Homework - Classes & Objects
Due: 03-25-2025

Part A: Conceptual Questions

1. What is a class in object-oriented programming?

- A class is like a blueprint that defines how something should behave or what it should have. It groups related data and functions together. In programming, we use it to create objects that follow that structure.

2. What is an object, and how does it relate to a class?

- An object is an actual instance of a class. While a class defines the structure, the object is something real that we can use in the program. It has actual values for the data members defined in the class.

3. Define a constructor. What is its role in a class?

- A constructor is a special function inside a class that runs automatically when we create an object. Its job is to set up the object's initial state, like giving starting values to variables.

4. Define a destructor. Why is it important in managing an object's lifecycle?

- A destructor is another special function that runs when the object is destroyed. It's used to clean up things like memory or files that the object used so that the program doesn't waste resources.

5. Briefly describe the lifecycle of an object from instantiation to destruction.

- First, an object is created when we call the constructor. Then, it can be used during the program. When it's no longer needed or goes out of scope, the destructor is called and the object is destroyed.

6. Why is it important for a class to manage its resources (e.g., memory) during its lifecycle?

- If a class doesn’t manage its resources, things like memory can be wasted and not released properly. This can lead to memory leaks, which can make the program slow or even crash over time.

Part B: Minimal Coding Example (C++)

```
#include <iostream>
#include <string>
using namespace std;

class Creature {
private:
    string name;
    int health;

public:
    // Constructor
    Creature(string creatureName, int creatureHealth) {
        name = creatureName;
        health = creatureHealth;
        cout << "Creature " << name << " has entered the world with " << health << "
HP." << endl;
    }

    // Destructor
    ~Creature() {
        cout << "Creature " << name << " has been destroyed." << endl;
    }

    // Public method to display the creature's status
    void displayStatus() {
        cout << "Name: " << name << ", Health: " << health << endl;
    }
};

int main() {
    Creature c1("Shadowfang", 100);
    c1.displayStatus();
```

```
    return 0;
}
```

Explanation:
In this example, the constructor initializes the creature's name and health when the object is created. The destructor prints a message when the object is destroyed, showing that its life cycle has ended. This demonstrates how the constructor sets up the object at the beginning, and the destructor handles cleanup at the end.

Part C: Reflection & Short-Answer

Importance of Constructors

- Constructors are helpful because they make sure that an object starts out with valid data. Without them, we'd have to manually set everything, and it could lead to mistakes or uninitialized variables.

Role of Destructors

- Destructors are important in C++ since there's no automatic garbage collection like in some other languages. They help clean up memory and other resources so we don't run into problems like memory leaks.

Lifecycle Management

- If a class doesn't manage its resources well, things like memory or file handles could be used up and never freed. This can make the program less efficient or even cause it to crash after running for a while.