III Examen Parcial de Diseño de Software (IC6821) I Semestre, 2019

Profesora: Ing.Ericka Solano Fernández Domingo 02 de junio, 2019

Valor de la prueba : 15% Puntos de la prueba: 50

Estudiante : \_\_***Nicole Carvajal y Eduardo Jirón***\_\_ Carne: \_***2017098785 y 2017101878***\_ **Nota\_\_\_\_\_\_**

**Instrucciones Generales**

La prueba se libera el día domingo 02 de junio al final de la tarde y se entrega el martes 04 de junio al final del día. Puede ser desarrollada en forma individual o en PAREJAS.

Debe ser respondida en este documento y ser entregada en formato PDF. El documento debe llamarse **IIIPARCIAL\_SusNombre.PDF** y ser subido en el TEC Digital en el apartado de **Evaluaciones>IIIParcial.**

Se penalizarán con 5 puntos de la nota final obtenida si esta instrucción no se sigue en la forma solicitada.

En los casos que se les solicite un diagrama de clases debe colocar la imagen en dentro de la prueba. No se aceptarán archivos adicionales.

Se revisará únicamente las respuestas que se coloquen en los espacios destinados para las mismas.

Cualquier otro comentario que se añada en lugares no solicitados, serán ignorados.

**Cuadro de evaluación**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **I Parte**  **Selección Unica** | **II Parte**  **Desarollo** | **III Parte**  **Modelaje y Programación** | | | **Puntos** |
| 10 | 10 | 10 | 10 | 10 | 50 |
|  |  |  |  |  |  |

**I Parte Selección Única (10 puntos)**

Del siguiente conjunto de patrones estudiados en clase, seleccione la que se ajusta de manera exacta a cada una de las afirmaciones que se muestran a continuación. Algunas de estas afirmaciones pueden repetirse a un mismo patrón

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cadena de responsabilidad | Memento | Mediator | Command | Strategy | Interpreter |

1. Permite definir un objeto que pueda encapsular la forma en que interactúan un conjunto de objetos asociados, por lo que al provocar bajo acoplamiento, la interacción entre ellos podría variar de manera independiente. \_\_\_\_\_\_\_\_\_Mediator\_\_\_\_\_\_\_\_\_
2. Patrón que permite crear casi un mini lenguaje para implementar la lógica del programa, a través de la interpretación de frases. \_\_\_\_\_\_\_\_Interpreter\_\_\_\_\_\_\_\_\_\_\_
3. Este patrón permite mantener un conjunto de algoritmos de los que el objeto cliente puede elegir aquel que le conviene e intercambiarlo según sus necesidades. El contexto o el cliente pueden elegir el algoritmo que prefiera de entre los disponibles el más apropiado para cada situación. \_\_\_\_\_\_\_\_\_Strategy\_\_\_\_\_\_\_\_\_
4. El patrón guarda parte o todo el estado interno de un objeto, sin romper el principio de encapsulamiento, con el fin de que este objeto pueda ser restaurado más tarde al estado guardado. \_\_\_\_\_\_\_\_\_Memento\_\_\_\_\_\_\_\_\_
5. Este patrón busca evitar el acoplar el emisor de una petición a su receptor, dando a más de un objeto la posibilidad de responder a la petición. \_\_\_\_\_\_\_\_\_Cadena de responsabilidad\_\_\_\_\_\_\_\_\_
6. Se puede considerar un caso particular de patrón Composite, pero dedicado específicamente a labores de parsing. \_\_\_\_\_\_\_\_\_Interpreter\_\_\_\_\_\_\_\_\_
7. Evita tener que acoplar el emisor de una petición con el receptor, dando a más de un objeto la posibilidad de responder a la petición \_\_\_\_\_\_\_\_\_Cadena de responsabilidad\_\_\_\_\_\_\_\_\_
8. La interacción de varios objetos en este patrón puede provocar que la estructura se torne bastante compleja, por lo que se debe establecer la forma en que dichos objetos interactuarán entre sí. \_\_\_\_\_\_\_\_\_Mediator\_\_\_\_\_\_\_\_\_
9. El patrón sistematiza el uso de implementaciones alternativas. \_\_\_\_\_\_\_\_\_Strategy\_\_\_\_\_\_\_\_\_
10. El patrón ofrece la posibilidad de parametrizar objetos para llevar a cabo acciones.

