```python
# Required pre-processors for program validity
from logging import critical

import MP2_VehicleSensor
import MP2_Vehicle
import random
import time
from MP2_VehicleSensor import UltrasonicSensor, CameraSensor, RadarSensor, TemperatureSensor,
SpeedSensor
from loadingModule import loadingAnimation

# COMMON ATTRIBUTES OF SENSORS: System Health, Sensor ID, & Sensor Status
# UNIQUE ATTRIBUTES OF TEMP/SPEED SENSORS: Current Temperature & Current Speed

# Displays the current sensor reading array (updated)
def displayArray(sensorArray, status):
    print("\nSensor ID:", end=" ")
    for i in range(len(sensorArray)):
        print(f"| {GREEN}{sensorArray[i].sensorID}", end=f"{RESET} |  ")
    print("\n\nStatus:", end="    ")
    for j in range(len(sensorArray)):
        if(sensorArray[j].system_health < 5):
            print(f"| {GREEN}Inactive", end=f"{RESET} |  ")
        else:
            print(f"| {GREEN}{status}", end=f"{RESET} |  ")
    print("\n\nHealth:", end="    ")
    for k in range(len(sensorArray)):
        if(sensorArray[k].system_health >= 50):
            print(f"|   {GREEN}{sensorArray[k].system_health}%", end=f"{RESET}   |  ")
        elif(sensorArray[k].system_health < 50 and sensorArray[k].system_health >= 5):
            print(f"|   {YELLOW}{sensorArray[k].system_health}%", end=f"{RESET}   |  ")
        else:
            print(f"|   {RED}{sensorArray[k].system_health}%", end=f"{RESET}   |  ")
    print("\n\nTemp:", end="      ")
    for a in range(len(sensorArray)):
        if(a == 3):
            if(sensorArray[a].system_health < 5):
                print(f"|   {RED}ISSUE", end=f"{RESET}   |  ")
            else:
                if(sensorArray[a].currentTemp < 100):
                    print(f"|   {GREEN}{str(sensorArray[a].currentTemp)} F", end=f"{RESET}   |  ")
                else:
                    print(f"|   {RED}{str(sensorArray[a].currentTemp)} F", end=f"{RESET}   |  ")
        else:
            print("|   N/A", end="   |  ")
    print("\n\nSpeed:", end="     ")
    for b in range(len(sensorArray)):
        if(b == 4):
            if(sensorArray[b].system_health < 5):
                print(f"|   {RED}ISSUE", end=f"{RESET}   |  ")
            else:
                if(sensorArray[b].currentSpeed < 80):
                    print(f"| {GREEN}{sensorArray[b].currentSpeed} MP/H", end=f"{RESET} |  ")
                else:
                    print(f"| {RED}{sensorArray[b].currentSpeed} MP/H", end=f"{RESET} |  ")
        else:
            print("|   N/A", end="   |  ")
    print()

# Updates the current sensor reading array to make live changes to Health, Temperature, Speed, & Status
 logistically
def updateArray(sensorArray, count):
    if(not check and sensorArray[4].currentSpeed >= 20):
        sensorArray[4].setSpeed(0)
    if(not check and sensorArray[3].currentTemp < 90):
        sensorArray[3].changeTemperature(90)
    for e in range(len(sensorArray)):
        if(not check):
```

```
            sensorArray[e].fixSystemHealth()
        if(sensorArray[e].system_health >= 5):
            flip = random.randint(0, 1)
            if(flip == 0):
                sensorArray[e].decreaseSystemHealth(random.randint(0, 1))
        else:
            count += 1
            if(count >= 5):
                return count
    flip = random.randint(0, 1)
    if(flip == 0):
        if(sensorArray[3].currentTemp < 180):
            sensorArray[3].changeTemperature(sensorArray[3].currentTemp + random.randint(0, 2))
        else:
            sensorArray[3].changeTemperature(sensorArray[3].currentTemp / 2)
        if(sensorArray[4].currentSpeed < 165):
            sensorArray[4].setSpeed(sensorArray[4].currentSpeed + random.randint(0, 3))
        else:
            sensorArray[4].setSpeed(sensorArray[4].currentSpeed - random.randint(10, 25))
    else:
        if(sensorArray[3].currentTemp >= 15):
            krand = random.randint(0, 1)
            if(krand == 1):
                sensorArray[3].changeTemperature(sensorArray[3].currentTemp - random.randint(0, 2))
        else:
            sensorArray[3].changeTemperature(sensorArray[3].currentTemp + random.randint(60, 100))
        if(sensorArray[4].currentSpeed >= 2):
            sensorArray[4].setSpeed(sensorArray[4].currentSpeed - random.randint(0, 3))
        else:
            sensorArray[4].setSpeed(sensorArray[4].currentSpeed + random.randint(5, 15))
    return 0

# ANSI escape sequences for terminal cursor control
def move_cursor_up(lines):
    print(f"\033[{lines}A", end='')

def move_cursor_down(lines):
    print(f"\033[{lines}B", end='')

# Gives a delay gap between each iteration of sensor reading
def delayFunc():
    time.sleep(1)

# Capitalizes the first letter of each detail inputted about vehicle
def fixFormat(userStr):
    newStr = list(userStr.capitalize())
    for r in range(1, len(newStr)):
        if(newStr[r - 1] == " "):
            newStr[r] = newStr[r].upper()
    return "".join(newStr)

# Loading animation for visual effect
def introOutroAnimation(status):
    for seconds in range(0, 3):
        if(status == "Active"):
            print(f"Starting Vehicle to Motion{'.' * (seconds + 1)}   ", end = " ")
        else:
            print(f"Stopping Vehicle from Motion{'.' * (seconds + 1)}   ", end=" ")
        time.sleep(1)



# Variables
sensorArray = [UltrasonicSensor(), CameraSensor(), RadarSensor(), TemperatureSensor(), SpeedSensor()]
GREEN = "\033[32m"
YELLOW = "\033[33m"
RED = "\033[31m"
BRIGHT_MAGENTA = "\033[95m"
```

```
135 BG_YELLOW = "\033[43m"
136 BLUE = "\033[34m"
137 BOLD = "\033[1m"
138 RESET = "\033[0m"
139 status = "Active"
140 check = False
141 criticalCount = 0
142 moderateCount = 0
143 count = 0
144
145 # Introduction
146 print("\t\t\t\tWelcome to the sub-program of the well known 'Vehicle Monitor Application'\n")
147 print("In this program, you will add one vehicle of your choice to do a sensor reading where you will
    see the sensor data live (as it is changing).")
