Assessment Cover Sheet

This Assessment Cover Sheet is only to be attached to hard copy submission of assessments.

ASSESSMENT DETAILS

Unit title	DATA STRUCTURES AND PATTERNS	Tutorial /Lab Group	2	Office use only
Unit code	COS30008	Due date	20/9/2024	
Name of lecturer/tutor	DR. MARK TEE KIT TSUN			
Assignment title	PROBLEM SET 1			Faculty or school date stamp

STUDENT(S) DETAILS

	Student Name(s)	Student ID Number(s)
(1)	MURUNGI ALLAN CHEBOIWO	102760283
(2)		
(3)		
(4)		
(5)		
(6)		

DECLARATION AND STATEMENT OF AUTHORSHIP

1. I/we have not impersonated, or allowed myself/ourselves to be impersonated by any person for the purposes of this assessment.
2. This assessment is my/our original work and no part of it has been copied from any other source except where due acknowledgement is made.
3. No part of this assessment has been written for me/us by any other person except where such collaboration has been authorised by the lecturer/tutor concerned.
4. I/we have not previously submitted this work for this or any other course/unit.
5. I/we give permission for my/our assessment response to be reproduced, communicated, compared and archived for plagiarism detection, benchmarking or educational purposes.

I/we understand that:

6. Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a form of cheating and is a very serious academic offence that may lead to exclusion from the University. Plagiarised material can be drawn from, and presented in, written, graphic and visual form, including electronic data and oral presentations. Plagiarism occurs when the origin of the material used is not appropriately cited.

Student signature/s

I/we declare that I/we have read and understood the declaration and statement of authorship.

(1)		(4)	
(2)		(5)	
(3)		(6)	

Further information relating to the penalties for plagiarism, which range from a formal caution to expulsion from the University is contained on the Current Students website at **https://www.swinburne.edu.my/current-students/manage-course/exams-results-assessment**

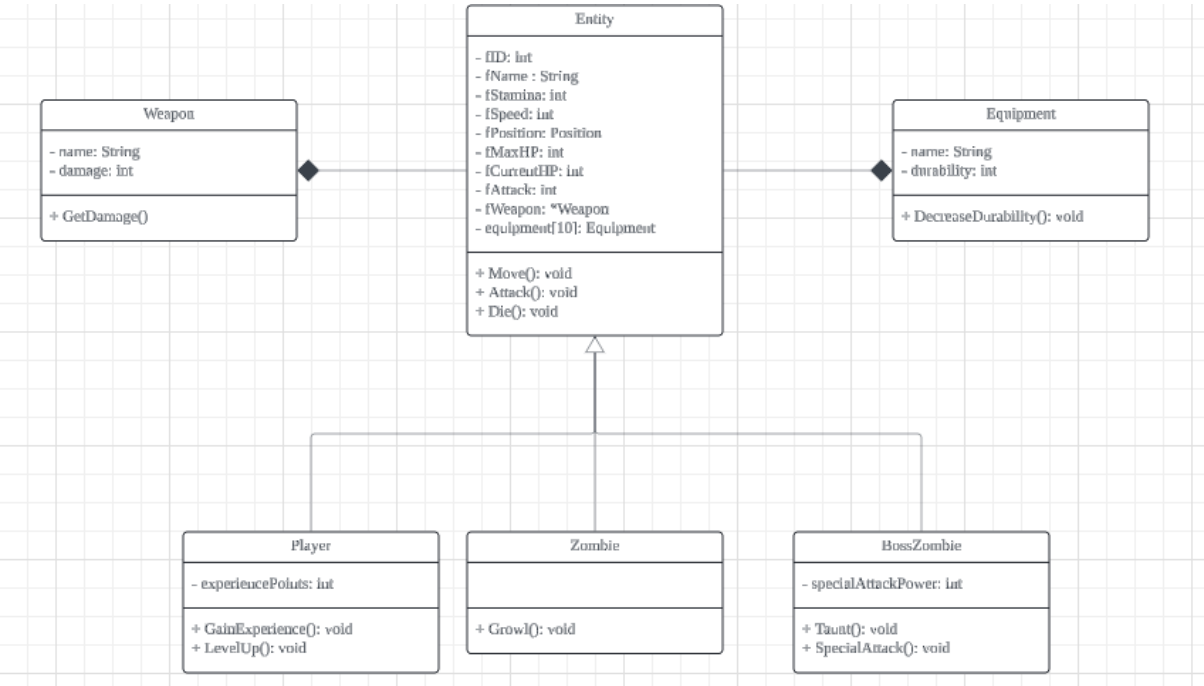
Copies of this form can be downloaded from the Student Forms web page at **https://www.swinburne.edu.my/current-students/manage-course/exams-results-assessment/how-to-submit-work.php**

