**EVALUACIÓN PARCIAL – RESOLUCIÓN DE CASOS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| FACULTAD: | **Tecnología Informática** | | | | |
| CARRERA: | **Ingeniería en sistemas** | | | | |
| ALUMNO/A: | **Francisco José Escobar** | | | | |
| SEDE: | **Buenos Aires** | | LOCALIZACIÓN: | **Centro** | |
| ASIGNATURA: | **Trabajo de diploma** | | | | |
| CURSO: | **3°** | | TURNO: | **Noche** | |
| PROFESOR: | **Nicolas Battaglia** | | FECHA: | **19/11/2021** | |
| TIEMPO DE RESOLUCIÓN: | | **7 días parte práctica** | EXAMEN PARCIAL NRO: | | **2** |
| MODALIDAD DE RESOLUCIÓN: | | | Virtual Domiciliario | | |

**Criterios de calificación:** Para acreditar los saberes deberá obtener, al menos, el 60% de los aspectos conceptuales, además de, al menos, el 60% de los aspectos procedimentales. La calificación final se obtendrá luego de la defensa oral del trabajo presentado.

**Criterios de resolución:** Los alumnos recibirán la consigna del examen en la fecha de evaluación prevista por el cronograma de la asignatura. Deberán resolver y entregar el este examen en el plazo conforme al siguiente cronograma:

|  |  |
| --- | --- |
| **Instancia 1** | **Instancia 2** |
| Entrega de las consignas por parte del docente. | Entrega de la resolución de los aspectos conceptuales y procedimentales por parte del alumno. Defensa oral del trabajo presentado. El horario de entrega será hasta las 12.00 hs. |

**Criterios de evaluación:** Se evaluará la claridad en el planteamiento de los aspectos conceptuales y procedimentales. Desarrollo del diagrama de flujo. Desarrollo del código acorde al diagrama de flujos. La evaluación se hará a partir de la siguiente grilla:

|  |  |  |  |
| --- | --- | --- | --- |
| **Criterio** | **Calificación** | **Observaciones** | |
| Instancia oral |  | |  |
| **Aspectos Conceptuales** | | | |
| Pregunta 1 |  |  | |
| Pregunta 2 |  |  | |
| Pregunta 3 |  |  | |
| Pregunta 4 |  |  | |
| **Aspectos procedimentales** | | | |
| Identificación de patrones |  | **Composite -** | |
| Desarrollo del diagrama de clase punto 1 |  |  | |
| Desarrollo del código acorde al diagrama punto 1 |  |  | |
| Conexión a BBDD |  |  | |
|  | | | |
| Calificación final |  |  | |

**Forma de entrega del examen**

Se deberá entregar un documento, la resolución de todos los aspectos conceptuales y la resolución del punto 1 de los aspectos procedimentales, junto con el código fuente para cumplir los puntos 2 y 3. La defensa oral de la parte práctica consistirá en mostrar el funcionamiento de la solución y responder algunas preguntas que el docente considere necesarias sobre aspectos teóricos.

Los archivos entregados deberán tener el siguiente formado:

* Día turno, Apellido y Nombre, 2do Parcial, TD1 3K 2020.PDF.

E**jemplo**:

* Jueves noche, Gomez Mario, 2do Parcial, TD1 3K 2020.PDF

**Aspectos conceptuales**

1. **¿Cuándo y por qué utilizaría composición de objetos en vez de herencia?**

*Yo utilizaría composición de objetos en vez de herencia en casos en los que el aplicar una herencia me genere la violación de algún patrón de diseño que no quiera que se viole, es decir también, cuando el problema que quiero atacar no entra dentro de un problema que la herencia me vaya a solucionar.   
Por ejemplo, si implemento una interfaz que necesito para algunas clases y para otras no, estaría violando el principio de segregación de interfaces.   
Hay situaciones en las que quiero que una clase mía pueda contar con una funcionalidad extra pero que esta funcionalidad no sea propia de lo que la clase representa, tal como se diseñó y se implementó siguiendo la abstracción de esta, sino que forma parte de una clase externa con la que se relaciona. De esta forma yo derivo a una clase externa la funcionalidad extra que necesito sin tener que implementarla en la misma clase, solucionando este problema.*

