Heshawa Cooray

Part A: Conceptual Questions

Inheritance Definition
Inheritance is when a class takes on properties and behaviors from another class. It helps reuse code and create a hierarchy. Unlike composition, which uses objects of other classes inside a class, inheritance extends to an existing class.

Types of Inheritance

1. **Single Inheritance** – One class inherits from another. Example: Car inherits from Vehicle.
2. **Multiple Inheritance** – A class inherits from more than one base class. Example: FlyingCar inherits from both Car and Aircraft.

Overriding Methods
Overriding allows a derived class to change how a method from the base class works. Instead of just adding a new method, overriding ensures consistent behavior while making necessary modifications. Example: A Vehicle class has drive(), but a Car class overrides it to specify how a car drives.

Real-World Analogy
A smartphone "inherits" features from basic phones (calling, texting) but extends them with internet and apps. The private/internal hardware system isn't exposed, just like OOP inheritance keeps some things private.

Part B: Minimal Coding

```
class Vehicle {
protected:
   string brand;
public:
   Vehicle(string b) : brand(b) {}
   virtual void drive() { cout << "Vehicle is driving" << endl; }
};

class Car : public Vehicle {
private:
   int doors;
public:
   Car(string b, int d) : Vehicle(b), doors(d) {}
   void drive() override { cout << "Car is driving with " << doors << " doors." << endl; }
};

int main() {
   Vehicle v("Generic");
   Car c("Toyota", 4);
   v.drive();
   c.drive();
   return 0;
}
```

Part C: Reflection & Discussion

When to Use Inheritance

- **Good Use**: A Bird class that extends Animal, keeping common attributes while adding flight-specific ones.
- **Bad Use**: Using inheritance just to share methods between unrelated classes instead of composition.

Overriding vs. Overloading

- **Overriding** happens when a derived class changes a base class method (runtime).
- **Overloading** is defining multiple methods with the same name but different parameters.
  Inheritance uses overriding to allow flexibility and specialization.

Inheritance vs. Interfaces/Abstract Classes
Abstract classes provide a blueprint with some implementation, while interfaces (in Java) define only method signatures. Inheritance is about code reuse; interfaces enforce structure.

Pitfalls of Multiple Inheritance
A class inheriting from two classes with the same method (drive()) can cause ambiguity. Solution: Use virtual inheritance or interfaces.

Part D: Research

Inheritance in Different Languages

- **C++** allows multiple inheritance but can lead to complexity.
- **Java** only supports single class inheritance but allows multiple interfaces.

Open-Closed Principle
A class should be open for extension but closed for modification. Example: Adding new vehicle types by creating new classes (Bike, Truck) that inherit from Vehicle, instead of modifying Vehicle itself.