

ÉVALUATION FORMATIVE THÉORIQUE

GEN241 : Modélisation et programmation orientées objet

SOLUTIONNAIRE

SESSION S2 – UNITÉ 1

QUESTION 1

Les sous-questions suivantes se rapportent à la modélisation objet. Donner des réponses suffisamment détaillées dans la limite de l'espace disponible.

Sous-question 1.1

Donnez trois caractéristiques d'un objet :

état (données membres, attributs), comportement (méthodes), identité (nom)

Sous-question 1.2

Décrivez la relation entre un objet et une classe.

Chaque objet est décrit par une classe. Une classe *modélise* et *classifie* une catégorie d'objets, c'est une description abstraite qui regroupe des objets qui ont des caractéristiques similaires. Les objets sont des instances d'une classe.

Sous-question 1.3

Dessiner le symbole de composition.



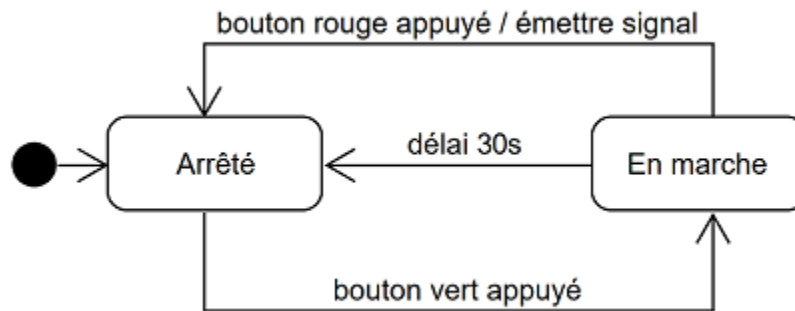
Sous-question 1.4

Qu'est-ce qu'un diagramme de classes permet de représenter ?

les types des objets (les classes) qui composent un système et les différents types de relations statiques qui existent entre eux. Il représente aussi les propriétés et les opérations des classes et les contraintes sur la façon dont les objets sont connectés.

Sous-question 1.5

Formuler un diagramme d'états-transitions pour décrire la situation suivante. Un moteur électrique est initialement à l'arrêt. Il se met en marche lorsque le bouton vert est appuyé et s'arrête automatiquement après un délai de 30 secondes, ou avant ce délai, lorsque le bouton rouge est actionné. Quand le bouton rouge est actionné, un bref signal sonore est émis.

