GLS University
Faculty of Computer Application and Information Technology
MCA – Semester – I

Web Development Using Python Framework (230701106)

Python OOPs Programming Assignment

1. Create a class **Car** with attributes **make** and **model**. Create an object and print its attributes.
2. Create a class **Calculator** with methods to add, subtract, multiply, and divide two numbers.
3. Create a class **Person** with a constructor and destructor method to print messages when an object is created and destroyed.
4. Create a class **Employee** with a class variable **employee_count** to keep track of the number of employees. Increment it in the constructor.
5. Create a class **Rectangle** with attributes **length** and **width**. Calculate the area using instance variables.
6. Create a class **Person** with a private attribute **__age**. Provide getter and setter methods to access and modify the age.
7. Create a static method in a class **MathUtils** to check if a number is even.
8. Create a class method in a class **MathUtils** to calculate the square of a number.
9. Create a base class **Animal** with a method **speak()**. Create a derived class **Dog** that inherits from **Animal** and overrides the **speak()** method.
10. Create two base classes, **A** and **B**, and a derived class **C** that inherits from both **A** and **B**.
11. Create a class **Circle** with a private attribute **__radius**. Provide methods to set and get the radius value.
12. Create a function **calculate_area()** that calculates the area of different shapes (e.g., circle, square) by passing different objects to it.
13. Create a **Car** class that has a **Engine** class as composition. The **Car** uses the **Engine** for functionality.
14. Create a **Department** class that aggregates multiple **Employee** objects. Calculate the average salary of the department.
15. Create a **Math** class with method overloading for addition to handle both single and double arguments.
16. Create a **Vector** class that supports vector addition using operator overloading.
17. Create a base class **Person** with a constructor, and a derived class **Employee** that extends the constructor using **super()**.
18. Create a class hierarchy with multiple levels of inheritance and show the Method Resolution Order using the **mro()** method.
19. Create a class **Student** with a class variable to keep track of the number of students. Initialize and increment it in the constructor.
20. Create a class **StringBuilder** with methods to add and capitalize strings, and demonstrate method chaining.
21. Create a base class **Base** with a private instance variable, and a derived class **Derived** that attempts to access it.
22. Create classes **A**, **B**, **C**, and **D**, where **A** and **B** inherit from **C**, and **D** inherits from both **A** and **B**. Demonstrate the diamond problem.

23. Create a base class **Shape** with a method **area()**, and a derived class **Triangle** that overloads the **area()** method.
24. Create a base class **Person** with an **__init__** constructor, and a derived class **Employee** with its **__init__** constructor.
25. Create a **Student** class that has a **Profile** class as an instance variable, demonstrating composition in inheritance.