BankAccount

After creating a class named BankAccount and instiated the instance variables.To make a variable private we use "\_\_" before the variable name.Two methods named deposit and withdraw are declared within the main class.It checks for the balnce and the minimum amount required to withdraw.Object was created to implement the deposit and the withdraw methods.

PasswordManager

built a PasswordManager class with a \_\_init\_\_ method to set the initial password for the private variable \_\_password. The class additionally has a set\_password method for changing the password if the new password is at least 8 characters long. The validate method determines whether or not the entered password matches the stored password.

Shape Hierarchy

The basic class is the Shape class, which has a \_\_init\_\_ function (initialized with pass) and an area method that returns 0. Subclasses are expected to override the area function. Shape has two derived classes: Circle and Rectangle. Each derived class includes its own area method for calculating the area of circles and rectangles based on the particular formula for circles and rectangles. Circle and Rectangle instances are generated. These instances' area methods are invoked with particular arguments to calculate and report the area of a circle and a rectangle.

Animal Sound

The basic class is Animal, which has a talk function that returns a generic sound.The Animal class gives rise to the Dog and Cat classes. Both classes override the talk function to create individual dog ("Bow") and cat ("Meow") sounds. A Cat class instance (cat1) is created. The Cat class's talk function is accessed via cat1.speak(). The result is "Meow" since the Cat class overrides the Animal class's talk function.

Vehicle Description Function

The \_\_init\_\_ constructor function of the Boat and Car classes is used to initialise the colour and model properties. The describe\_vehicle method in each class prints the colour and model information. Creating Instances (obj1):Using obj1 = Car("white", "s90"), an instance of the Car class is created with the colour "white" and model "s90". (obj1.describe\_vehicle()) method call: obj1.describe\_vehicle() invokes the Car class's describe\_vehicle function on the obj1 instance.The output will be the car's description: "white, s90."

Calculator system

The Calculate class is defined as the base class. It has a method named operate that takes two parameters (x and y). Inside the operate method, it assigns the values of x and y to the attributes self.x and self.y. The Adder class is a subclass of the Calculate class. It overrides the operate method of the base class. The overridden operate method performs addition (x + y) and returns the result. The Multiplier class is another subclass of the Calculate class. Similar to Adder, it overrides the operate method of the base class. The overridden operate method performs multiplication (x \* y) and returns the result.

Sensor system

Sensor is an abstract base class (ABC) that was created with the ABC module. Using the @abstractmethod decorator, it declares an abstract method read\_value. This method must be concretely implemented by subclasses. Subclasses of Concrete (TemperatureSensor and PressureSensor): The concrete subclasses of the Sensor abstract class are TemperatureSensor and PressureSensor. They give concrete implementations of the abstract read\_value function. obj1 = TemperatureSensor() creates an instance of the TemperatureSensor class. TemperatureSensor's read\_value method is called, and the value 28 is returned.

Storage Mechanism

code demonstrates the use of abstract classes and abstract methods to define a common interface (Storage). Concrete subclasses (FileStorage and DatabaseStorage) provide specific implementations for saving and reading data. Instances of these subclasses can be created, and their methods can be called based on the common interface defined by the abstract base class.