148 print("If all the sensor's health percent were to reach below 5%, the vehicle would stop immediately
    and give you the next steps.")
149 print("The vehicle would be in motion and active while the sensor reading is in progress.\n")
150
151 # Data inputting
152 make = fixFormat(input("Enter the make of the vehicle: "))
153 model = fixFormat(input("Enter the model of the vehicle: "))
154 electric = fixFormat(input(f"Is the {make} {model} electric (True/False): "))
155 while(electric != "True" and electric != "False"):
156     electric = fixFormat(input("Make sure to enter either True or False: "))
157 color = fixFormat(input(f"What color is the {make} {model}: "))
158 while(color.isdigit()):
159     color = fixFormat(input("Please enter a valid color. A color cannot be expressed in numbers: "))
160 while True:
161     try:
162         year = int(input(f"What year is the {make} {model}: "))
163         while(year < 1880 or year > 2025):
164             year = int(input(f"Please enter a valid year for {make} {model}: "))
165         break
166     except ValueError:
167         print("You entered string instead of numbers. Please enter the year again.\n")
168         continue
169
170 # Loading animation for visual effect
171 loadingAnimation()
172
173 # Vehicle detail printed in an arrowhead format
174 print(f"\n\nVEHICLE DATA:\n\t\t\t[Vehicle Make: {make}]\n\n\t\t\t\t\t\t[Vehicle Model: {model}]\n\n\t\
    t\t\t\t\t\t\t[Electric: {electric}]\n\n\t\t\t\t\t\t[Vehicle Color: {color}]\n\n\t\t\t[Vehicle Year:
    {year}]\n\n")
175
176 # Gather the time limit for sensor reading
177 while True:
178     try:
179         num = int(input("How long do you want the sensor reading to measure for in seconds (>0): "))
180         while(num == 0):
181             num = int(input("Input must not be 0. Only seconds greater than 0 are accepted: "))
182         break
183     except ValueError:
184         print("Cannot input string instead of numbers. Try again\n")
185
186 introOutroAnimation(status)
187
188 # Iterations of displaying and updating sensor data for num seconds
189 print(f"\n\n{BLUE}{year} {make} {model} (before complete){RESET}:")
190 print("\t   UltraSonic     Camera     Radar     Temperature     Speed")
191 while(num > 0):
192     print(f"\r{BOLD}{BRIGHT_MAGENTA}{BG_YELLOW}{int(num / 60)} minute(s) and {num % 60} second(s)
    remain{RESET}", end=" ")
193     updateArray(sensorArray, count)
194     count = updateArray(sensorArray, count)
195     if(count >= 5):
196         allSensorsDown = displayArray(sensorArray, status)
197         break
```

```
    else:
        displayArray(sensorArray, status)
        check = True
        delayFunc()
        num -= 1
        if(num != 0):
            move_cursor_up(10)

# End statistics w/ after summary
if(count < 5):
    print("\n")
    status = "IDLE"
    introOutroAnimation(status)
    print("\n")
    sensorArray[4].setSpeed(0)
    if(sensorArray[3].currentTemp >= 100):
        sensorArray[3].changeTemperature(sensorArray[3].currentTemp - random.randint(30, 60))
    print(f"\n\n{BLUE}{year} {make} {model} (after complete){RESET}:")
    displayArray(sensorArray, status)
    for u in range(len(sensorArray)):
        if(sensorArray[u].system_health < 5):
            criticalCount += 1
        elif(sensorArray[u].system_health >= 5 and sensorArray[u].system_health < 50):
            moderateCount += 1
    print(f"\n\nAll sensor reading tests are complete for the {year} {make} {model}.")
    print(f"\n* {criticalCount} sensor(s) in a critical condition (% < 5 [RED]).\n* {moderateCount}
sensor(s) in a moderate condition (% >= 5 & < 50 [YELLOW]).\n* {5 - (criticalCount + moderateCount)}
sensor(s) in a good condition (% >= 50 [GREEN])")
elif(count >= 5):
    move_cursor_down(10)
    print(f"\nThe system health of all sensor's are critical. Replace the sensor's with new ones and
then re-run for further sensor reading and safe utilization of the {year} {make} {model}.")
```