Problem Set 1

Task 1: Conceptual Design of Game Content

The title of the game I plan to create is 'Escaping Zombies I', classified under the genres 'shooter games' and 'survival games'. This game consists of a player character called June, whose mission is to climb a hill and reach the summit in order to proceed to a higher level of the game. This hill will have zombies that June has to conquer using weapons on his way up. June is a first-person shooter, meaning he will be played within the protagonist's perspective (Wikipedia 2024). The main objective is for June to defeat zombies, level up and reach the summit while upgrading equipment and maintaining health and stamina.

Class Diagram



External Link to the UML diagram: https://lucid.app/lucidchart/af664b15-7fd2-47ec-a3d2-ba258705a92b/edit?viewport_loc=-218%2C-638%2C2650%2C1014%2C0_0&invitationId=inv_28fe9d38-4e5f-40e0-a665-a88f250d31b0

Task 2: Base Class for all Units and Creating Derived Classes

Full Source Code

Header Files

Entity.h

```
#pragma once

#include <iostream>

#include <string>

#include "Weapon.h"

#include "Equipment.h"


using namespace std;


struct Position {

  float x;

  float y;

};


class Entity {

protected:

  int fID; // unique identifier for this Entity

  string fName; // Entity's name

  int fStamina; // Entity's stamina during the game

  int fSpeed; // Entity's speed while moving

  Position fPosition; // Entity's position in the game

  int fMaxHP; // Entity's maximum health

  int fCurrentHP; // Entity's current health

  int fAttack; // Entity's attack damage

  Weapon* fWeapon; // Pointer to Weapon class

  Equipment equipment[10]; // Array of Equipment


public:
```

```
    // Default constructor
    Entity();

    // Entity and its attributes
    Entity(int id, const string& name, int stamina, int speed, Position position, int maxHP, int currentHP, int attack, Weapon* weapon);

    // Getters
    int GetID();
    string GetName();
    int GetStamina();
    int GetSpeed();
    Position GetPosition();
    int GetMaxHP();
    int GetCurrentHP();
    int GetAttack();
    Weapon* GetWeapon();

    // Setters
    void SetfID(int id);
    void SetfName(const string& name);
    void SetfStamina(int stamina);
    void SetfSpeed(int speed);
    void SetfPosition(Position position);
    void SetfMaxHP(int maxHP);
    void SetfCurrentHP(int currentHP);
    void SetfAttack(int attack);
    void SetfWeapon(Weapon* weapon);

    // Methods
    void Move(int deltaX, int deltaY); // Move the entity
```

```
    void Attack(Entity& target); // Attack a target

    void Die(); // Handle the death of the entity

    // Destructor
    virtual ~Entity();
};
```

Player.h

```
#pragma once
#include "Entity.h"
using namespace std;

class Player : public Entity {
private:
    int experiencePoints;

public:
    // Constructor
    Player(int id, const string& name, int stamina, int speed, Position position, int maxHP, int currentHP, int attack, Weapon* weapon, int xp);

    // Player-specific methods
    void GainExperience(int xp);
    void LevelUp();

    // Destructor
    virtual ~Player();
};
```

Zombie.h

```
#pragma once
```

```
#include "Entity.h"

using namespace std;


class Zombie : public Entity {

public:

  // Constructor

  Zombie(int id, const string& name, int stamina, int speed, Position position, int maxHP, int currentHP, int attack, Weapon* weapon);


  // Zombie-specific method

  void Growl();


  // Destructor

  virtual ~Zombie();

};
```

Weapon.h

```
#pragma once

#include <string>

using namespace std;


class Weapon {

private:

  string name;

  int damage;


public:

  // Constructor

  Weapon(const string& name = "Default Weapon", int damage = 10);


  // Getters

  string GetName() const;

  int GetDamage() const;
```

```
    // Setters
    void SetName(const string& name);
    void SetDamage(int damage);

    // Destructor
    ~Weapon();
};
```

Equipment.h

```
#pragma once
#include <string>
using namespace std;

class Equipment {
private:
    string name;
    int durability;

public:
    // Constructor
    Equipment(const string& name = "Default", int durability = 100);

    // Getters
    string GetName() const;
    int GetDurability() const;

    // Setters
    void SetName(const string& name);
    void SetDurability(int durability);

    // Decrease durability method
```

```
    void DecreaseDurability(int amount);

    // Destructor
    ~Equipment();
};
```

