Programación 4

PARCIAL FINAL EDICIÓN 2022 - SOLUCIÓN

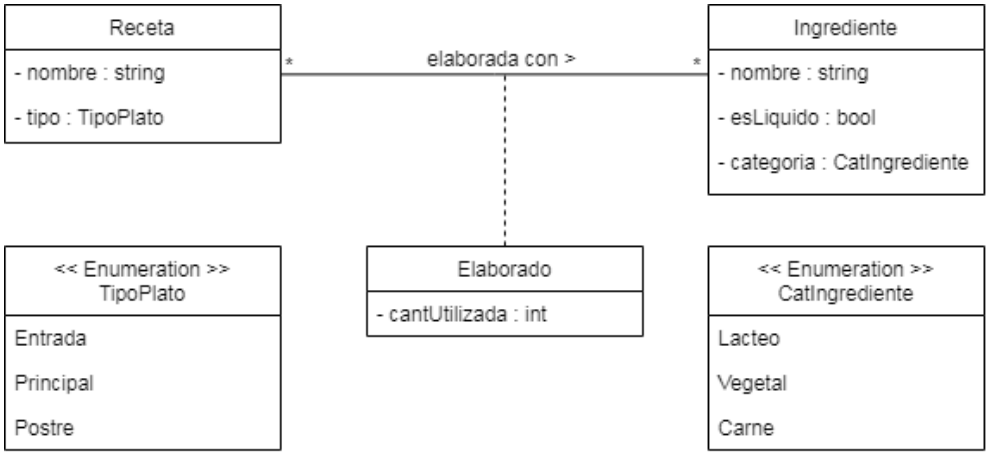
Por favor siga las siguientes indicaciones:

- Escriba con lápiz y de un solo lado de las hojas
- Escriba su nombre y número de documento en todas las hojas que entregue
- Numere las hojas e indique el total de hojas en la primera de ellas
- Recuerde entregar su número de parcial junto al parcial
- Está prohibido el uso de computadoras, tabletas o teléfonos durante el parcial

Problema 1 (25 puntos)

Parte A

a) Realizar el Modelo de Dominio de la realidad planteada **sin incluir** restricciones no estructurales.



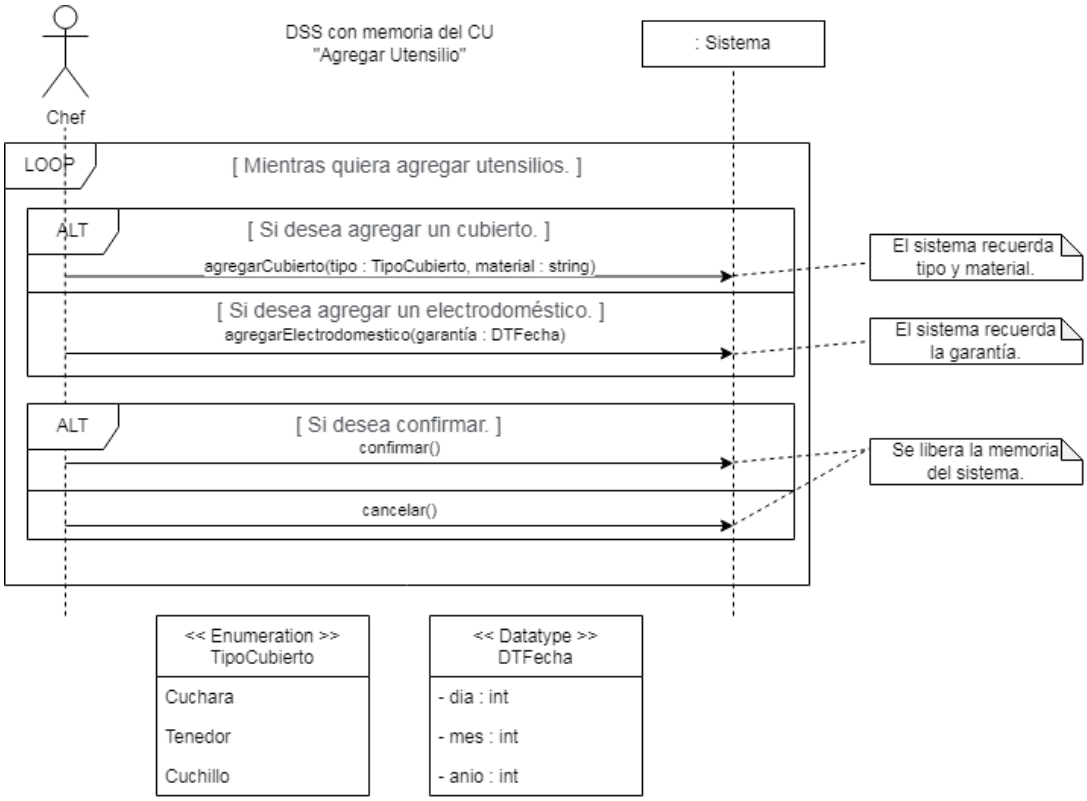
b) Escribir las pre y post condiciones de los contratos de **todas** las operaciones del sistema identificadas para el caso de uso “Consultar Receta”.

