Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software

Prof.Ing.Ericka Solano Fernández II Semestre, 2018 III Examen Parcial, Grupo **40**

Valor 15% - 75 puntos (25 puntos cada caso) Para ser desarrollado en parejas

Liberado S10 noviembre, 2018. Se entrega M13 noviembre medianoche.

A continuación encontrará unas serie de casos de estudio enumerados del 1 al 3.

Los equipos de trabajo deberán:

Anotar sus nombres y números de carné en la tabla que se presenta abajo.

Equipo de Trabajo
1. Andrés Brenes Maleaño - 2016078594
2. Axel Fernández Jiménez - 2016098894

Para cada patrón escogido el equipo de trabajo deberá:

A. Justifique el patrón que se ajusta al caso escogido

B. Diagrama de UML original del patrón seleccionado

C. Modele el caso de acuerdo al patrón seleccionado

D. Explicación de la responsabilidad que tendrá cada uno de los componentes en el modelo propuesto.

E. Programe un ejemplar del funcionamiento de dicho patrón.

- Los puntos A..D de cada patrón deben ser incluidos en este enunciado en el caso escogido.
- El punto E de los casos expuestos deberán ser programados en un proyecto Java (preferiblemente usando IDE Netbeans) llamado IIIExamenGrupoNNMM (donde N es 40 si es de la sede CASJ o 02 si es de la Sede Cartago, y MM es el número de grupo asignado de trabajo). Construyan en el proyecto tres carpetas, cada una con el nombre CasoX_Patron (donde X es el número de caso a resolver y Patron es el nombre del patrón con el que proponen la solución). Ejemplo Caso7_Flyweight.

Al tec Digital deberá subir un archivo comprimido llamado IIIExamen_GrupoNNMM siguiendo la misma nomenclatura del proyecto programado que contiene el proyecto Java y el enunciado del examen en formato **WORD**.

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software

Prof.Ing.Ericka Solano Fernández II Semestre, 2018 III Examen Parcial, Grupo **40**

Valor 15% - 75 puntos (25 puntos cada caso) Para ser desarrollado en parejas

Liberado S10 noviembre, 2018. Se entrega M13 noviembre medianoche.

Caso 1 (Memento)

Se requiere desarrollar una calculadora que pueda trabajar tres variables de tipo doble y poder realizar operaciones aritméticas sobre dichas variables, las cuales tienen un identificador y un posible valor, por ejemplo x es el nombre de una variable y almacena un valor de tipo doble en un momento del tiempo.

Sobre las variables se pueden desarrollar varias acciones: asignar un valor específico, realizar una operación aritmética con otra variable o valor literal (sumar, restar, multiplicar, dividir, elevar a la potencia, sacar raíz) y el resultado de esa operación quedar almacenado en la misma variable alterando su valor original. Ejemplo a la variable x le puedo sumar 3 y dejar el resultado en x.

La calculadora tiene una opción para guardar el estado de la calculadora en el momento en el que el usuario así lo desee y lo almacena en un historial de "savepoints" que se van incrementando secuencialmente en su identificación conforme el usuario indique que desea guardar el estado.

El historial de las operaciones registradas siempre indica la operación que se realizó, los valores de los operandos antes y después de la operación.

Cuando se solicita la opción de guardar el estado se despliega el indicador de SAVEPOINT i , donde i es es i-ésimo savepoint solicitado.

La calculadora provee además las opciones de deshacer en varias modalidades:

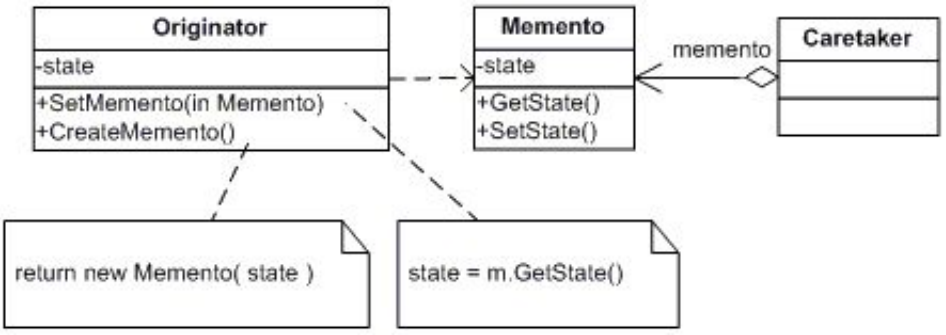
- Un simple deshacer restablecerá el estado de la calculadora al estado antes de la última operación realizada.
- Un deshacer a un punto específico del historial de operaciones que restaura la calculadora y los valores de las variables a ese punto. Este punto particular se referencia por parte del usuario indicando el número de save point al que desean retornar.
- Un deshacer todo que borrará todo el estado de la calculadora y restaura las variables al punto de inicio.