BossZombie.h

```
#pragma once
#include "Entity.h"
using namespace std;

class BossZombie : public Entity {
private:
    int specialAttackPower; // Additional attack power for special moves

public:
    // Constructor
    BossZombie(int id, const string& name, int stamina, int speed, Position position, int maxHP, int currentHP, int attack, Weapon* weapon, int specialAttackPower);

    // Special attack method
    void SpecialAttack(Entity& target);

    // Taunt method
    void Taunt();

    // Destructor
    virtual ~BossZombie();
};
```

Source Files

Main.cpp

```
#include "Entity.h"
```

```
#include "Weapon.h"
#include "Equipment.h"
#include <iostream>

using namespace std;

int main() {
    // Variables for entity creation
    int id, stamina, speed, maxHP, currentHP, attack;
    float posX, posY;
    string name;
    Weapon* weapon = nullptr;

    // Input from the user
    cout << "Enter Entity ID: ";
    cin >> id;
    cout << "Enter Entity Name: ";
    cin.ignore(); // To avoid issues with getline after cin
    getline(cin, name);
    cout << "Enter Stamina: ";
    cin >> stamina;
    cout << "Enter Speed: ";
    cin >> speed;
    cout << "Enter Position X: ";
    cin >> posX;
    cout << "Enter Position Y: ";
    cin >> posY;
    cout << "Enter Max HP: ";
    cin >> maxHP;
    cout << "Enter Current HP: ";
    cin >> currentHP;
```

```
    cout << "Enter Attack Power: ";
    cin >> attack;

    // Initialize Position
    Position entityPosition = { posX, posY };

    // Initialize Weapon
    weapon = new Weapon("Sword", 25); // Example weapon

    // Create an Entity using user input
    Entity myEntity(id, name, stamina, speed, entityPosition, maxHP, currentHP, attack, weapon);

    // Example: Move and attack demonstration
    myEntity.Move(5, 10);  // Move the entity
    Entity enemyEntity(2, "Enemy", 80, 1, { 10.0f, 10.0f }, 50, 50, 10, nullptr); // Create an enemy entity
    myEntity.Attack(enemyEntity); // Attack the enemy

    // Clean up dynamically allocated memory
    delete weapon;

    return 0;
}
```

Entity.cpp

```
#include "Entity.h"
using namespace std;

Entity::Entity()
        : fID(0), fName("June"), fStamina(99), fSpeed(1), fPosition({ 0.0f, 0.0f }),
        fMaxHP(99), fCurrentHP(99), fAttack(99), fWeapon(nullptr) {
        for (int i = 0; i < 10; i++) {
```

```
            equipment[i] = Equipment();
        }
        cout << "Default Entity has been created." << endl;
}

Entity::Entity(int id, const string& name, int stamina, int speed, Position position, int maxHP, int currentHP, int attack, Weapon* weapon)
        : fID(id), fName(name), fStamina(stamina), fSpeed(speed), fPosition(position),
        fMaxHP(maxHP), fCurrentHP(currentHP), fAttack(attack), fWeapon(weapon) {
        cout << "Entity " << fID << " has been created." << endl;
}

// Getters
int Entity::GetID() {
        return fID;
}

string Entity::GetName() {
        return fName;
}

int Entity::GetStamina() {
        return fStamina;
}

int Entity::GetSpeed() {
        return fSpeed;
}

Position Entity::GetPosition() {
        return fPosition;
```

```
}

int Entity::GetMaxHP() {
	return fMaxHP;
}

int Entity::GetCurrentHP() {
	return fCurrentHP;
}

int Entity::GetAttack() {
	return fAttack;
}

Weapon* Entity::GetWeapon() {
	return fWeapon;
}

// Setters
void Entity::SetfID(int id)
{
	fID = id;
}

void Entity::SetfName(const string& name)
{
	fName = name;
}

void Entity::SetfStamina(int stamina)
{
```

```
	fStamina = stamina;
}

void Entity::SetfSpeed(int speed)
{
	fSpeed = speed;
}

void Entity::SetfPosition(Position position) {
	fPosition = position;
}

void Entity::SetfMaxHP(int maxHP)
{
	fMaxHP = maxHP;
}

void Entity::SetfCurrentHP(int currentHP)
{
	if (currentHP >= 0 && currentHP <= fMaxHP) {
		fCurrentHP = currentHP;
	}
	else {
		cout << "Invalid HP value!" << endl;
	}
}

void Entity::SetfAttack(int attack)
{
	fAttack = attack;
}
```

```
void Entity::SetfWeapon(Weapon* weapon)
{
	fWeapon = weapon;
}



// Movement method
void Entity::Move(int deltaX, int deltaY) {
	fPosition.x += deltaX * fSpeed;
	fPosition.y += deltaY * fSpeed;
	cout << fName << " moved to (" << fPosition.x << ", " << fPosition.y << ")." << endl;
}

// Attack method
void Entity::Attack(Entity& target) {
	if (fWeapon) {
		cout << fName << " attacks " << target.GetName() << " with " << fWeapon-
>GetName() << endl;
		target.SetfCurrentHP(target.GetCurrentHP() - fWeapon->GetDamage());
		cout << target.GetName() << " now has " << target.GetCurrentHP() << " HP left."
<< endl;
	}
	else {
		cout << fName << " has no weapon to attack with!" << endl;
	}
}

// Die method
void Entity::Die() {
	cout << fName << " has died." << endl;
}
```

```
// Destructor
Entity::~Entity() {
        cout << "Entity Destructor called." << endl;
}
```