`listarRecetas():Set(string)`
PRE: -
POST: Se retorna un conjunto de string que se corresponden con los valores de los atributos "nombre" de las instancias de Receta existentes en el sistema.

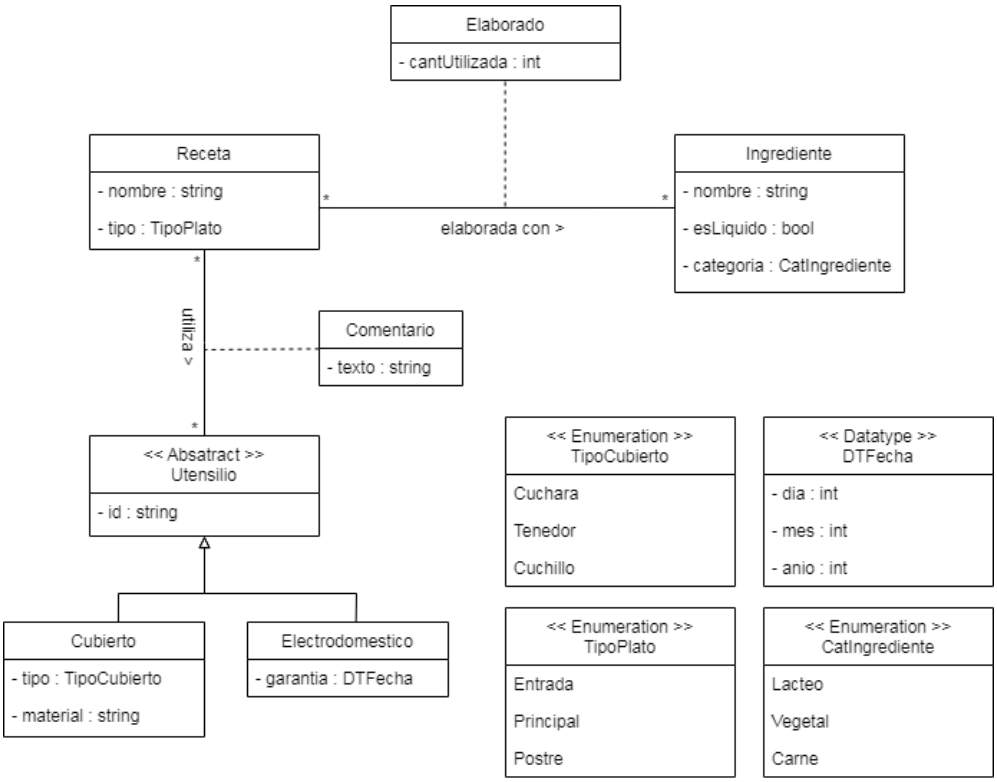
`consultarReceta(nombreReceta:string):DTRecetaExtendido`
PRE: Existe en el sistema una instancia de Receta “r” tal que r.nombre sea igual a nombreReceta
POST1: Se retorna un datavalue RecetaExtendido "re" tal que re.nombreReceta = r.nombre. Además, para cada instancia de Ingrediente "i" asociado a "r" mediante el tipo asociativo Elaborado "e", se genera un datavalue ElaboradoIngrediente "ei" tal que ei.nombreIngrediente = i.nombre y ei.cantUtiliza = e.cantUtiliza y se lo asocia a "re".

Parte B

a) Realizar el DSS para el caso de uso “Agregar Utensilios”, indicando el uso de memoria del Sistema así como los datatypes utilizados.



b) Realizar un nuevo Modelo de Dominio que contemple toda la realidad planteada en ambas fases. Incluir las restricciones en lenguaje natural que correspondan.



Restricciones:

- El atributo nombre identifica a una instancia de Receta.
- El atributo nombre identifica a una instancia de Ingrediente.
- El atributo id identifica a una instancia de Utensilio.