Respuestas

- Justifique el patrón que se ajusta al caso escogido
    - El patrón elegido para solucionar este caso es el Memento. El patrón de diseño Memento es útil para guardar el estado actual de un objeto

en cierto momento del tiempo. En el caso, se desea preservar el estado de la calculadora cada vez que el usuario desee, además de que se debe guardar el anterior cada vez que se modifica, para la opción de undo.

- Diagrama de UML original del patrón seleccionado



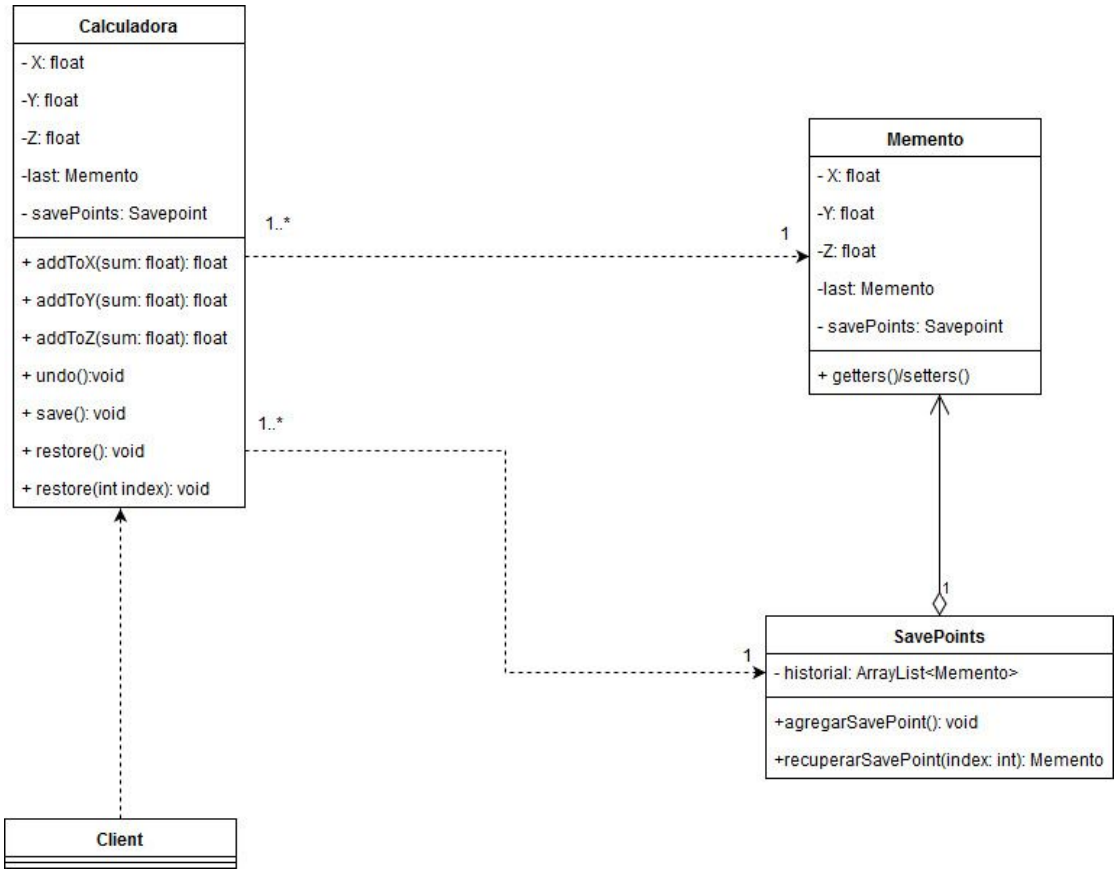
Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software

Prof.Ing.Ericka Solano Fernández II Semestre, 2018 III Examen Parcial, Grupo **40**

Valor 15% - 75 puntos (25 puntos cada caso) Para ser desarrollado en parejas

Liberado S10 noviembre, 2018. Se entrega M13 noviembre medianoche.