Zombie.cpp

```
#include "Zombie.h"
#include <iostream>
using namespace std;

Zombie::Zombie(int id, const string& name, int stamina, int speed, Position position, int maxHP, int currentHP, int attack, Weapon* weapon)
  : Entity(id, name, stamina, speed, position, maxHP, currentHP, attack, weapon) {
  cout << "Zombie " << name << " has been created." << endl;
}

void Zombie::Growl() {
  cout << fName << " growls menacingly." << endl;
}

Zombie::~Zombie() {
  cout << "Zombie " << fName << " has been destroyed." << endl;
}
```

Player.cpp

```
#include "Player.h"
#include <iostream>
using namespace std;

Player::Player(int id, const string& name, int stamina, int speed, Position position, int maxHP, int currentHP, int attack, Weapon* weapon, int xp)
```

```
    : Entity(id, name, stamina, speed, position, maxHP, currentHP, attack, weapon),
experiencePoints(xp) {
    cout << "Player " << name << " has been created." << endl;
}

void Player::GainExperience(int xp) {
    experiencePoints += xp;
    cout << fName << " gained " << xp << " experience points." << endl;
}

void Player::LevelUp() {
    if (experiencePoints >= 100) {
        fMaxHP += 10;
        fAttack += 5;
        experiencePoints = 0;
        cout << fName << " has leveled up! New Max HP: " << fMaxHP << ", Attack: " << fAttack <<
endl;
    }
}

Player::~Player() {
    cout << "Player " << fName << " has been destroyed." << endl;
}
```

Equipment.cpp

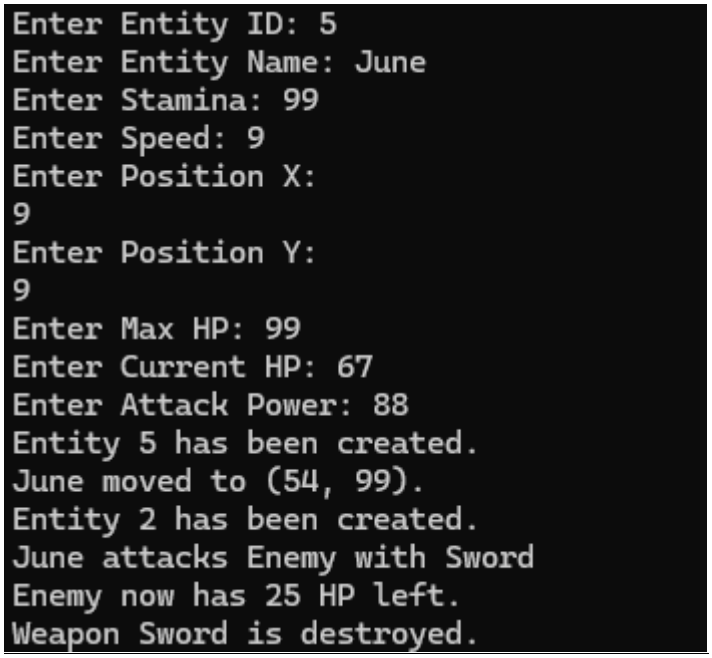
```
#include "Equipment.h"
#include <iostream>
using namespace std;

Equipment::Equipment(const string& name, int durability)
    : name(name), durability(durability) {
}
```