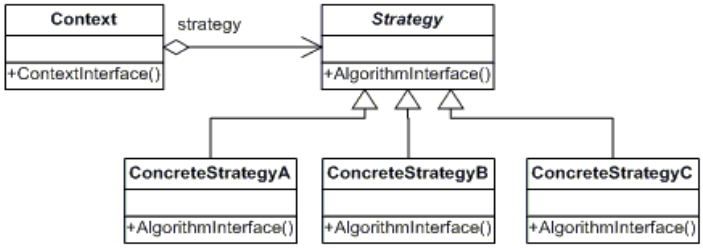
Problema 2 (30 puntos)

Parte A

Explique el problema tipo que resuelve el patrón Strategy e ilustre gráficamente su estructura general y participantes.

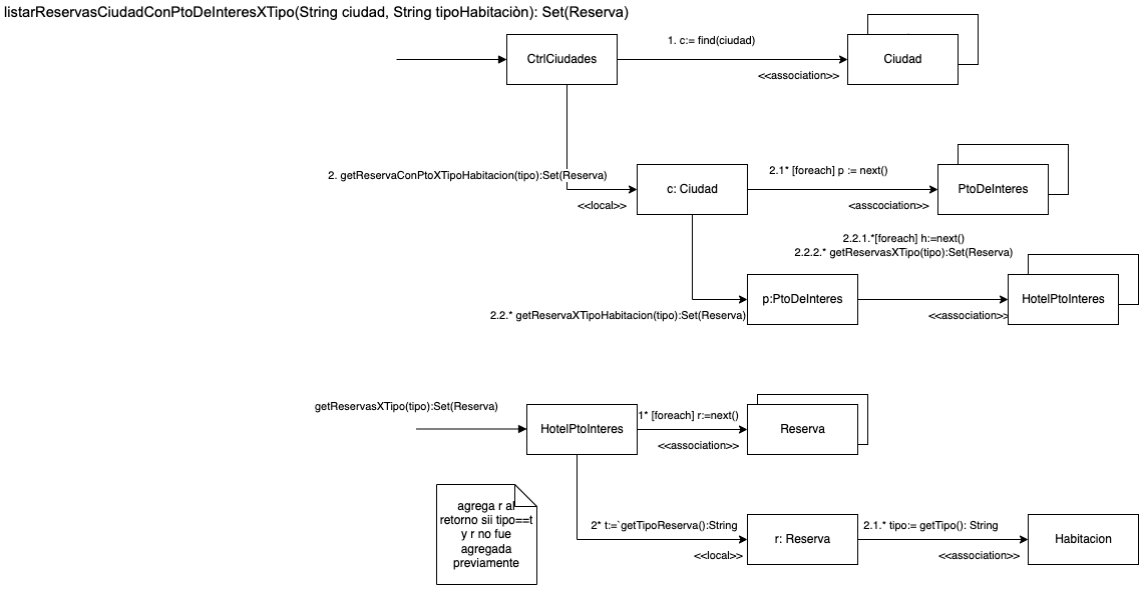
El patrón Strategy permite:

1. encapsular una familia algoritmos/estrategias
2. y su intercambiabilidad en tiempo de ejecución.

