Lab 10: Inheritance and Object Relations

Course Code: CSE 4302
Week: 10
Topics: Function Overriding, Multi-Level Inheritance, Multiple Inheritance, Aggregation, Virtual Functions
Instructors:
Md. Bakhtiar Hasan, Assistant Professor, CSE
Ishmam Tashdeed, Lecturer, CSE
Submission Deadline: 12:00 P.M., January 28, 2026

Upload and Turn In your `.cpp` files to **Lab 10** on **Google Classroom**. If you are unable to finish all the tasks, **submit the tasks you have completed within the given time** (You have the chance to complete the remaining ones individually later on and demonstrate them during the next week's lab evaluation). No late submissions are allowed.
DO NOT forget to rename the files like `XXX_T1.cpp`, `XXX_T2.cpp`, ... (Replace `XXX` with the last three digits of your student ID).
After the submission deadline, you will undergo a one-on-one evaluation by one of the course instructors. The instructor will ask you to show your submissions on Google Classroom. They may ask you to run the code locally and ask you questions about it.

Lab Tasks

1. A Class of Their Own - Part 2

You need to extend "A Class of Their Own" from the previous lab. The university administration system is being updated, and you are in charge of modeling the hierarchy of people on campus. Everyone is a person, but a student has more specific data, and a student with scholarship has even more. **You can use the solution to previous lab tasks as necessary.**
Class Specification:

- `Person` (Base Class)
  - **Member Variables:**
    - `name` (`String`)
    - `age` (`int`)
  - **Member Functions:**
    - Constructor to set these values (along with sensible defaults).
    - `displayDetails()`: Prints the name and age formatted neatly.
    - `getRole()`: Returns a string describing the person's role (default: "General Person").
- `Student` (Derived from `Person`)

- **Member Variable:**
  - `studentID` (`String`)
  - `currentCGPA` (`float`)
- **Member Functions:**
  - Constructor to set these values of the derived and base class (along with sensible defaults).
  - `displayDetails()`: Overrides the parent function. It must explicitly call the `Person`'s `displayDetails()` first to get the name and age, then print the ID and CGPA on new lines.
  - `getRole()`: Overrides to return "Student".
  - `updateCGPA()`: Updates the student's CGPA.
- **Constraints:**
  - Ensure that the members inherited from Person cannot be directly accessed by an object of Student class or any subclass of Student. You can add any number of functions to indirectly access these members if necessary.

- `ScholarshipStudent` (Derived from `Student`)
  - **Member Variable:**
    - `scholarshipPercentage` (`float`)
    - `minCGPARequirement` (`float`)
  - **Member Functions:**
    - Constructor to set these values of the derived and base class (along with sensible defaults).
    - `displayDetails()`: Overrides the parent function. It calls the `Student`'s `displayDetails()` first to get the basic information, then prints the scholarship percentage.
    - `checkEligibility()`: Checks if the CGPA of the student is greater than or equal to the minimum requirement. Prints a success or warning message based on the result.
    - `getRole()`: Overrides to return "Scholarship Student".
- `Faculty` (New class derived from `Person`)
  - **Member Variable:**
    - `employeeID` (`String`)
    - `Department` (`String`)
    - `CoursesTaught` (int): numbers of courses taught
  - **Member Functions:**
    - Constructor to set these values of the derived and base class (along with sensible defaults).
    - `displayDetails()`: Overrides to show name, age, employee ID, department, and number of courses taught.
    - `getRole()`: Overrides to return "`Faculty Member`".
    - `assignNewCourse()`: Increments the number of courses taught by 1.

- Create an array of `Person*` pointers containing four different individuals (One object from each class). Iterate through the array and call `displayDetails()` on each person. Each should execute their specific version of the function:
    - Basic person shows only name and age
    - Student adds ID and CGPA
    - Scholarship student adds scholarship percentage
    - Faculty shows employee details

  Iterate again and call `getRole()` on each person to collect their role information. For all `Student` objects in the array (both `Student` and `ScholarshipStudent`), call a function to check if they're eligible for academic awards (CGPA $\geq$ 3.5). You must use type checking/dynamic casting to identify Student objects within the `Person*` array. For all `ScholarshipStudent` objects, additionally call `checkEligibility()` to verify if they maintain their scholarship requirements.