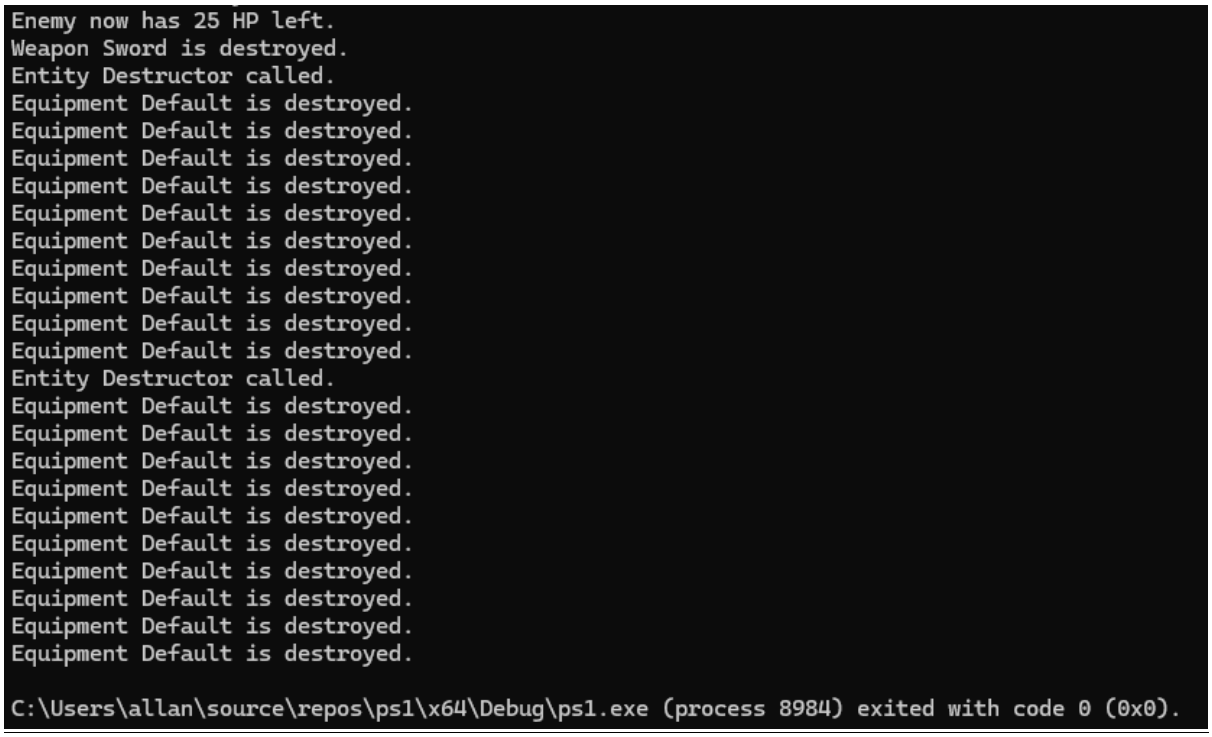
```
string Equipment::GetName() const {
   return name;
}

int Equipment::GetDurability() const {
   return durability;
}

void Equipment::SetName(const string& name) {
   this->name = name;
}

void Equipment::SetDurability(int durability) {
   this->durability = durability;
}

void Equipment::DecreaseDurability(int amount) {
   if (durability - amount >= 0) {
      durability -= amount;
   }
   else {
      durability = 0;
   }
}

Equipment::~Equipment() {
   cout << "Equipment " << name << " is destroyed." << endl;
}
```

BossZombie.cpp

```
#include "BossZombie.h"

#include <iostream>


using namespace std;


BossZombie::BossZombie(int id, const string& name, int stamina, int speed, Position position, int maxHP, int currentHP, int attack, Weapon* weapon, int specialAttackPower)

   : Entity(id, name, stamina, speed, position, maxHP, currentHP, attack, weapon), specialAttackPower(specialAttackPower) {

   cout << "BossZombie " << name << " has been created." << endl;

}


void BossZombie::SpecialAttack(Entity& target) {

   int totalDamage = fAttack + specialAttackPower;

   cout << fName << " uses a special attack on " << target.GetName() << " causing " << totalDamage << " damage!" << endl;

   target.SetfCurrentHP(target.GetCurrentHP() - totalDamage);

   cout << target.GetName() << " now has " << target.GetCurrentHP() << " HP left." << endl;

}


void BossZombie::Taunt() {

   cout << fName << " taunts the enemies with a menacing laugh!" << endl;

}


BossZombie::~BossZombie() {

   cout << "BossZombie " << fName << " has been destroyed." << endl;

}
```

Screenshots





Task 3: Unit Communications

References

Wikipedia. *List of Video Game Genres. Shooter Games.* (July 19, 2024) https://en.wikipedia.org/wiki/List_of_video_game_genres