paquetes**:
   - Extraer el nombre del paquete de las declaraciones `import` (por ejemplo, `import java.util.ArrayList;` da `java.util`).
   - Utilice las convenciones de nomenclatura de Java: los paquetes suelen estar en minúsculas y los nombres de clases comienzan con mayúsculas.
3. **Lógica de Truncamiento de Paquetes**:
   - Si `level = 0`, agregue el nombre completo del paquete al conjunto de resultados.
   - Si `level > 0`, divida el nombre del paquete por puntos (`.`) y tome las primeras `level` partes.
4. **Registro y Salida**:
   - Indique claramente si se están utilizando nombres completos de paquetes o truncados.
   - Liste todos los paquetes únicos encontrados.

Implementación

Aquí está el script completo de Python que cumple con sus requisitos:

```
import os
import sys

def find_java_files(root_dir):
    """
    Encuentra recursivamente todos los archivos .java en el directorio dado y sus subdirectorios.

    Args:
        root_dir (str): El directorio raíz desde donde comenzar la búsqueda.

    Yields:
        str: La ruta completa a cada archivo .java.
    """
    for dirpath, dirnames, filenames in os.walk(root_dir):
        print(f"[INFO] Entrando en el directorio: {dirpath}")
        for filename in filenames:
            if filename.endswith('.java'):
                yield os.path.join(dirpath, filename)

def extract_package(import_statement):
    """
    Extrae el nombre del paquete de una declaración de importación.

    Asume que los nombres de paquetes están en minúsculas y se detiene en nombres de clases (mayúsculas) o co

    Args:
        import_statement (str): La línea de declaración de importación de un archivo Java.

    Returns:
        str: El nombre del paquete, o cadena vacía si no se determina.
    """
    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__':
    # Analizar argumentos de la línea de comandos
    if len(sys.argv) == 2:
        root_dir = sys.argv[1]
        level = 0  # No se proporciona nivel, usar nombres completos de paquetes
    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] El nivel debe ser un entero no negativo.")
            sys.exit(1)
    else:
        print("Uso: python script.py <directorio_raíz> [nivel]")
        sys.exit(1)

    # Verificar que el directorio existe
    if not os.path.isdir(root_dir):
        print(f"[ERROR] La ruta especificada no es un directorio: {root_dir}")
        sys.exit(1)

    # Registrar el inicio del análisis
    if level == 0:
        print(f"[INFO] Iniciando análisis del directorio: {root_dir} con nombres completos de paquetes.")
    else:
        print(f"[INFO] Iniciando análisis del directorio: {root_dir} a nivel: {level}")

    # Inicializar variables
```

```
packages = set()
total_files = 0
error_files = 0

# Procesar archivos Java
for java_file in find_java_files(root_dir):
    print(f"[INFO] Procesando archivo: {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:
                            # Usar el nombre completo del paquete
                            packages.add(package)
                        else:
                            # Truncar a las primeras 'level' partes
                            parts = package.split('.')
                            truncated_package = '.'.join(parts[:level])
                            packages.add(truncated_package)
        total_files += 1
    except Exception as e:
        print(f"[ERROR] No se pudo leer el archivo {java_file}: {e}")
        error_files += 1
        continue

# Imprimir resumen
print(f"[INFO] Total de archivos Java intentados: {total_files + error_files}")
print(f"[INFO] Procesados correctamente: {total_files}")
print(f"[INFO] Archivos con errores: {error_files}")
if level == 0:
    print(f"[INFO] Total de paquetes completos únicos: {len(packages)}")
else:
    print(f"[INFO] Total de paquetes únicos a nivel {level}: {len(packages)}")

# Imprimir resultados
if packages:
    if level == 0:
```

```
            print("[INFO] Análisis completo. Imprimiendo paquetes completos únicos:")
        else:
            print(f"[INFO] Análisis completo. Imprimiendo paquetes únicos a nivel {level}:")
        for package in sorted(packages):
            print(package)
    else:
        print("[INFO] No se encontraron paquetes.")
```

Cómo Funciona

- **Ejecutar el Script**:
  - `python script.py /ruta/a/proyecto/java`: Analiza todos los archivos `.java` en el directorio y usa nombres completos de paquetes (`level = 0`).
  - `python script.py /ruta/a/proyecto/java 2`: Trunca los nombres de paquetes a las primeras 2 partes (por ejemplo, `com.google.common.eventbus` se convierte en `com.google`).
- **Salida de Ejemplo**: Supongamos que tiene un archivo Java con:

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

  - **Con `level = 0` (o sin nivel proporcionado)**:

```
[INFO] Iniciando análisis del directorio: /ruta/a/proyecto/java con nombres completos de paquetes.
[INFO] Entrando en el directorio: /ruta/a/proyecto/java
[INFO] Procesando archivo: /ruta/a/proyecto/java/MyFile.java
[INFO] Total de archivos Java intentados: 1
[INFO] Procesados correctamente: 1
[INFO] Archivos con errores: 0
[INFO] Total de paquetes completos únicos: 2
[INFO] Análisis completo. Imprimiendo paquetes completos únicos:
com.google.common.eventbus
java.util
```

  - **Con `level = 2`**:

```
[INFO] Iniciando análisis del directorio: /ruta/a/proyecto/java a nivel: 2
[INFO] Entrando en el directorio: /ruta/a/proyecto/java
[INFO] Procesando archivo: /ruta/a/proyecto/java/MyFile.java
[INFO] Total de archivos Java intentados: 1
[INFO] Procesados correctamente: 1
[INFO] Archivos con errores: 0
[INFO] Total de paquetes únicos a nivel 2: 2
```

```
[INFO] Análisis completo. Imprimiendo paquetes únicos a nivel 2:
com.google
java.util
```

- **Características Clave**:
  - **Nombres Completos de Paquetes**: Cuando `level = 0` o no se pasa, usa el nombre completo del paquete extraído (por ejemplo, `java.util`, `com.google.common.eventbus`).
  - **Truncamiento**: Cuando `level > 0`, toma las primeras `level` partes.
  - **Manejo de Errores**: Verifica la validez del directorio y el nivel no negativo.
  - **Uniqueness**: Almacena paquetes en un `set` para evitar duplicados.

Casos Límite Manejados

- **Nivel Ausente**: Predetermina `level = 0`, usando nombres completos de paquetes.
- **Nivel Inválido**: Sale con un error si `level` es negativo o no es un entero.
- **Paquetes Cortos**: Si un paquete tiene menos partes que `level` (por ejemplo, `java` con `level = 2`), usa el paquete completo (`java`).
- **Importaciones Vacías**: Omite líneas donde no se extrae ningún paquete.