2. Ctrl + Alt + Defend - Part 2

You need to extend the previous task "Ctrl + Alt + Defend". **You can use the solution to previous lab tasks as necessary.**

Class Specification:

- `AutonomousUnit` (New Abstract Base Class)
    - **Member Variables:**
        - `unitID(String)`
        - `operationalStatus (bool)`
    - **Member Functions:**
        - Constructor to set these values (along with sensible defaults).
        - `executeMission()`: Abstract method to be implemented by all autonomous units.
        - `getUnitType():` Returns unit classification string.
        - `diagnosticCheck():` Prints system status.
        - Destructor
- `VehicleUnit` (Derived from `AutonomousUnit`)
    - **Member Variables:**
        - `maxSpeed (int)`
        - `fuelLevel (float)`
        - `currentLocationX (int)`
        - `currentLocationY (int)`
    - **Member Functions:**
        - Constructor to set these values (along with sensible defaults).
        - Getters and setters.
        - Destructor that prints "`Vehicle going dark.`".

        - `move()`: Decreases fuel by one unit and prints "`Moving to [X,Y] at speed [maxSpeed]. Fuel remaining: [fuelLevel].`".
        - `executeMission()`: Overrides to perform vehicle-specific mission tasks.
        - `getUnitType()`: Returns "`Ground Vehicle`".
- `CameraUnit` (Derived from AutonomousUnit)
    - **Member Variables:**
        - `nightVisionMode` (`bool`)
        - `storageCapacity` (`int`)
        - `recordingQuality` (`string`, e.g., "HD", "4K")
    - **Member Functions:**
        - Constructor to set these values (along with sensible defaults).
        - Getters and setters.
        - Destructor that prints "`Camera going dark.`".
        - `recordFootage()`: Reduces storage capacity by one unit and prints "`Recording... Night vision is [On/Off]. Storage remaining: [storageCapacity] GB.`".
        - `executeMission()`: Overrides to perform vehicle-specific mission tasks.
        - `getUnitType()`: Returns "Surveillance Unit".
- `WeaponArm` (Component Class)
    - **Member Variables:**
        - `ammoCount` (`int`)
        - `damage` (`int`)
        - `weaponType` (`string`, e.g., "Kinetic", "Energy")
    - **Member Functions:**
        - Constructor to set these values (along with sensible defaults).
        - Getters and setters.
        - Destructor that prints "`Weapon going dark.`".
        - `fire()`: Checks if `ammoCount > 0`. If yes, decrements `ammo` and prints "`Bang! Dealt [damage] damage.`" If not, prints "`Click... Out of ammo.`".
        - `reload(int amount)`: Adds to the ammo count.
        - `getWeaponInfo()`: Returns formatted weapon specifications.
- `MissionObjective` (New Abstract class)
    - **Member Functions:**
        - `planExecution()`: Abstract planning method.
        - `execute()`: Abstract execution method.
        - `getMissionType()`: Returns mission classification.
        - `difficultyLevel()`: Returns integer difficulty (1-10).
- `ReconMission` (Derived from `MissionObjective`)
    - **Member Variables:**
        - `areaToScan` (`string`)

        - `intelPriority (int)`
    - **Member Functions:**
        - Constructor to set these values (along with sensible defaults).
        - `planExecution()`: Prints "`Planning reconnaissance of [areaToScan]. Priority: [intelPriority].`"
        - `execute()`: Prints "`Executing stealth surveillance protocol.`"
        - `getMissionType()`: Returns "`Reconnaissance`".
- `DefenseMission`(Derived from MissionObjective)
    - **Member Variables:**
        - `perimeterToGuard (string)`
        - `threatLevel (int)`
    - **Member Functions:**
        - Constructor to set these values (along with sensible defaults).
        - `planExecution()`: Prints "`Establishing defense at [perimeterToGuard]. Threat: [threatLevel].`"
        - `execute()`: Prints "`Activating defensive countermeasures.`"
        - `getMissionType()`: Returns "`Defense`".
- `PatrolBot` (Derived from `VehicleUnit` and `CameraUnit`)
    - **Member Variables:**
        - `modelNumber` (`string`)
        - A `WeaponArm` object
        - `assignedMissions (MissionObjective** pointer array)`
        - `missionCount (int)`
    - **Member Functions:**
        - A constructor that takes arguments for Vehicle, Camera, model number, Weapon Arm, and mission capacity. Must use Member Initialization List. The value of the `WeaponArm` objects should be set accordingly.
        - Destructor that prints "`[modelNumber] going dark.`".
        - `patrolArea()`: Moves the bot and records footage.
        - `engageTarget()`: Fires the weapon.
        - `statusReport()`: Display the status of the vehicle, camera, and weapon.
        - `assignMission(MissionObjective newMission)`: Adds mission to the mission array.
        - `executeAllMissions()`: Iterates through the mission array and executes each mission.
        - virtual `getCapabilityReport()`: Returns detailed capabilities assessment.
- `main()` Function Requirements
    - Create one `PatrolBot` object.
    - Simulate a patrol.
    - Simulate an encounter by engaging with the target multiple times until ammo runs out.

- Reload the weapon.