APLICANDO LO APRENDIDO 2

Ejercicio 1

Parte A

Considera el lenguaje TypeScript acotado al paradigma de programación estructurada y analízalo en términos de los cuatro componentes de un paradigma mencionados por Kuhn.

1. Generalización simbólica: ¿Cuáles son las reglas escritas del lenguaje?
2. Creencias de los profesionales: ¿Qué características particulares del lenguaje se cree que sean "mejores" que en otros lenguajes?
3. Valores: ¿Qué pensamiento o estilo de programación consideraron mejor los creadores.
4. Ejemplares: ¿Qué clase de problemas pueden resolverse más fácilmente en el lenguaje?

RESPUESTA

1) Reglas escritas del lenguaje TypeScript:

- Tipado estático: A diferencia de JavaScript, TypeScript añade un sistema de tipos estáticos, lo que significa que las variables, funciones y objetos deben tener tipos explícitos o inferidos. Esto ayuda a prevenir errores en tiempo de ejecución al identificar problemas en tiempo de compilación.
- Estructuras de control: Al igual que JavaScript, TypeScript sigue las reglas de programación estructurada con el uso de estructuras de control clásicas como `if`, `for`, `while`, y `switch`.
- Funciones: TypeScript permite definir funciones, tanto tradicionales como funciones de flecha, y pueden ser tipadas con los tipos de entrada y retorno, mejorando la claridad y seguridad del código.
- Modularidad: TypeScript promueve la modularidad mediante el uso de módulos `import` y `export(Si usa moduleES)`, lo que facilita la separación y organización del código en bloques reutilizables.

2) Los profesionales que usan TypeScript suelen tener las siguientes creencias sobre sus ventajas:

- **Seguridad y mantenibilidad**: Se cree que TypeScript mejora significativamente la seguridad y mantenibilidad del código en comparación con JavaScript, gracias a su tipado estático y la detección de errores en tiempo de compilación.
- **Escalabilidad**: TypeScript es visto como más adecuado para proyectos grandes y escalables, ya que su sistema de tipos facilita la gestión de grandes bases de código al hacer que las dependencias y las interacciones entre módulos sean más claras y seguras.

3) Los creadores valoraron la capacidad de escribir código más predecible y estructurado, manteniendo al mismo tiempo la flexibilidad que caracteriza a JavaScript.

4) Para problemas donde se necesita garantizar que los tipos de datos sean correctos y que las interacciones entre componentes sean predecibles (por ejemplo, aplicaciones financieras o sistemas críticos), TypeScript es ideal.