\_\_\_Command\_\_\_\_\_\_\_\_\_\_\_

**II Parte Desarrollo (10 puntos)**

1. **Mediator, Command y Chain of Responsibility** son patrones que desacoplan el proceso de comunicación entre sus partes.

Anote el mecanismo que propone cada patrón para lograr un contraste entre los tres patrones que permita reflejar las diferencias que existen entre ellos en cuanto a la forma en que administran la comunicación entre objetos.

|  |  |
| --- | --- |
| **Patrón** | **Detalle de cómo el patrón administra la comunicación entre objetos.** |
| Mediator | Para utilizar este patrón la comunicación debe estar bien definida y seguir un orden. Su objetivo desacoplar los objetos que se comunican entre ellos a través de un mediador. Existe una necesidad de ordenar la comunicación entre las partes. |
| Command | La comunicación entre objetos es independiente, además este patrón son acciones a realizar a un objeto. Estandariza una orden, también desacopla los objetos de las acciones que realiza. Este patrón no se preocupa por el orden en que se realiza la comunicación. |
| Chain of Responsibility | Para manejar la comunicación entre objetos, Chain of Responsability forma una cadena de manejadores de objetos en específico. Si el objeto de entrada no es recibido por un manejador pasa al siguiente y así sucesivamente hasta encontrar el adecuado o llegar al final de la cadena. |

**III Parte Modelaje y Programación (30 puntos, 10 puntos cada caso)**

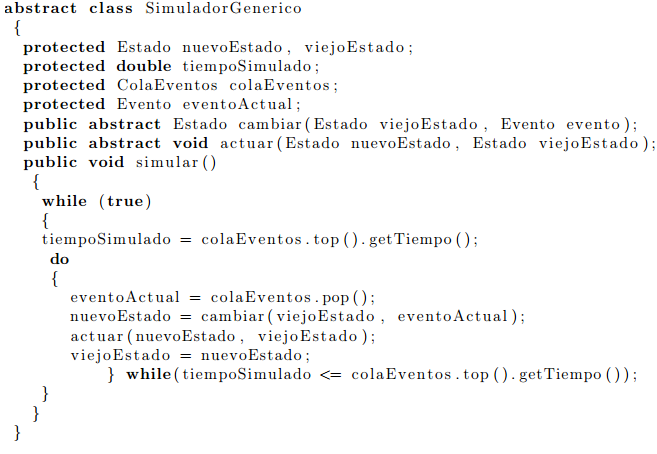
1. Suponga un portal web que permite el registro de usuarios, publicar contenidos en varios formatos (texto, imágenes/video, archivos, entre otros), un usuario puede enviar mensajes privados a otros usuarios, solicitar la suscripción a las publicaciones de otros usuarios, pueden realizar comentarios a de los otros usuarios o sus propias publicaciones.

Proponga la utilización de un patrón adecuado que maneje esta situación, justifique su respuesta, construya el modelo en notación UML 2.0 que soporte esta necesidad y programe el funcionamiento de su propuesta de solución en un proyecto usando el lenguaje de programación Java en un paquete llamado **CasoA\_PortalWEB\_Patron** (donde Patron es el nombre del patrón propuesto)

Interesa la propuesta de diseño del modelo de datos (no el patrón creacional que los genere) pero que se requiere para dar vida al patrón utilizado en su propuesta

|  |  |
| --- | --- |
| Patrón seleccionado (1 puntos)  Mediator | Justificación (2 puntos)  Es un mediator porque el portal web se encarga de manejar la comunicación entre los usuarios; sea publicaciones, mensajes, comentarios o subscripciones |
| Modelo UML 2.0 de implementación propuesta (3 puntos) | |
| Screenshoots del código generado y muestras de funcionamiento adecuado del patrón. (5 puntos)  Mediator    IUsuario    Publicacion    Usuario        Mensaje    PortalWeb      MediatorMain    Resultados | |

