



Évaluation Finale: 40%

Identification du cours

| | |
|--------------------------|---------------------------------------------------------------------------|
| Nom du programme – Code: | COMPUTER SCIENCE TECHNOLOGY – VIDEO GAME PROGRAMMING – 420.AX – 420.BX |
| Titre du cours : | CONCEPTS ET PROGRAMMATION ORIENTÉE OBJECTS II |
| Numéro du cours : | 420-JV7-AS |
| Groupe : | 07343 |
| Nom de l'enseignante : | Salima Hassaine |
| Durée de l'évaluation | 3 périodes (150 minutes) |
| Session : | Automne 2020 |

Identification de l'étudiant

Nom: _____

Numéro d'étudiant : _____

Date: **2020 – 12 –09**

Résultat: _____

☐ Je déclare que ceci est un travail original et que j'ai mentionné toutes les sources du contenu dont je ne suis pas l'auteur (sources en ligne et imprimées, images, graphiques, films, etc.) dans le style de citation requis.

Standard des Compétences évaluées

Énoncé des compétences évaluées – Codes

Use an object-oriented development approach – 016T

Éléments de la compétence évalués

3. Programmer une classe.

Corriger des programmes – 0171

Éléments de la compétence évalués

2. Déterminer la nature du problème.

3. Corriger le problème.

Consignes

- Les notes de cours ne sont pas permis.
- Aucune pause ne sera accordée pendant l'examen. Aucun étudiant ne peut quitter la salle d'examen avant que la moitié du temps alloué ne se soit écoulée. L'étudiant qui quitte la salle d'examen ne peut y revenir (PIEA – Article 5.12.4).
- L'enseignant(e) ne répondra pas aux questions.
- Les étudiants ne sont pas autorisés à communiquer entre eux.
- Le plagiat, la tentative de plagiat ou la coopération à un plagiat lors d'une épreuve sommative entraîne la note zéro (0). Dans le cas de récidive, dans le même cours ou dans un autre cours, l'étudiant se voit octroyer un « 0 » pour le cours concerné. (PIEA – Article 5.16)

Consignes générales pour les évaluations en ligne :

- Se connecter au moins 10 minutes à l'avance de l'heure de l'examen.
- En cas de problème de connexion, contacter tout de suite l'enseignant par MIO ou TEAMS pour l'informer de la problématique. Ajouter une capture écran si possible.
- Votre caméra doit être ouverte à la demande de l'enseignant.
- Les écouteurs ne sont pas permis (sauf pour des exercices de compréhension à l'oral)
- Toute capture d'écran envoyée à la place de votre travail sera refusée.
- Assurez-vous d'enregistrer la dernière version avant la remise.
- Assurez-vous que vous avez bien envoyé votre examen avant la date limite.
- Matériel permis : Aucun
- Format du document : Fichier zip
- Plateforme utilisé pour l'envoi : Section Travaux dans Omnivox

Si nous estimons que vos réponses ne sont peut-être pas les vôtres, le département se réserve le droit de compléter votre évaluation par une réunion virtuelle afin de vérifier que vous avez bien atteint la compétence requise.