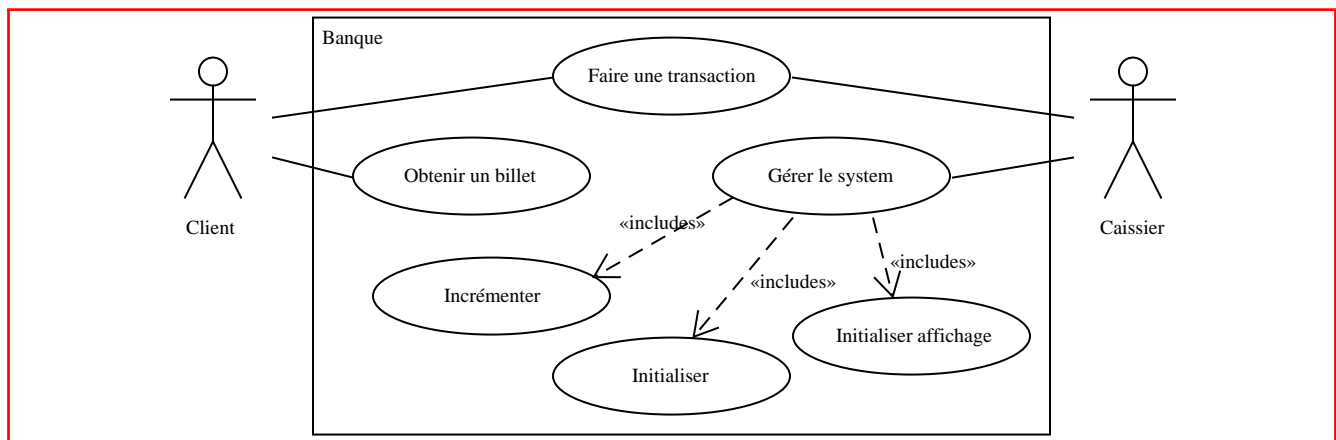


QUESTION 2

Pour obtenir un service au comptoir d'une banque, les clients utilisent un système de billets numérotés. Les clients arrivent dans la succursale, obtiennent un billet numéroté en séquence d'une machine distributrice et ils attendent l'appel de leur numéro pour se rendre au guichet effectuer une transaction avec le caissier. Un dispositif d'affichage ayant un compteur maintient à jour le numéro à appeler. Ce dispositif d'affichage est incrémenté par le caissier au guichet. Dans le scénario considéré on se limite à un seul guichet ouvert. Après avoir pris un billet, les clients attendent l'affichage du numéro de leur billet pour se présenter au guichet et faire une transaction bancaire. Le caissier initialise la machine et l'affichage.

Sous-question 2.1

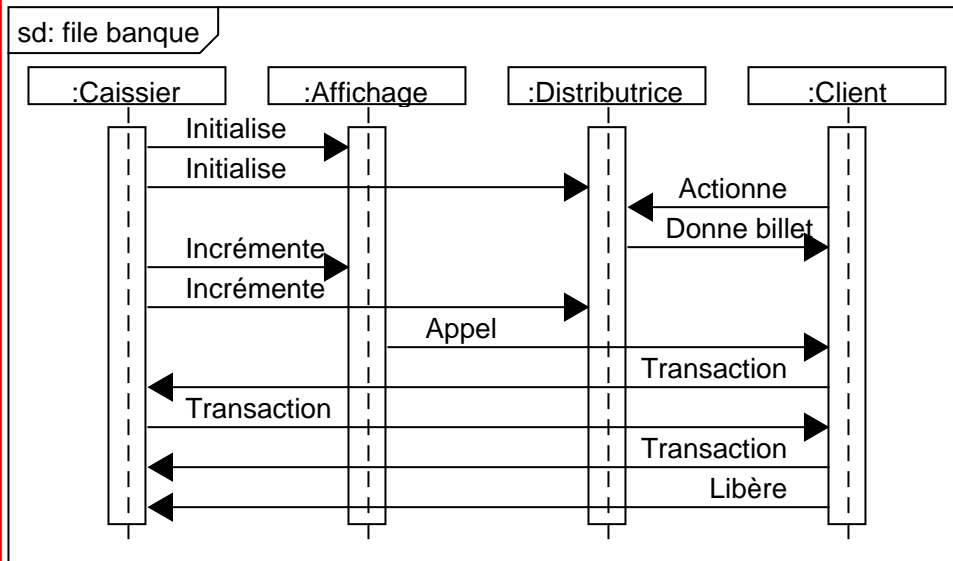
Formuler un diagramme de cas d'utilisation cohérent sur la base de l'information disponible.



Sous-question 2.2

Formuler un diagramme de séquences cohérent sur la base de l'information disponible.

