Práctica de Programación III – Programación Orientada a Objetos (POO)

Escenario: Gestión de Proyectos en una Empresa Corporativa

La empresa ficticia *TechCorp Solutions* utiliza un software interno para gestionar sus proyectos. Como desarrollador de software de la empresa, se te ha solicitado mejorar este sistema usando principios de Programación Orientada a Objetos (POO) en C#. Tu objetivo es crear un sistema básico que maneje proyectos, asignaciones de empleados y reporte de estados usando los conceptos de clases, herencia, interfaces, abstracción, polimorfismo, encapsulación, ciclos y manejo de excepciones con try-catch.

Instrucciones:

1. **Clases y Objetos:**
    - Define una clase llamada Proyecto que tenga las siguientes propiedades:
        - NombreProyecto (cadena)
        - FechaInicio (DateTime)
        - FechaFin (DateTime)
        - EstadoProyecto (cadena) – Puede ser "Activo", "Finalizado", "En Proceso".
        - List<Empleado> (Lista de empleados asignados al proyecto)
    - Crea una clase llamada Empleado con las propiedades:
        - Nombre (cadena)
        - Posicion (cadena) – Puede ser "Gerente", "Desarrollador", "Tester", etc.
        - HorasTrabajadas (entero)
    - Implementa métodos dentro de la clase Proyecto que permitan:
        - Agregar empleados al proyecto.
        - Calcular el total de horas trabajadas por todos los empleados del proyecto.
        - Mostrar el estado actual del proyecto.
2. **Herencia y Polimorfismo:**

- Crea una clase derivada de Empleado llamada GerenteProyecto que extienda la clase Empleado. Esta clase debe agregar una propiedad:
  - CantidadProyectos (entero) – Número de proyectos que gestiona el gerente.
- Sobrescribe el método HorasTrabajadas para el GerenteProyecto, de forma que se aplique una bonificación de 10% sobre las horas trabajadas.

3. **Interfaces y Abstracción:**
   - Define una interfaz IReporteable que contenga los métodos:
     - GenerarReporte() – Método para generar un reporte del estado del proyecto.
     - EnviarReporte() – Método para simular el envío del reporte a un gerente.
   - Implementa la interfaz en la clase Proyecto. En el método GenerarReporte(), debes generar un reporte con el nombre del proyecto, estado y horas trabajadas por cada empleado. El método EnviarReporte() debe simular el envío del reporte a un gerente.
4. **Encapsulación:**
   - Asegúrate de que las propiedades de todas las clases sean privadas o protegidas, y accedidas mediante métodos públicos (getters y setters). Por ejemplo, las horas trabajadas por un empleado solo pueden modificarse mediante un método SetHorasTrabajadas() que valide que las horas no sean negativas.
5. **Ciclos:**
   - En el método GenerarReporte(), utiliza un ciclo foreach para iterar a través de la lista de empleados asignados al proyecto y mostrar el nombre de cada empleado y sus horas trabajadas.
6. **Try-Catch (Manejo de Excepciones):**
   - En el método SetHorasTrabajadas() de la clase Empleado, usa un bloque try-catch para manejar el caso en que el valor ingresado sea negativo, lanzando una excepción ArgumentException con el mensaje: "Las horas trabajadas no pueden ser negativas."

Requisitos Adicionales:

- Utiliza la clase Program.cs para probar la funcionalidad de tu sistema.
    - Crea un objeto de la clase Proyecto e inicializa sus propiedades.
    - Crea al menos dos empleados y un gerente, asigna horas trabajadas e intenta agregar un número negativo de horas para probar el manejo de excepciones.
    - Agrega los empleados al proyecto.
    - Genera y envía un reporte del proyecto.

Evaluación:

1. Correcta implementación de clases, objetos y herencia.
2. Uso adecuado de encapsulación mediante métodos getters y setters.
3. Implementación de interfaces y abstracción correctamente aplicada.
4. Uso de ciclos para iterar sobre listas.
5. Manejo de excepciones con try-catch.
6. Pruebas adecuadas en la clase Program para verificar la funcionalidad.