Répartition des points

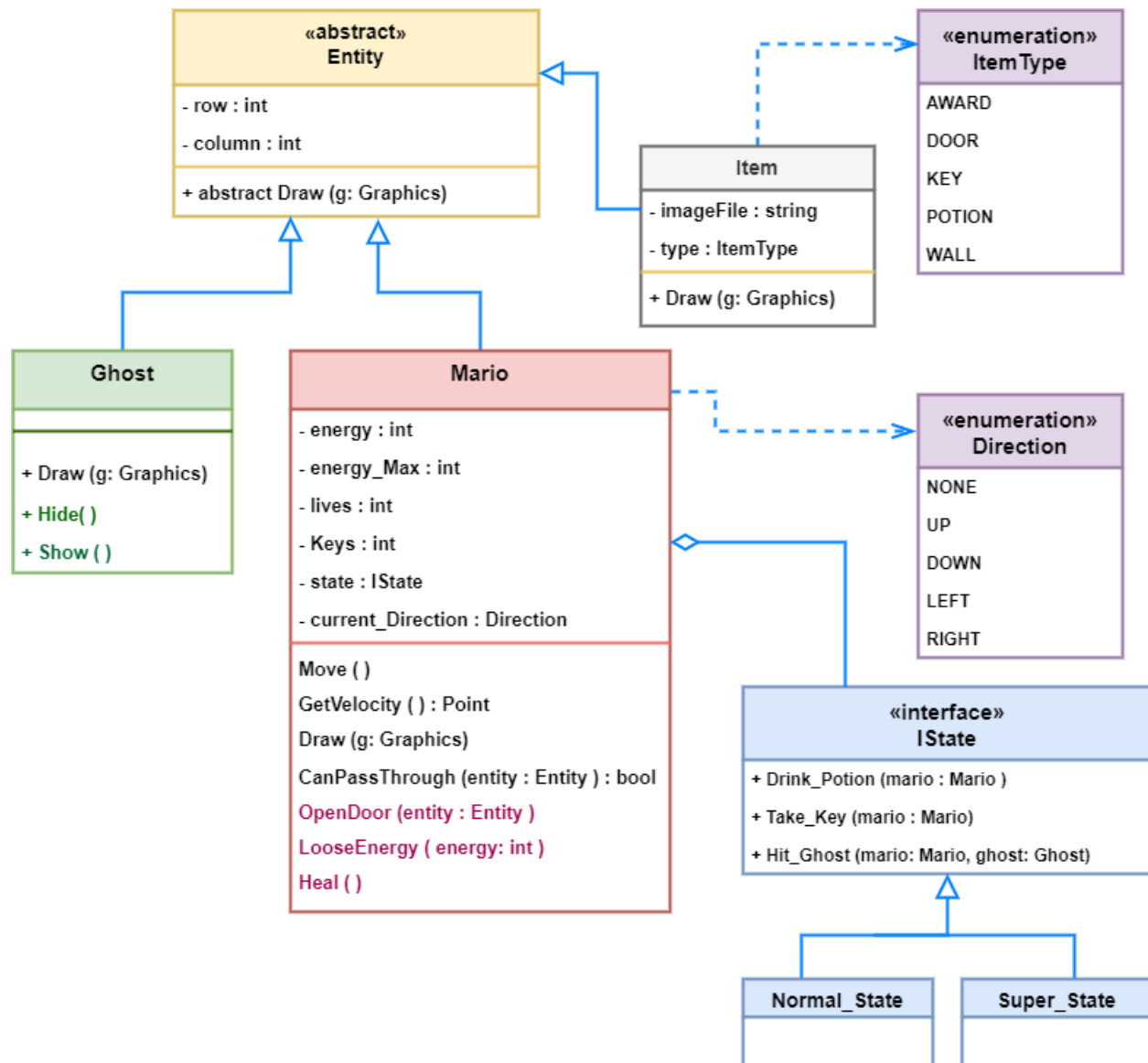
Cette évaluation est sur 100 points répartis comme suit :

| | |
|-----------------------------------------------------------|-----------------------------------|
| Partie 1: Héritage & Polymorphisme | Pour un total de 55 points |
| ▪ Question 1: Classe Abstraite Entity | 10 points |
| ▪ Question 2: Classe Abstraite LivingEntity | 20 points |
| ▪ Question 3: Classe Ghost | 05 points |
| ▪ Question 4: Classe Mario | 40 points |
| Partie 2: Patrons de Conception State et Singleton | Pour un total de 45 points |
| ▪ Question 1: Interface IState | 05 points |
| ▪ Question 2: Classe Normal_State | 20 points |
| ▪ Question 3: Classe Super_State | 20 points |
| Bonus Question: Delegates and Events | For a total of 10 points |

TOTAL: 100 POINTS + 10 points Bonus

Jeu du labyrinthe (Total 100 points)

Dans cet examen, vous développerez un jeu de labyrinthe en utilisant une application Windows Forms pour appliquer les concepts de la programmation orientée objet et les patrons de conception. Téléchargez le TEMPLATE Final_Exam.zip dans la section Travaux sur Omnivox, puis répondez aux questions suivantes :



Partie 1. Héritage & Polymorphisme (Total 55 points)

Question 1.1) Abstract Class Entity (10 points)

Créez la classe **abstraite Entity** comme suit :

- Ajoutez les champs suivants : **row** et **column** de type **int**, ces attributs ne doivent être accessibles que dans la classe **Entity** - (02 points)
- Encapsulez tous les champs – (02 points)
- Définissez un **constructeur** avec 2 paramètres (**int row**, **int column**) - (04 points)
- Ajoutez une méthode **abstraite void Draw (Graphics g)** – (02 points)

Question 1.2) Class Ghost (05 points)

Créez la classe **Ghost** qui hérite de la classe **Entity** comme suit:

- Ajoutez le champ suivant : **visible** de type **bool** - (02 points)
- Encapsulez le champ – (02 points)
- Définissez un **constructeur** avec 2 paramètres (**int row**, **int column**), tel que: le champ **visible** sera initialisé par default par **true** – (02 points)
- Définissez la méthode **void Hide ()**, qui initialise **visible** par **false** – (1.5 points)
- Définissez la méthode **void Show ()**, qui initialise **visible** par **true** – (1.5 points)

Ces deux dernières méthodes sont appelées dans la classe **GameController** pour cacher les fantômes et les faire apparaître de façon aléatoire dans le labyrinthe.