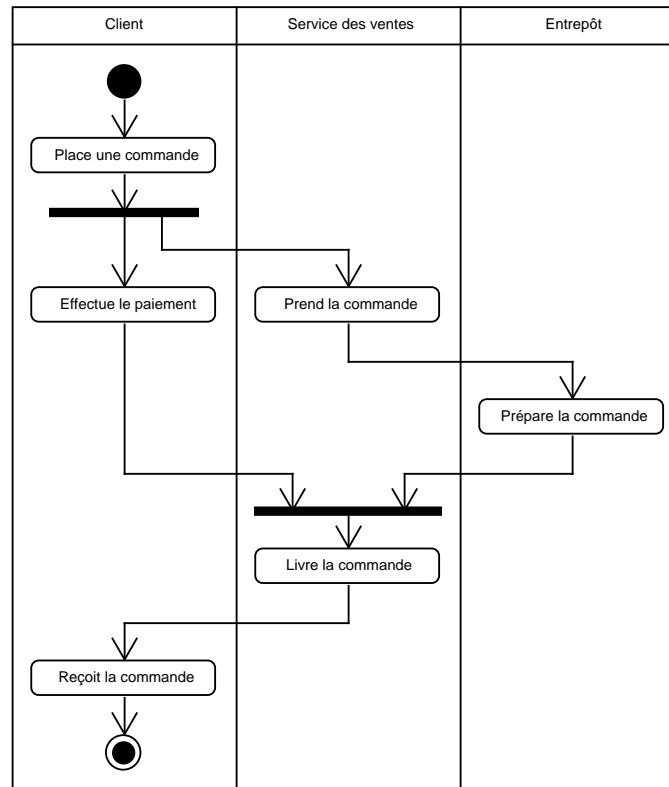
Voici un exemple de solution valide :



*À noter qu'il y a plus d'une réponse valide pour ce diagramme (tel qu'en présentant des boîtes d'activation de durée différente ou d'autres types de séquences de message)

QUESTION 3

On considère le diagramme d'activités suivant :



Sous-question 3.1

Donner une description des éléments graphiques présents dans le diagramme et de leur signification.

Le point noir indique le début. Le point noir encerclé indique la fin. Les rectangles arrondis décrivent des activités. Les travées (*swimlane*) permettent d'associer un acteur à une activité. Les barres horizontales (débranchement) indiquent un début d'activités simultanées ou la jonction de deux fils d'activités. Les flèches indiquent les séquences.

Sous-question 3.2

Donner une description narrative de ce que décrit ce diagramme.

Le client place une commande. Simultanément, le service des ventes prend la commande et le client effectue le paiement. L'entrepôt prépare et la commande fournie par le service des ventes. Le service des ventes fait la livraison quand la commande préparée par l'entrepôt est prête et que le paiement par le client est fait. Le client reçoit finalement la livraison.

QUESTION 4

Proposer un code C++ pour la méthode `inverser` pour la définition de classe suivante qui définit un vecteur de nombres réels. Si un vecteur contenait les valeurs 1, 2, 3, 4 et 5, alors après l'appel à cette méthode, le vecteur contiendrait les valeurs 5, 4, 3, 2 et 1.

```
class VecteurReel
{
private:
    int taille;                // Nombre d'éléments dans le vecteur
    int capacite;              // Nombre max d'éléments dans le vecteur
    double *donnees;          // Tableau pour les données
public:
    VecteurReel(int cap);      // Constructeur avec capacité initiale
    ~VecteurReel();            // Destructeur
    bool ajouter(double valeur); // Ajoute un élément à la fin
    double obtenir(int index) const; // Donne la valeur d'un élément
    void inverser();            // Inverse les valeurs dans le vecteur
    void afficher(ostream & s) const; // Affiche les valeurs dans le vecteur
};

void VecteurReel::inverser()
{
    int milieu = taille/2;
    int droite = taille-1;
    int gauche = 0;
    double temp;

    while(gauche < milieu)
    {
        temp = donnees[gauche];
        donnees[gauche++] = donnees[droite];
        donnees[droite--] = temp;
    }
}
```

