Documentación del Proyecto: Simulación del Problema de Parar con el Patrón Abstract Factory en Java

1. Introducción

Este proyecto implementa una simulación conceptual del "Problema de Parar", basado en la prueba de Alan Turing sobre la indecibilidad de ciertos problemas computacionales. Se utiliza el patrón de diseño Abstract Factory para la creación de instancias de programas que pueden detenerse o ejecutarse indefinidamente.

2. Estructura del Proyecto

El código está organizado en paquetes según la funcionalidad de cada componente:

```
src/
 └── main/
    └── java/
       └── abstractFactory/
          ├── factories/     # Fábricas para la creación de programas
          │   ├── HaltingFactory.java  (Fábrica de programas que se detienen)
          │   ├── NHaltingFactory.java (Fábrica de programas que no se detienen)
          │   ├── ProgramFactory.java  (Fábrica abstracta)
          │
          ├── haltchecker/   # Verificador de detención
          │   ├── HaltChecker.java
          │
          ├── model/         # Modelos de programas
          │   ├── HaltingP.java   (Programa que se detiene)
          │   ├── NoHaltingP.java (Programa que no se detiene)
          │   ├── Program.java    (Interfaz base)
          │
          ├── reverse/       # Implementación de inversión de respuestas
          │   ├── Reverse.java
          │
          ├── Main.java      # Punto de entrada
```

3. Descripción de las Clases

3.1. ProgramFactory.java

Define la interfaz abstracta para la creación de diferentes tipos de programas.

3.2. HaltingFactory.java

Fábrica concreta que crea instancias de programas que eventualmente se detienen.

3.3. NHaltingFactory.java

Fábrica concreta que genera programas que nunca se detienen.

3.4. Program.java

Interfaz base que define la estructura de un programa en el sistema.

3.5. HaltingP.java

Implementación de un programa que se ejecuta y se detiene después de cierto tiempo.

3.6. NoHaltingP.java

Implementación de un programa que entra en un bucle infinito y nunca se detiene.

3.7. HaltChecker.java

Intenta verificar si un programa se detendrá o no, pero, debido a la indecibilidad del problema, no puede hacerlo de manera general.

3.8. Reverse.java

Clase que invierte la respuesta del HaltChecker, alterando la predicción del comportamiento del programa evaluado.

3.9. Main.java

Punto de entrada del programa donde se prueban las diferentes implementaciones y se ejecutan los experimentos.

4. Conclusión

Este proyecto ilustra la indecibilidad del Problema de Parar y cómo no existe una solución universal para determinar si un programa se detendrá o no. Además, muestra la aplicación del patrón Abstract Factory, permitiendo la creación de diferentes tipos de programas de manera flexible y desacoplada.