INFORME PATRON ELEGIDO

Mejor Opción: Builder
El Patrón Builder es la opción más adecuada en este caso porque:
- Permite construir programas complejos de forma modular, definiendo diferentes configuraciones sin exponer detalles de implementación.
- Es flexible para agregar nuevas características en el futuro sin modificar demasiado el código existente.
- Evita la sobrecarga de múltiples constructores al proporcionar una manera estructurada de definir los atributos de cada programa.
- Mantiene el código organizado y más legible gracias a la separación del proceso de construcción.

DESCARTADOS

- **Abstract Factory**

El Patrón Abstract Factory es menos adecuado porque:

  - Está diseñado para crear familias de objetos relacionados, pero en este caso solo hay dos tipos de programas (que se detienen y que no), por lo que no se necesita una familia completa de objetos.
  - Introduce demasiada abstracción innecesaria en un problema relativamente simple.
  - Requiere muchas clases adicionales, lo que aumenta la complejidad sin un beneficio claro.

- **Prototype**

El Patrón Prototype no es la mejor opción porque:

  - Está diseñado para clonar objetos complejos cuando su creación es costosa en términos de rendimiento, pero en este caso, los programas son simples y no requieren clonación.
  - No aporta grandes beneficios en un sistema donde los objetos pueden crearse directamente sin necesidad de duplicarlos.
  - Puede introducir problemas de clonación si no se manejan bien las referencias internas de los objetos.

Conclusión

Para el "Problema de Parar", el Patrón Builder es la mejor opción porque permite construir programas de manera flexible sin introducir complejidad innecesaria.

Si el sistema creciera y necesitará múltiples tipos de programas con distintas implementaciones, Abstract Factory podría ser útil en el futuro.

Prototype, en este caso, no es necesario porque la clonación no aporta valor significativo.