



UNIVERSIDAD ABIERTA INTERAMERICANA
FACULTAD DE TECNOLOGÍA INFORMÁTICA
Carrera: Ingeniería en sistemas informáticos

Asignatura: Trabajo de diploma

Mesa: ☐ Examen Parcial ☐ Recuperatorio ☐ Recuperatorio de Materia ☐ Examen Final

Localización: UAI – ☐ Campus Centro ☐ Sede Castelar ☐ Sede Boulogne

Turno: ☐ Mañana ☐ Tarde ☐ Noche

Fecha:

Alumno:

Legajo:

Obs:

Nota:

Notas: Las respuestas deberán estar escritas con tinta.

Las preguntas que solicitan justificación serán consideradas válidas si poseen la misma correctamente.

Temas a evaluar: El Software. Complejidad del software. La construcción del software como negocio. Costo – Beneficio en la migración de software a nuevas tecnologías. Patrones respecto al nivel de Abstracción Concepto de patrón de diseño. Relación entre los patrones de diseño y el desarrollo de software. Análisis, detección e implementación de los problemas que pueden ser resueltos por los patrones de diseño. Serialization. Reflection. Pruebas.

Requisitos para aprobar: Para que el parcial esté aprobado el alumno deberá tener correctamente desarrollado:

- El 60% del examen teórico.
- El ejercicio práctico completo.

Tiempo: Domiciliario. Entrega y defensa viernes 4 de octubre.

Recomendaciones: a) Lea todo el parcial antes de comenzar a responder. b) desarrolle una redacción clara y precisa contestando lo que la pregunta requiere. c) observe la ortografía ya que la misma forma parte del parcial. d) si considera que no comprende alguna consigna antes de comenzar a contestar consulte a su profesor.

Teórico

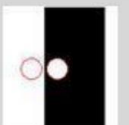
- 1) ¿Cuál es la diferencia entre validar y verificar software?
- 2) ¿Qué tipo y estrategias de pruebas implementará en su trabajo práctico cuatrimestral?
- 3) ¿Cuál es la diferencia entre serialization y persistencia en base de datos relacionales?
- 4) ¿Para qué utilizamos los patrones observer y factory en el trabajo práctico?
- 5) ¿Qué métricas estáticas y dinámicas cree conveniente medir en su proyecto?

Práctico

Se plantea la necesidad de construir un robot Line Follower que permite avanzar sobre una línea negra con el uso de sus actuadores y además, que emita una advertencia sonora en caso de salirse del circuito demarcado por una línea negra.

Para ello el robot cuenta con dos sensores infrarrojos, dos motores y un parlante. Dado que las acciones del robot dependen de su ubicación con respecto al circuito demarcado se deben evaluar los valores leídos por ambos sensores. La siguiente tabla refleja en la primera fila los casos posibles del par de sensores, en la segunda fila una imagen representativa de los mismos y en la tercera fila cómo deben comportarse los actuadores según cada uno de los casos.

Tabla de los 4 casos posibles casos en los que pueden reflejar los sensores:

CASO 1 Sensor izquierdo y derecho sobre la línea	CASO 2 Sensor izquierdo sobre la línea y derecho no	CASO 3 Sensor derecho sobre la línea e izquierdo no	CASO 4 Ningún sensor sobre la línea
			
Avanzar	Girar a izquierda	Girar a derecha	Retroceder y emitir sonido

El par de sensores determinan 4 posibles estados, como se detalló en la figura anterior. Constantemente se deberá evaluar la lectura de ambos sensores y de ser necesario se cambiará el estado del robot. Cada sensor posee un valor del tipo bool: [True: para lectura del color negro] y [False: Para lectura del color blanco]. Para avanzar simplemente se encienden los motores a potencia 100% con giro hacia adelante, en caso de girar hacia la derecha, se envía potencia 50% y giro atrás al motor derecho y potencia 50% y giro hacia adelante del motor izquierdo. Por el contrario, si se desea girar hacia la izquierda, se envía la misma potencia de 50% a los motores, pero girando hacia adelante el motor derecho y hacia atrás el izquierdo. Si el movimiento es retroceder, se le envía giro hacia atrás y potencia 50% a ambos motores.

A fines prácticos la verificación de la posición podrá determinarse cada cierto intervalo de tiempo desde una consola. También puede hacerse una simulación visual, pero es de carácter opcional.

- 1) Desarrolle el diseño de clases de la solución propuesta. (20)
- 2) Implemente el/los patrón/es de diseño solicitado/s en C#. (30)
- 3) Cada vez que el robot detecte un cambio de estado deberá serializar la información en un archivo llamado "mision.bin" teniendo en cuenta los atributos: Fecha/Hora, Valor sensor 1 y Valor sensor 2. Tener en cuenta que la persistencia podría cambiar en cualquier momento a base de datos, archivo de texto plano, etc. (30)
- 4) Permitir buscar todos los cambios de estado del robot en el archivo "mision.bin" según los parámetros FechaDesde, FechaHasta y mostrar los datos por pantalla. (20)

Firma y aclaración

1. La verificación se asegura de que el software se desarrolle correctamente según las especificaciones, mientras que la validación garantiza que el producto final satisface las necesidades y expectativas del usuario.
2. En mi trabajo practico cuatrimestral voy a usar pruebas unitarias, pruebas de integración, pruebas de sistema, pruebas de aceptación.

3. **Serialization (Serialización):**

Definición: Es el proceso de convertir un objeto en una secuencia de bytes u otro formato que pueda ser almacenado o transmitido y posteriormente reconstruido.

Propósito:

- **Transmisión:** Enviar objetos a través de redes (por ejemplo, en APIs REST).
- **Almacenamiento:** Guardar el estado de un objeto en archivos o sistemas de almacenamiento.

Formato Común: JSON, XML, Protobuf.

Uso en Programación: Permite que los objetos en memoria sean convertidos a formatos estándar para su interoperabilidad entre diferentes sistemas o componentes.

Persistencia en Base de Datos Relacionales:

Definición: Es el proceso de almacenar datos de manera permanente en una base de datos relacional, asegurando que los datos sobrevivan a reinicios del sistema y otros eventos.

Propósito:

- **Almacenamiento Permanente:** Guardar datos de manera estructurada para acceso y manipulación futura.
- **Integridad y Consistencia:** Garantizar que los datos cumplen con las reglas de integridad definidas (clave primaria, relaciones, etc.).

Componentes Clave: Tablas, filas, columnas, relaciones, índices.

Uso en Aplicaciones: Permite que las aplicaciones accedan, consulten y actualicen datos de manera eficiente y segura.

4. En el trabajo practico el patron observer lo use para el multilenguaje y en una parte de mi negocio en donde tenia que observar el compartamiento de una clase para que luego en base a eso haga una acción. Por otro lado, la factory la use para la parte de los repositorios en donde va cambiando e instanciando 'x' repositorio que tengo en mi TP.
5. En mi opinión las métricas que podría llegar a utilizar pueden ser:
 - Complejidad Ciclomática: Midiendo la complejidad de mi programa.
 - Duplicación de código: Para verificar en donde se repite el código y poder de alguna forma refactorizarlo
 - Calidad de documentación: Sea consistente , fácil de interpretar para el mantenimiento del proyecto.
 - Rendimiento: Cantidad de transacciones o procesos que el sistema puede manejar en un periodo de tiempo.