University of Jijel
Department of Electronics
Electronics of Embedded Systems – Master 1

Practical work N°04

Exercise 01: Identifying attributes and methods in a Car System

You will analyze and identify attributes (properties) and methods (behaviors) for various features of a Car system. Some of the features might involve both attributes and methods, so you'll need to differentiate between them.

Questions :

- Is Car Speed an attribute or a method? Why?
- Is Car Color an attribute or a method? Why?
- Is Start Engine an attribute or a method? Why?
- Is Turn Off Engine an attribute or a method? Why?
- Is Fuel Level an attribute or a method? Why?
- Is Honk Horn an attribute or a method? Why?
- Is Mileage an attribute or a method? Why?
- Is Accelerate an attribute or a method? Why?
- Is Apply Brakes an attribute or a method? Why?
- Is Number of Doors an attribute or a method? Why?
- Is Display Fuel Status an attribute or a method? Why?
- Is GPS Activation an attribute or a method? Why?
- Is Temperature Control an attribute or a method? Why?
- Is Open Door an attribute or a method? Why?
- Is Turn On Lights an attribute or a method? Why?
- Is Check Engine Status an attribute or a method? Why?
- Create a Car class that contains all the specified attributes and methods. Then, instantiate objects of this class (representing different cars), and demonstrate the use of all OOP principles. Provide explanations of where each principle is applied.

Exercise 02: Simulating an embedded System for controlling drones using OOP

In this practical work, you will simulate the control of multiple drones (such as Red, Black, and Yellow) and their components, such as engines, GPS, and sensors, using Object-Oriented Programming (OOP) principles. You will design and implement a system that models drone behaviors using encapsulation, inheritance, polymorphism, and abstraction. Provide explanations of where each principle is applied.

Algorithm: Control system for drones

Step 1: Define Base Class - DroneComponent

1. Initialize:
   - Attributes:
     - name: Name of the component (e.g., engine, GPS, sensor).
     - status: Status of the component (True for active, False for inactive).
2. Methods:
   - turn_on(): Set status to True and print "Component is now ON."
   - turn_off(): Set status to False and print "Component is now OFF."
   - get_status(): Return the current status of the component (either True or False).

Step 2: Define Derived Classes for Specific Components

For each derived class (Engine, GPS, Sensor), follow this structure:

1. Inherit from DroneComponent.
2. Additional Methods for Specific Components:
   - For Engine:
     - start(): Call turn_on() and print "Engine started."
     - stop(): Call turn_off() and print "Engine stopped."

- For GPS:
  - activate(): Call turn_on() and print "GPS activated."
  - deactivate(): Call turn_off() and print "GPS deactivated."
- For Sensor:
  - activate(): Call turn_on() and print "Sensor activated."
  - deactivate(): Call turn_off() and print "Sensor deactivated."

Step 3: Define Class - Drone

1. Attributes:
   - color: Color of the drone (e.g., red, black, yellow).
   - Components:
     - Instantiate Engine, GPS, and Sensor for the drone, passing the drone's color as an identifier.
2. Methods:
   - start_engine(): Call start() on the Engine component.
   - stop_engine(): Call stop() on the Engine component.
   - activate_gps(): Call activate() on the GPS component.
   - deactivate_gps(): Call deactivate() on the GPS component.
   - activate_sensor(): Call activate() on the Sensor component.
   - deactivate_sensor(): Call deactivate() on the Sensor component.
   - check_status(): Print the status of all components of the drone (engine, GPS, sensor).

Step 4: Define Class - DroneManager (Optional, if managing multiple drones is needed)

1. Attributes:
   - drones[]: List of drones managed by the system.
2. Methods:
   - add_drone(drone): Add a drone object to the drones list.
   - manage_all_drones(): For each drone in drones[], call check_status() to display the status of all drones and their components.

Step 5: Main Simulation

1. Create Drones:
   - Instantiate Red Drone, Black Drone, and Yellow Drone using the Drone class.
2. Control Operations for Each Drone:
   - Red Drone:
     - Start engine, activate GPS, and activate sensor.
     - Display the status.
     - Deactivate the GPS and stop the engine.
     - Display the final status.
   - Black Drone:
     - Start engine, activate sensor.
     - Display the status.
     - Deactivate the sensor and stop the engine.
     - Display the final status.
   - Yellow Drone:
     - Start engine, deactivate GPS and sensor.
     - Display the status.
     - Deactivate the GPS and sensor, stop the engine.
     - Display the final status.

Step 6: Final Cleanup

1. Turn off all components:
   - For each drone, ensure that all components (engine, GPS, sensor) are turned off and display their status.