- Modele el caso de acuerdo al patrón seleccionado



- Explicación de la responsabilidad que tendrá cada uno de los componentes en el modelo propuesto.
    - Calculadora: Es el ***originator***, ya que es el objeto que recibe los cambios en su estructura y cuya información se desea guardar.
    - Memento: Como su nombre lo indica, es el ***Memento***. Se encarga de guardar el estado de la calculadora en diferentes momento.
    - SavePoints: Es el ***CareTaker***. Se encarga de almacenar un conjunto de Mementos que serán las diferentes versiones.

Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software

Prof.Ing.Ericka Solano Fernández II Semestre, 2018 III Examen Parcial, Grupo **40**

Valor 15% - 75 puntos (25 puntos cada caso) Para ser desarrollado en parejas

Liberado S10 noviembre, 2018. Se entrega M13 noviembre medianoche.

CASO 2 (Command)

Un sistema requiere llevar a cabo distintos tipos de trabajos, por ejemplo, enviar correos electrónicos, enviar SMS, abrir la galería de fotografías ubicadas en determinada carpeta, sonar una canción almacenada en su repertorio musical, entre otros.

A veces se desea que estos trabajos se puedan realizar en forma simultánea, es decir que se pueda escuchar una canción al mismo tiempo que se envía un correo electrónico, por ejemplo. Como estos trabajos son independientes entre sí, la secuencia de ejecución de estos trabajos no es realmente importante. Se considera crear un grupo de subprocesos para limitar el número de subprocesos para ejecutar trabajos y no saturar el sistema, suponga 10. El sistema desconoce el detalle de cómo se llevan a cabo los trabajos, sólo sabe que tiene una cierta cantidad de tareas que procesar en la cola de trabajos asignada en un momento del tiempo y brinda el servicio de atención.

Respuestas

- Justifique el patrón que se ajusta al caso escogido.
    - Se eligió el patrón ***command***, debido a que existe la posibilidad de ejecutar varias tareas en cola, y el ***invoker*** de este patrón posee esta capacidad, debido a que podemos disponer los diferentes procesos que se nos solicitan de manera secuencial, tales como escuchar música y enviar un correo, o ver las fotos y escuchar música.

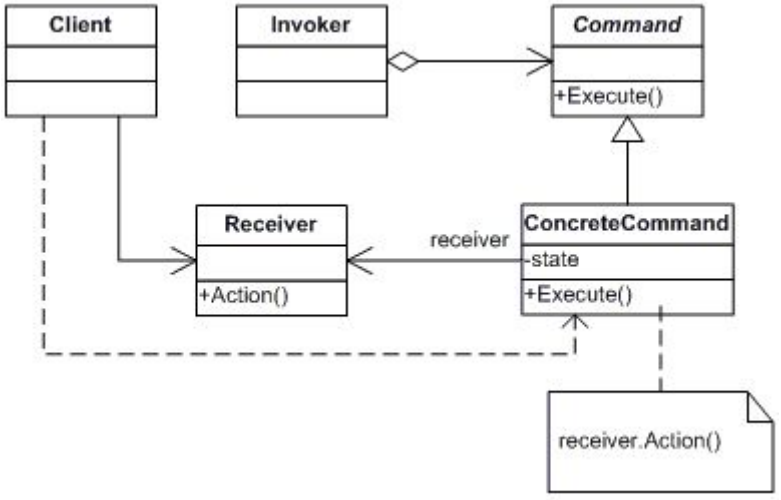
Tecnológico de Costa Rica Ingeniería en Computación IC 6821 Diseño de Software

Prof.Ing.Ericka Solano Fernández II Semestre, 2018 III Examen Parcial, Grupo **40**