QUESTION 5

Proposer un code C++ pour la méthode **ajuster** pour la définition de classe suivante qui définit un vecteur de pointeurs de forme géométriques. Cette méthode permet d'ajuster le vecteur afin que sa capacité soit réduite et être égale à sa taille. Le tableau alloué dynamiquement doit aussi être ajusté en conséquence.

```
class VecteurFormes
{
public:
    VecteurFormes(int capacite = 10);
    ~VecteurFormes();
    // ...
    void ajuster();
private:
    int taille;
    int capacite;
    Forme **data;
};

void VecteurFormes::ajuster()
{
    Forme **p;
    int destination = 0;

    if(taille < capacite)
    {
        p = new Forme * [taille];
        for(int i = 0 ; i < capacite ; i++)
        {
            if (data[i] != nullptr)
                p[destination++] = data[i];
        }
        delete [] data;
        data = p;
        capacite = taille;
    }
}
```


QUESTION 6

Considérons le programme C++ suivant :

Fichier composant.h

```
class Composant {
public:
    Composant();
    virtual ~Composant();
    virtual void decrire() const = 0;
};

class Resistance : public Composant {
public:
    Resistance(double v = 0);
    ~Resistance();
    void set_valeur(double v);
    void decrire() const;
    double valeur_ohm() const;
private:
    double valeur;
};

class Condensateur : public Composant {
public:
    Condensateur(double v = 0);
    ~Condensateur();
    void set_valeur(double v);
    void decrire() const;
    double valeur_farad() const;
private:
    double valeur;
};

class CircuitRC{
public:
    CircuitRC(Resistance*, Condensateur*);
    ~CircuitRC();
    double cste_temps();
private:
    Resistance* r;
```

```
Condensateur* c;  
};
```

Fichier composant.cpp

```
#include "iostream"
#include "composant.h"
using namespace std;

Composant::Composant() { cout << "Constructeur Composant" << endl; }
Composant::~~Composant() { cout << "Destructeur Composant" << endl; }

Resistance::Resistance(double v) {
    cout << "Constructeur Resistance" << endl;
    valeur = v;
}

Resistance::~~Resistance() { cout << "Destructeur Resistance" << endl; }

void Resistance::set_valeur(double v) { valeur = v; }

void Resistance::decrire() const
{ cout << "Resistance : " << valeur << " K-ohms" << endl; }

double Resistance::valeur_ohm() const { return (valeur*1e3); }

Condensateur::Condensateur(double v) {
    cout << "Constructeur Condensateur" << endl;
    valeur = v;
}

Condensateur::~~Condensateur()
{ cout << "Destructeur Condensateur" << endl; }

void Condensateur::set_valeur(double v) { valeur = v; }

void Condensateur::decrire() const
{ cout << "Condensateur : " << valeur << " nF" << endl; }

double Condensateur::valeur_farad() const { return (valeur*1e-9); }

CircuitRC::CircuitRC(Resistance* res, Condensateur* cond) {
    cout << "Constructeur CircuitRC" << endl;
    r = res;
```

```

        c = cond;
    }

CircuitRC::~CircuitRC() { cout << "Destructeur CircuitRC" << endl; }

double CircuitRC::cste_temps(){
    cout << "CircuitRC::cste_temps" << endl;
    return (r->valeur_ohm()*c->valeur_farad());
}

```

Fichier main.cpp

```
#include "iostream"
#include "composant.h"

using namespace std;

int main() {
    Resistance    r1(10);
    Condensateur *c1 = new Condensateur(0.47);
    cout << "---CircuitRC " << endl;
    CircuitRC    circuit(&r1, c1);
    cout << "constante de temps = " << circuit.cste_temps() << endl;
    delete c1;
    return 0;
}
```

Sous-question 6.1

Quel est la sortie à la console lors de l'exécution de ce code.

```
Constructeur Composant
Constructeur Resistance
Constructeur Composant
Constructeur Condensateur
---CircuitRC
Constructeur CircuitRC
CircuitRC::cste_temps
constante de temps = 4.7e-006
Destructeur Condensateur
Destructeur Composant
Destructeur CircuitRC
Destructeur Resistance
Destructeur Composant
```

Sous-question 6.2

Formuler un diagramme de classes pour le programme précédent.