Question 1.4) Class Mario (40 points)


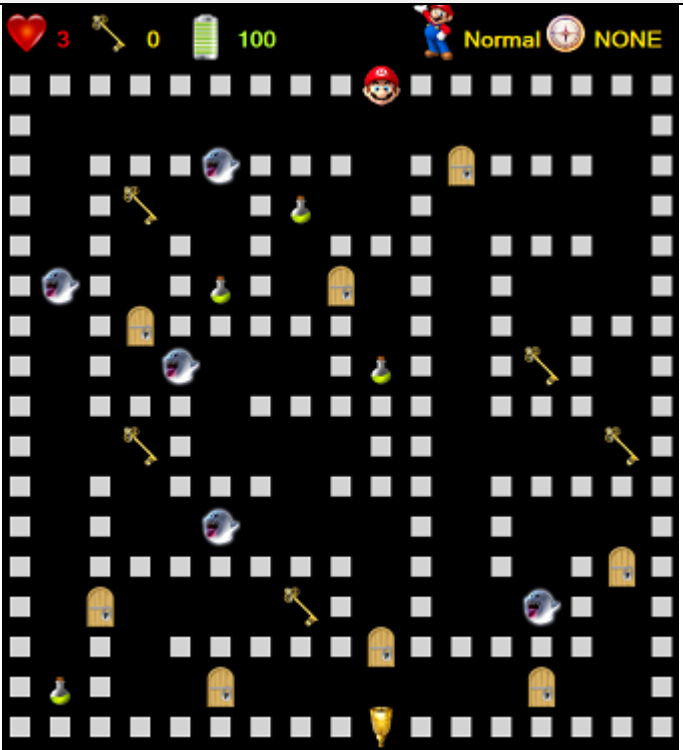





Créez la classe **Mario** qui hérite de la classe **Entity** comme suit:

- Ajoutez les champs suivants (seront tous accessibles seulement dans la classe **Mario**) :
 - **energy** et **energy_Max** de type **int** - (02 points)
 - **keys** et **lives** de type **int** - (02 points)
 - **current_Direction** de type **Direction**, et **state** de type **IState** - (02 points)
- Encapsuler tous les champs - (04 points)
- Définissez un **constructeur** avec 2 paramètres (**int row**, **int column**), comme suit: (05 points)
 - Les champs **energy** et **energy_Max** seront initialisés par **100**.
 - Le champ **keys** est initialisé par **0**.
 - Le champ **lives** est initialisé par **3**.
 - Le champ **current_Direction** est initialisé par **NONE**.
 - Le champ **state** est initialisé par une instance de la classe **Normal_State**
- Définissez la méthode **void Heal ()**, qui initialise **energy** par **energy_Max**. – (05 points)

- Définissez la méthode **void LooseEnergy (int energy)**, elle diminue l'énergie de mario en utilisant le paramètre energy. Puis, vérifiez si son énergie est inférieure ou égale à zéro, si oui alors diminuez la valeur de ses lives. Si lives est inférieure ou égale à zéro alors, appelez la méthode GameOver() de la classe GameController. Sinon (lives est supérieure à zéro), elle réinitialise l'énergie en appelant Heal(). **(10 points)**
- Définissez la méthode **bool OpenDoor (Entity entity)**, si l'entité est une instance de la classe **Item** et que son type est **Door**. Alors, vérifiez si la valeur de Keys est supérieure à zéro. Si oui, diminuez la valeur de Keys et retirez l'entité du labyrinthe, en appelant la méthode **RemoveEntity(int row, int column)** de la classe Maze. **(10 points)**

Partie 2: Patrons de Conception State et Singleton **(Total 45 points)**

Le comportement de l'objet Mario dépend de son état actuel qui change selon le tableau suivant :

| État actuel | Événement survenu | État et Comportement | Labyrinthe |
|--------------|-----------------------------------------------------------------------------------------------------|--------------------------------------------------------------|-------------------------------------------------------------------------------------|
| Normal State |  Drink_Potion | Mise à jour de son état par Super State Heal Mario |  |
| |  Take_Key | Mario incrémente ses Keys par +1 | |
| |  Hit_Ghost | Mario perd de l'energy | |
| Super State |  Drink_Potion | Heal Mario | |
| |  Take_Key | Mario incrémente ses Keys par +2 | |
| |  Hit_Ghost | Mise à jour de son état par Normal Stat | |

Question 2.1) Interface IState (05 points)

Définissez l'interface IState qui a 3 méthodes:

- **void Drink_Potion(Mario mario)**
- **void Take_Key(Mario mario)**
- **void Hit_Ghost (Mario mario, Ghost ghost)**

Question 2.2) Classe Normal State (20 points)

Créez la classe **Normal_State** qui implemente l'interface **IState** comme suit:

