ACIT 2515 – Object Oriented Programming - Lab 6
Python Class Built-In Methods and Template Method Pattern

Instructor	Mike Mulder (mmulder10@bcit.ca) Also available on Discord.
Total Marks	10
Due Date	End of Class

Goals

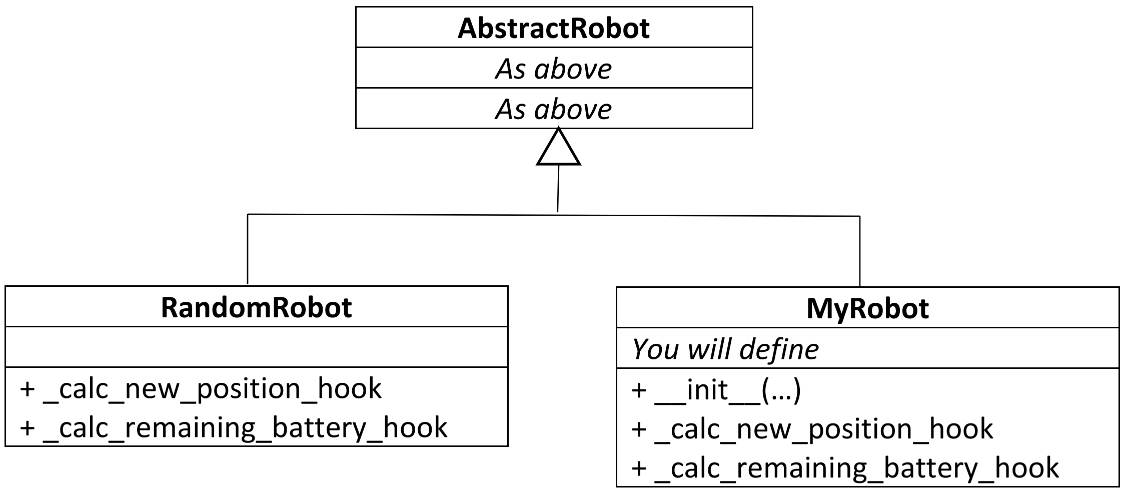
- Implement the Template Method Design Pattern.
- Override one of the built-in methods in Python classes.

Overview

You are given the following AbstractRobot class that acts as a template for different autonomous robot implementations (i.e., the Template Method design pattern). It has a template method called *move* that calculates a new position and battery level of the robot through two abstract two methods: _calc_new_position_hook() and _calc_remaining_battery_hook(). Child classes of AbstractRobot must override these two methods with their own model of the robot's movement and battery life.

AbstractRobot
- _name : string - _battery_percent : float - _min_x : int - _max_x : int - _min_y : int - _max_y : int - _curr_x : int - _curr_y : int - _prev_x : int - _prev_y : int
+ __init__(name : string, min_x : int, min_y : int, max_x : int, max_y : int) + get_name() : string + get_curr_position() : tuple + get_prev_position() : tuple + get_battery_level() : string + move()   ***Note: this is the template method*** *+ _calc_new_position_hook()*   ***Note: this is a hook method*** *+ _calc_remaining_battery_hook()*   ***Note: this is a hook method*** + _validate_curr_position() + _validate_battery_level()

In this lab, you will be creating two robot classes, RandomRobot and MyRobot, that are derived from AbstractRobot and override and implement the hook methods.



AbstractRobot
As above
As above
RandomRobot
+ _calc_new_position_hook
+ _calc_remaining_battery_hook
MyRobot
You will define
+ __init__(…)
+ _calc_new_position_hook
+ _calc_remaining_battery_hook

You are also give the Python script report_main.py that will create an instance of your robot class and then call the *move* method repeatedly until the battery is dead. It then plots both the path and battery of the robot over course of the simulation. It uses a 3rd party library called matplotlib to plot this data.

Make sure your _calc_remaining_battery_hook method is implemented correctly otherwise you might get into an infinite loop if the battery never gets to 0% remaining.

Instructions

Setup

- Create a new project in PyCharm called Lab 6
- Download and copy in the following files from the Week 6 course content on D2L:
  - robot_main.py
  - abstract_robot.py
- Install the matplotlib 3rd party library in your PyCharm project:
  - Windows: File -> Settings -> Project -> Project Interpreter -> +
    - Search for matplotlib and install
  - Mac: PyCharm -> Preferrences -> Project –P Project Interpreter -> +
    - Search for matplotlib and install
  - Pip: pip install matplotlib

<u>RandomRobot</u>

- Create a new class named RandomRobot in a file called random_robot.py
- The RandomRobot should be a child of AbstractRobot (remember to import AbstractRobot)
- In RandomRobot, override the _calc_new_position_hook abstract hook method from AbstractRobot:
    - Move the current x position (_curr_x) by a random value between -2 and 2 (hint: use random.randint(min, max) from the built-in random module)
    - Make sure the current x position (_curr_x) stays within the minimum and maximum x values (_min_x and _max_x)
    - Move the current y position by a different random value between -2 and 2
    - Make sure the current y position (_curr_y) stays within the minimum and maximum y values (_min_y and _max_y)
- In RandomRobot, override the _calc_remaining_battery_hook abstract hook method from AbstractRobot:
    - Decrement the battery percent (_battery_percent) by a random value between 0.0 and 1.0 (hint: use random.random() from the built-in random module)
    - Make sure it does not decrement to a value lower than 0.0
- In report_main.py, uncomment the line that creates an instance of RandomRobot.
- Run the report_main.py script and see the output. You should see a plot with the path of your robot on the grid in black, and dots overlaying that path with the battery level (green for full, yellow for normal and red for low).
- If you run report_main.py again, the robot should have followed a different path before running out of battery power.

<u>MyRobot</u>

- Create a new class named MyRobot in a file called my_robot.py
- The RandomRobot should be a child of AbstractRobot (remember to import AbstractRobot)
- Think of your own algorithms for moving your robot's position and decrementing it's battery level. Your algorithm should have at least one configurable parameter.
- Override the constructor from AbstractRobot, adding your new configurable parameter(s).
- Override the _calc_new_position_hook method, implementing your own algorithm for robot position.
- Override the _calc_remaining_battery_hook method, implementing your own algorithm for decrementing the battery percentage.
- In report_main.py, update the line of code that creates an instance of RandomRobot to create an instance of MyRobot instead.

- Run the report_main.py script and see the output. You should see a plot with the path of your robot on the grid in black, and dots overlaying that path with the battery level (green for full, yellow for normal and red for low).
- You may need to debug your algorithms if the output isn't as you expect.

<u>Built-in Method (__str__)</u>

- In AbstractRobot, override the built-in __str__ method.
- It should return a string in the following format:

  Robot **Mike**: Position (**1**, **5**) with **LOW** battery level.

  Where Mike is the Robot name, 1/5 are the current x/y coordinates and LOW is the current battery level.
- In report_main.py, replace the print outs of the position and battery level with a print out of your MyRobot object (i.e., print(my_robot)) which will call your implementation of __str__

Demonstration

- Demonstrate your two robot classes to your instructor before you leave
- Address any comments
- If you address the comments, you will get 10/10
- Submit your my_robot.py code to the dropbox