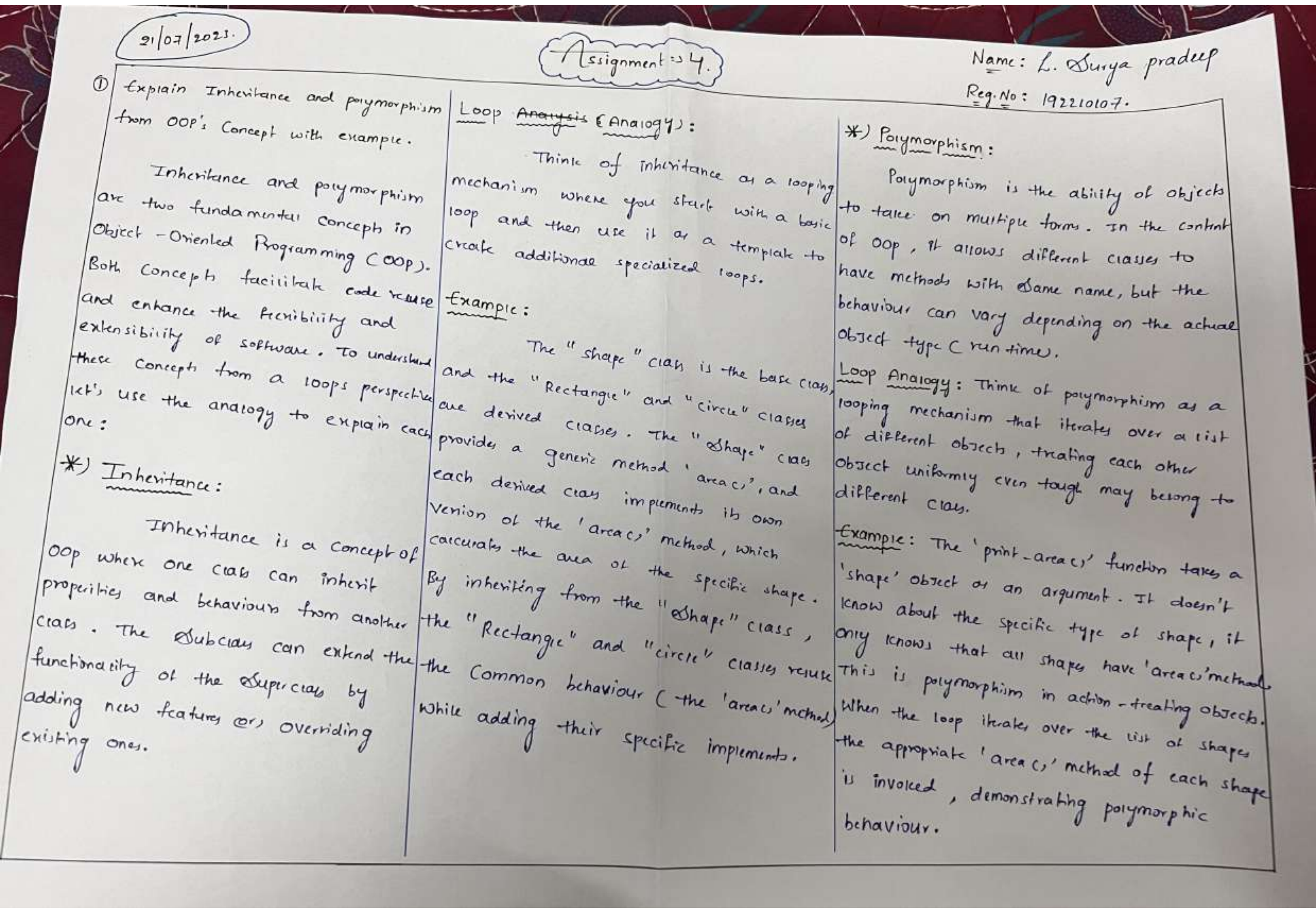

21/07/2023.

Assignment :- 4.

Name: L. Surya pradeep

Reg. No: 192210107.

① Explain Inheritance and polymorphism from OOP's Concept with example.

Inheritance and polymorphism are two fundamental concepts in Object-Oriented Programming (OOP). Both Concepts facilitate code reuse and enhance the flexibility and extensibility of software. To understand these Concepts from a loops perspective let's use the analogy to explain each one:

*) Inheritance:

Inheritance is a concept of oop where one class can inherit properities and behaviours from another class. The Subclass can extend the functionality of the Superclass by adding new features (or) overriding existing ones.

Loop ~~Analysis~~ (Analogy):

Think of inheritance as a looping mechanism where you start with a basic loop and then use it as a template to create additional specialized loops.

Example:

The "shape" class is the base class, and the "Rectangle" and "circle" classes are derived classes. The "Shape" class provides a generic method 'area()', and each derived class implements its own Version of the 'area()' method, which calculates the area of the specific shape. By inheriting from the "Shape" class, the "Rectangle" and "circle" classes reuse the Common behaviour (the 'area()' method) While adding their specific implements.

*) Polymorphism:

Polymorphism is the ability of objects to take on multiple forms. In the content of oop, it allows different classes to have methods with Same name, but the behaviour can vary depending on the actual object type (run time).

Loop Analogy: Think of polymorphism as a looping mechanism that iterates over a list of different objects, treating each other object uniformly even tough may belong to different class.

Example: The 'print_area()' function takes a 'shape' object as an argument. It doesn't know about the specific type of shape, it only knows that all shapes have 'area()' method. This is polymorphism in action - treating objects. When the loop iterates over the list of shapes the appropriate 'area()' method of each shape is invoked, demonstrating polymorphic behaviour.