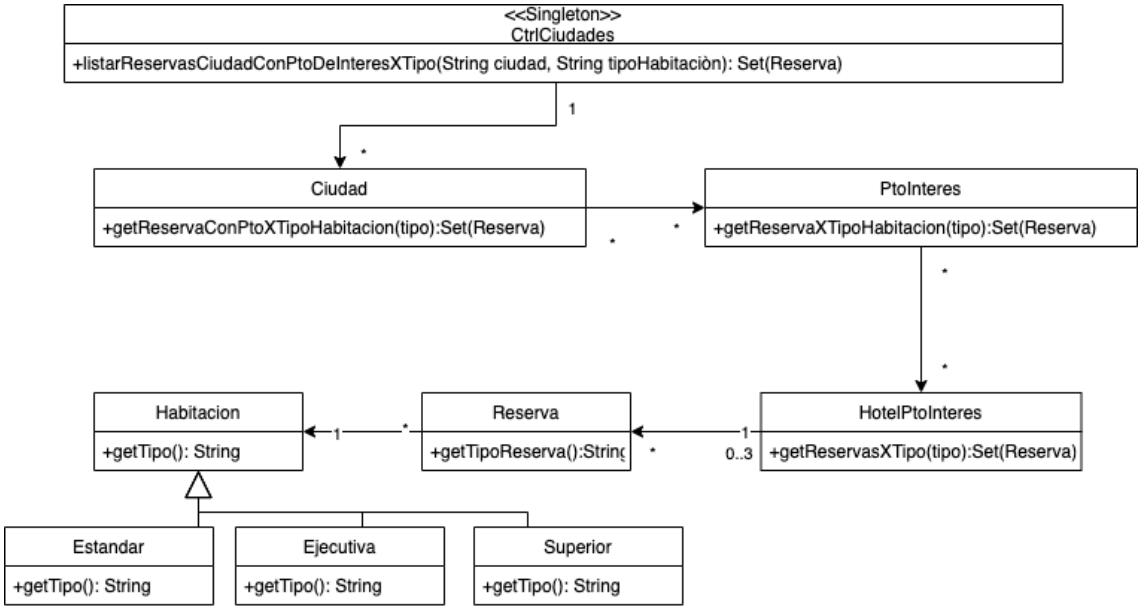


Parte B

a) Realizar el Diagrama de Comunicación correspondiente a la operación especificada en el controlador CtrlCiudades, incluyendo las visibilidades. Asuma que CtrlCiudades posee una colección global de ciudades por la cual se debe acceder a la información. Ninguna otra colección global ni controladores adicionales pueden ser definidos.



b) Realizar el Diagrama de Clases de Diseño (DCD) resultante del diseño realizado en a).



Problema 3 (30 puntos)

a. Implementar en C++ los .h del enumerado `EventType` y de las clases `ActionListener`, `Component` y `MyFrame`.

```
enum EventType { ERROR, MOUSE_CLICKED, KEY_PRESSED };


class ActionListener
{
public:
    virtual void actionPerformed(EventType) = 0;
};

class MyFrame : public Frame, public ActionListener
{
private:
    int clicks;
public:
    MyFrame(string);
    ~MyFrame();
    void actionPerformed(EventType);
    void show();
};
```

```
class Component
{
private:
    string name;
    list<ActionListener *> listeners;
public:
    Component(string);
    virtual ~Component();
    void action(EventType);
    void addActionListener(ActionListener *);
    virtual void show() = 0;
};
```

b. Implementar en C++ el .cpp de la clase `Component`, incluyendo constructor y destructor.

```
Component::Component(string n)
{
    name = n;
}

Component::~Component()
{
}

void Component::action(EventType at){
    list<ActionListener *>::iterator it;
    for (it = listeners.begin(); it != listeners.end(); ++it)
        (*it)->actionPerformed(at);
}

void Component::addActionListener(ActionListener * al){
    listeners.push_back(al);
}
```

c. Implementar en C++ los métodos de las operaciones `Frame::closeFrame` y `MyFrame::actionPerformed`.

```
void Frame::closeFrame(){
    while (!components.empty()){
        Component *c = components.front();
        components.pop_front();
        delete c;
    }
    delete this;
}
```

```
void MyFrame::actionPerformed(EventType et){
    switch (et) {
        case EventType::ERROR:
            closeFrame();
            throw std::runtime_error("ERROR");
            break;
        case EventType::MOUSE_CLICKED:
            clicks++;
            break;
       default:
            break;
    }
}
```

d. Implementar un main que permita hacer lo siguiente, con manejo de excepciones imprimiendo en pantalla el mensaje de error:
   - Crear un `MyFrame` *mf* que incluya un `Button` *bu*
   - Hacer que *mf* observe los eventos de *bu*
   - Generar un evento `MOUSE_CLICKED` sobre el botón *bu*
   - Generar un evento `ERROR` sobre el botón *bu*

```
int main(){
    // Crear un Frame mf que incluya un Button bu
    MyFrame *mf = new MyFrame("My Frame");
    Button *bu = new Button("My Button");
    mf->addComponent(bu);

    //Hacer que mf observe los eventos de bu
    bu->addActionListener(mf);

    //Generar un evento MOUSE_CLICKED sobre el botón bu
    //La ventana que lo contiene debería incrementar en 1 los clicks
    EventType et = MOUSE_CLICKED;
    bu->action(et);

    try{
        //Generar un evento ERROR sobre el botón bu
        //La ventana que lo contiene debería cerrarse y lanzar una excepción
        et = ERROR;
        bu->action(ERROR);
    }
    catch (std::runtime_error& e){
         cout << "EXCEPTION: " << e.what() << '\n';
    }
}
```