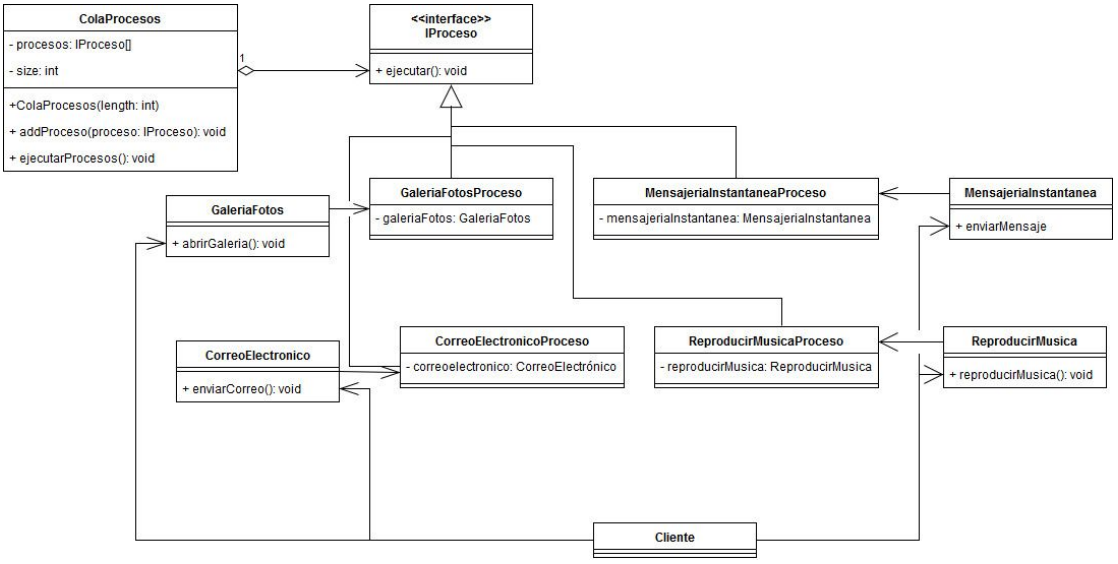
Valor 15% - 75 puntos (25 puntos cada caso) Para ser desarrollado en parejas

Liberado S10 noviembre, 2018. Se entrega M13 noviembre medianoche.

- Diagrama de UML original del patrón seleccionado.



- Modele el caso de acuerdo al patrón seleccionado



- Explicación de la responsabilidad que tendrá cada uno de los componentes en el modelo propuesto.
    - ColaProcesos: Es el ***invoker***, encargado de tener los procesos en cola para luego ir uno por uno realizando lo que se solicita por el cliente.
    - IProceso: Es la ***interface*** encargada de delegar los procesos de ejecución a las distintas subclases.

- GaleriaFotosProceso, MensajeríaInstantaneaProceso, ReproducirMúsicaProceso, CorreoElectronicoProceso: Estos serían los ***concreteCommands***, cada uno es necesario para el cliente puesto que él requiere que se puedan enviar correos, mensajes, ver fotos y escuchar música de manera simultánea como dice la especificación.
- GaleriaFotos, MensajeríaInstantanea, ReproducirMúsica, CorreoElectronico: Son los ***commands*** que el cliente puede solicitar, y cada uno tiene un método encargado de ejecutar lo que se solicita.

CASO 3 (STATE)

Una empresa está buscando construir un robot para cocinar. La compañía quiere un robot simple que simplemente pueda caminar y cocinar.

Un usuario puede operar un robot por medio del accionar 4 botones en un control remoto: ON, OFF, WALK, COOK. La compañía ha establecido protocolos para definir la funcionalidad del robot.

Si el robot está apagado, podría encenderse y esperar una nueva orden. Si el usuario presiona el botón WALK, automáticamente se enciende y camina, pero no pasa nada si presiona el botón COOK.

Estando encendido y esperando instrucciones, podría hacer cualquiera de las tres opciones disponibles, apagarse, caminar o cocinar.

Si el robot está cocinando puede seguir cocinando o detenerse y caminar pero no puede apagarse.

Si se encuentra caminando, sólo puede apagarse.

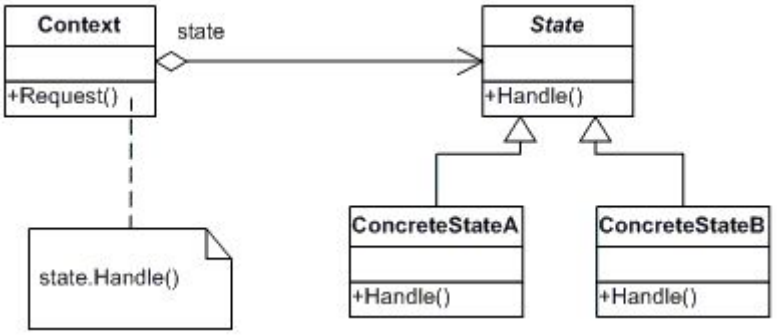
Se desea que pueda elaborar un prototipo del robot y construir una simulación del funcionamiento del mismo a través un programa de prueba.