1. **Describa, diagrame y ejemplifique el patrón State.**

*El patrón state es un patrón de comportamiento, lo que establece es plantear una solución al problema del manejo de múltiples estados finitos de un objeto cuando este, dependiendo del estado en el que se encuentre, va a cambiar su comportamiento y funcionamiento.   
El problema que viene a solucionar es el problema con el manejo finito de estados, donde originalmente se definiría un bloque condicional preguntando particularmente por cada estado posible e implementando que haga una u otra cosa dependiendo del estado donde se encuentre, siendo esto muy difícil de mantener a medida que se agreguen o modifiquen los estados existentes. Este problema lo viene a solucionar creando nuevas clases para todos los estados posibles de un objeto y delegar los comportamientos posibles a estas clases, reduciendo considerablemente el costo de mantenimiento del sistema.*

*Ejemplo:   
En el siguiente ejemplo, lo que tenemos es una clase “Auto” que hace una composición con una clase abstracta “Estado” de la cual heredan varias clases concretas que representan cada estado existente. De esta manera dependiendo en qué estado se encuentre el Auto, cada clase en particular va a implementar diferente las acciones que se definen en la clase abstracta “Estado”, rompiendo así con el bloque condicional que es poco mantenible.*

**

1. **¿Qué es y para que se utiliza la serialización? Explique las diferencias entre XML, Json Y Binario.**

*La serialización es el proceso de convertir un objeto en una secuencia de bytes para almacenarlo o transmitirlo a la memoria, a una base de datos o a un archivo. Su propósito principal es guardar el estado de un objeto para poder volver a crearlo cuando sea necesario.*

*La diferencia que existe entre XML, Json y Binario es que son formas distintas de almacenar el estado de un objeto en particular. Tanto en XML como en Json voy a estar almacenando el objeto con la definición de sus atributos y el valor de sus atributos, lo que hace que a simple vista sea más fácil y comprensible de leer y hacer referencia a sus atributos y valores en particular, en comparación tal vez con almacenar el objeto como una secuencia de bytes.*

1. **¿Qué es reflexión? De un ejemplo de su utilización.***La reflexión nos brinda objetos que describen ensamblados, módulos y tipos. Se puede usar reflexión para crear dinámicamente una instancia de un tipo, relacionar el tipo con un objeto existente, o leer el tipo de un objeto existente e invocar sus métodos o acceder a sus campos y propiedades.  
   Un ejemplo de su utilización es en mi caso en la gestión de dígitos verificadores dentro de mi sistema como trabajo final, yo lo que hago es generar un Json con un objeto que quiera guardar su estado (por ejemplo, la clase “Usuario”). Esta cadena de texto del Json la almaceno en la base de datos en una tabla “Estados\_Objetos” en un campo de tipo varchar, llamado “Objeto”, junto con otro campo donde describo el nombre de ensamblado de su clase (“NombreClase”) dentro del sistema. En el momento que yo deseo reestablecer este registro, traigo de nuevo el string del Json con el estado del objeto y también su nombre de ensamblado de su clase, y usando reflexión, dependiendo de la clase obtengo su tipo y a partir de su tipo puedo des serializarlo, lo que me hace que pueda compartir una tabla en la base de datos para múltiples clases y no una tabla por clase, gracias a la serialización y a la reflexión, ahorrándome mucho trabajo de mantenimiento.*

**Aspectos procedimentales**

Crear un escenario para resolver un problema utilizando el patrón: Strategy o State y en base a esto proponer un problema y luego resolverlo por medio de un diagrama de clases y codificarlo utilizando winforms.

Además, deberá cumplimentar las siguientes premisas:

* Debe presentarse el diagrama de clases e implementación en C#
* Debe utilizarse una arquitectura de 4 o más capas
* El escenario debe considerar acceso a datos
* No utilizar ejemplos dados en clase ni disponibles en el repositorio de la asignatura
* No puede haber 2 enunciados ni implementaciones iguales

*Respuesta:*

*El escenario que elegí yo es similar al del punto 2, aplicando el patrón “State”.   
Existe la clase “Auto” que puede contener por el momento cuatro estados posibles: “Bloqueado”, “Apagado”, “Encendido”, “EnContacto”, las cuales van a heredar de una clase abstracta “Estado”. El sistema permite transicionar de estados entre los distintos estados del auto existentes de forma altamente mantenible y respetando el patrón “State”. El sistema también consiste en cuatro capas (UI, BE, BLL, DAO) y tiene la habilidad de persistir en la base de datos cada acción que se realiza sobre el auto.  
  
Diagrama de clases:*

**