1. Considere un simulador orientado a eventos genérico, es decir un simulador esqueleto que simula un rango amplio de sistemas. El simulador contiene una cola de eventos y una variable que indica la hora de simulación (contador). Cada evento tiene grabado el tiempo que indica el momento en que éste debe ocurrir. La cola contiene eventos, los cuales son almacenados en orden ascendente respecto al tiempo en que el evento ocurre. El simulador orientado a eventos genérico ejecuta el siguiente ciclo infinito en el método ***simular*** de la clase SimuladorGenerico:



Considere las siguientes cuestionantes y conteste el siguiente cuadro:

* + - * Si pudiera sugerir la incorporación de un patrón de diseño que satisfaga la necesidad, ¿cuál sería?
      * Justifique su análisis, incluya en esta sección una explicación de cómo sería el cambio en el código dado para incorporar su propuesta?
      * Muestre el diagrama del patrón sugerido usando la notación UML 2.0
      * Programe el funcionamiento de su propuesta de solución en un proyecto usando el lenguaje de programación Java en un paquete llamado **CasoB\_Simulador\_Patron** (donde Patron es el nombre del patrón propuesto). Aporte muestras visuales (screenshoots) del nuevo código según su propuesta y muestras de funcionamiento que solucionan esta inquietud

|  |  |
| --- | --- |
| Patrón seleccionado (1 puntos)  Strategy | Justificación (2 puntos)  El problema busca permitir distintas implementaciones del método simular y para que el usuario pueda escoger el que le sirva.  El cambio principal es hacer el método simular un método abstracto y crear una implementación concreta con el contenido de este método. |
| Modelo UML 2.0 de implementación propuesta (3 puntos) | |
| Screenshoots del código generado y muestras de funcionamiento adecuado del patrón. (5 puntos) | |
| Evidencia del funcionamiento: | |

1. Dentro de sus metas a corto plazo está volverse ingeniero en computación en un plazo **no mayor** a dos años, por lo cual, usted ya tiene planes de comprar un automóvil en el primer año de graduado y debe acondicionar su casa de habitación (o negociar con sus familiares) para hacer la remodelación necesaria y contar con un nuevo garaje en el que pueda guardar su vehículo. Como parte del plan, ha decidido poner en práctica todos su conocimientos e implementar un dispositivo que le permita tener acceso a este sitio, y aprovechando, desarrollar otras características que le ofrezcan una serie de beneficios, pues es un hecho que va a pasar bastante tiempo admirando y cuidando su próxima adquisición.

Por lo que ha incluirá en el nuevo garaje unsistema de ventilación (podría ser unos **ventiladores** o talvez un sistema de **AC**, talvez ambos), juegos de **luces adicionales** a la **estándar** por si debe revisar algo del vehículo con detenimiento por ejemplo en la parte inferior del vehículo, un **estéreo** que pueda hacer sonar la música que esté sonando en el **radio interno** del vehículo o bien habilitar otro **dispositivo para escuchar música**, entre otros y por supuesto, el sistema de manejo de la **puerta** del garaje que le permita ingresar a él o salir sin abandonar su vehículo. Ya ha hecho los contactos con quien se requiera para el diseño físico de este dispositivo que le permitirá vía programación configurar los servicios que se dispondrán y el mecanismo para activar o desactivar cada uno de ellos considerando que en ciertos servicios se debe manipular o graduar las intensidades de funcionamiento de dicho servicio, por ejemplo, **subir** o **bajar** el volumen del estéreo, la intensidad de las luces, la temperatura del AC o la velocidad del ventilador, por ejemplo.

Proponga la utilización de un patrón adecuado que maneje esta situación, justifique su respuesta, construya el modelo en notación UML 2.0 que soporte esta necesidad y programe el funcionamiento de su propuesta de solución en un proyecto usando el lenguaje de programación Java en un paquete llamado **CasoC\_MiGarage\_Patron** (donde Patron es el nombre del patrón propuesto)

|  |  |
| --- | --- |
| Patrón seleccionado (1 puntos)  Command | Justificación (2 puntos)  Es un command porque estandariza la acción de “subir” y “bajar” sin importar si la realiza el sistema de ventilación, las luces, el portón o la música. |
| Modelo UML 2.0 de implementación propuesta (3 puntos) | |
| Screenshoots del código generado y muestras de funcionamiento adecuado del patrón. (5 puntos)                          Muestras de funcionamiento: | |