Respuestas

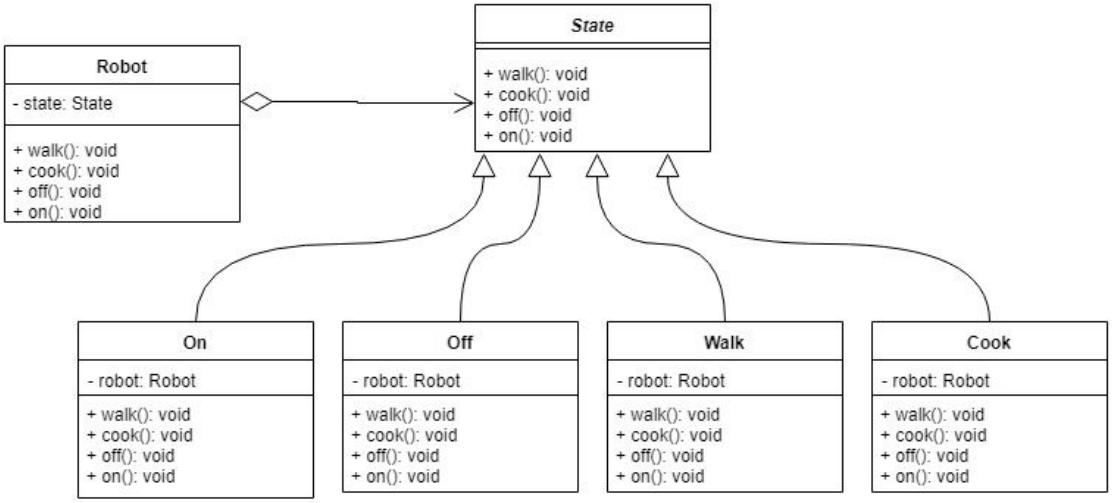
- Justifique el patrón que se ajusta al caso escogido.
  - Se eligió el patrón **state**, esto debido a que el robot posee la capacidad de estar en diferentes estados en diferentes períodos de tiempo, se dice que puede cocinar, caminar, estar prendido o apagado, siendo manejado con un control de 4 botones. Existen ciertas restricciones como que no se puede apagar si está cocinando,

por lo que en la implementación de la clase “Cook”, cuando se desea apagar el robot se imprime en pantalla que no es posible. También existe la restricción que cuando se está apagado y se quiere presionar el botón “Cook” el robot no debe reaccionar, y en la implementación esto se representa.

- Diagrama de UML original del patrón seleccionado



- Modele el caso de acuerdo al patrón seleccionado



- Explicación de la responsabilidad que tendrá cada uno de los componentes en el modelo propuesto.
    - Robot: El robot en este caso sería el ***context***. El robot posee un estado único, y este varía dependiendo de lo que se le solicite a la interfaz *state.*

- State: Es una interfaz encargada de delegar las diferentes actividades que puede realizar el robot, en este caso son 4 procesos diferentes que se determinan en esta *interface*.
- Off: Es un ***estado concreto***. Se encarga de decir qué puede hacer el robot empezando en el estado ***off***, por ejemplo, se sabe que no puede cocinar, por lo que si el cliente selecciona cocinar y está apagado, este proceso va a retornar que es imposible para el robot hacer esto.
- On: Es un ***estado concreto***. Cuando se está encendido, el robot espera órdenes, y puede realizar cualquier operación que se le indique, por lo que esta clase implementa de la misma manera que ***off***, solo que sin las restricciones.
- Walk: Es un ***estado concreto***. Si el robot está apagado y se presiona ***walk***, el robot se enciende y empieza a caminar, en esta clase se desarrolla este proceso. E implementar el proceso de cocinar.
- Cook: Es un ***estado concreto***. Cuando el robot se encuentra en estado de cook, no se puede pasar al estado ***off***, por lo que la restricción está presente. E implementa la manera de caminar también.