- Utilisez le patron de conception **Singleton** pour vous assurer que vous n'aurez qu'une seule instance de la classe **Normal_State** – (05 points)
- Définissez les méthodes de l'interface:
 - Définissez **void Drink_Potion(Mario mario)**, qui appelle **Heal()** de mario et initialise son **state** par une instance de **Super_State** – (05 points)
 - Définissez **void Take_Key (Mario mario)**, qui incrémente **Keys** de mario par 1– (05 points)
 - Définissez **void Hit_Ghost (Mario mario, Ghost ghost)**, qui décrémente **energy** de mario, en appelant **LooseEnergy(damage)**, tel que damage est une valeur aléatoire entre 20 et 30– (05 points)

Question 2.3) Classe Super State (20 points)

Créez la classe **Super_State** qui implemente l'interface **IState** comme suit:

- Utilisez le patron de conception **Singleton** pour vous assurer que vous n'aurez qu'une seule instance de la classe **Super_State** – (05 points)
- Définissez les méthodes de l'interface:
 - Définissez **void Drink_Potion(Mario mario)**, en appelant **Heal()** de mario (5 points)
 - Définissez **void Take_Key (Mario mario)**, qui incrémente **Keys** par 2 (5 points)
 - Définissez **void Hit_Ghost (Mario mario, Ghost ghost)**, et initialise son **state** par une instance de **Normal_State** – (05 points)

Question Bonus: Delegates et Events (10 points)

- Créez un **delegate** appelé **Winner**, qui représente les méthodes sans paramètres et avec un type de retour void. (02 points)
- Dans la classe Mario, ajoutez un **event** de type **Winner**, appelé **winnerEvent** qui fait référence à la méthode **GameWinner** de la classe **GameController** (l'événement doit être initialisé dans le constructeur de Mario) (02 points)
- Dans la méthode **Move()** de la classe Mario, invoquez l'événement **winnerEvent** si entity est une instance de la classe Award (04 points)

GRILLE DE CORRECTION - Partie 1 (Total 55 points)

Questions : Q1.2

| Element of competency: 3. Program a class | | | | | |
|-----------------------------------------------------------------------------------------------------------------------------|----------------------------|---------------------|-----------------------|------------------------------|--------------|
| Performance criteria | Highly satisfactory | Satisfactory | Unsatisfactory | Highly unsatisfactory | Total |
| 3.3 Proper use of the capacities of the language in applying the principles of encapsulation, inheritance and polymorphism. | 5 | 4.5 – 3 | 2.5 - 1 | 0 | /5 |

Questions : Q1. 1

| Element of competency: 3. Program a class | | | | | |
|-----------------------------------------------------------------------------------------------------------------------------|----------------------------|---------------------|-----------------------|------------------------------|--------------|
| Performance criteria | Highly satisfactory | Satisfactory | Unsatisfactory | Highly unsatisfactory | Total |
| 3.3 Proper use of the capacities of the language in applying the principles of encapsulation, inheritance and polymorphism. | 10 – 9.5 | 9 – 6 | 5.5 – 4 | 3.5 -0 | /10 |

Questions : Q1.3

| Element of competency: 3. Program a class | | | | | |
|-----------------------------------------------------------------------------------------------------------------------------|----------------------------|---------------------|-----------------------|------------------------------|--------------|
| Performance criteria | Highly satisfactory | Satisfactory | Unsatisfactory | Highly unsatisfactory | Total |
| 3.3 Proper use of the capacities of the language in applying the principles of encapsulation, inheritance and polymorphism. | 40 – 37 | 36 – 24 | 22 – 16 | 14 -0 | /40 |

GRILLE DE CORRECTION - Partie 2 (Total 45 points)

Questions : Q 2.1

| Element of competency: 3. Program a class. | | | | | |
|-------------------------------------------------------------------------------------------------------------|----------------------------|---------------------|-----------------------|------------------------------|--------------|
| Performance criteria | Highly satisfactory | Satisfactory | Unsatisfactory | Highly unsatisfactory | Total |
| 3.2 Declaration and definition of the class, respecting the syntactical and semantic rules of the language. | 5 | 4.5 – 3 | 2.5 - 1 | 0 | /5 |

Questions : Q2.2, Q 2.3

| Element of competency: 3. Program a class | | | | | |
|-----------------------------------------------------------------------------------------------------------------------------|----------------------------|---------------------|-----------------------|------------------------------|--------------|
| Performance criteria | Highly satisfactory | Satisfactory | Unsatisfactory | Highly unsatisfactory | Total |
| 3.3 Proper use of the capacities of the language in applying the principles of encapsulation, inheritance and polymorphism. | 10 – 9.5 | 9 – 6 | 5.5 – 4 | 3.5 -0 | /10 |