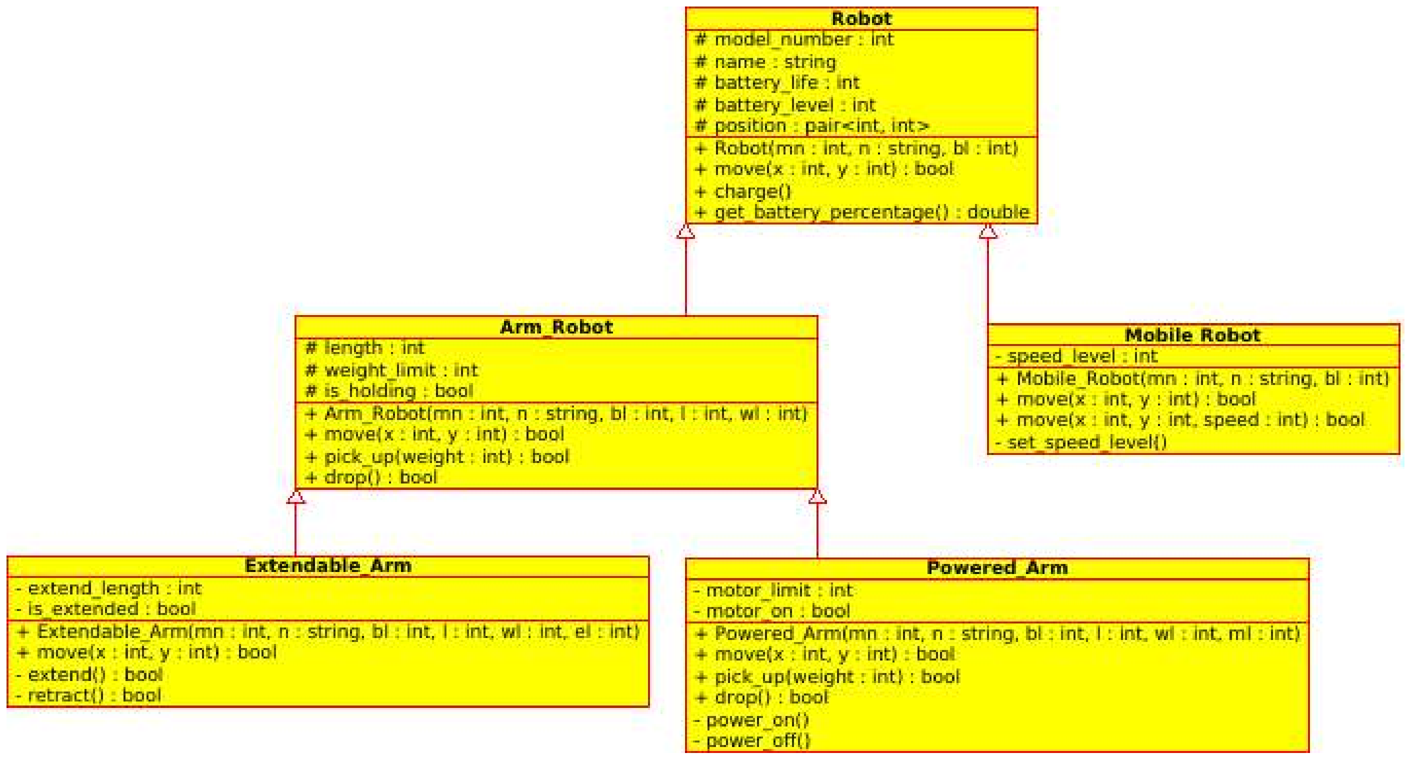
CSE 1325-001 Homework #6 – Inheritance

In this homework assignment, you will be creating multiple classes based off the following UML diagram.


Robot
# model_number : int
# name : string
# battery_life : int
# battery_level : int
# position : pair<int, int>
+ Robot(mn : int, n : string, bl : int)
+ move(x : int, y : int) : bool
+ charge()
+ get_battery_percentage() : double
Arm_Robot
# length : int
# weight_limit : int
# is_holding : bool
+ Arm_Robot(mn : int, n : string, bl : int, l : int, wl : int)
+ move(x : int, y : int) : bool
+ pick_up(weight : int) : bool
+ drop() : bool
Mobile Robot
- speed_level : int
+ Mobile_Robot(mn : int, n : string, bl : int)
+ move(x : int, y : int) : bool
+ move(x : int, y : int, speed : int) : bool
- set_speed_level()
Extendable_Arm
- extend_length : int
- is_extended : bool
+ Extendable_Arm(mn : int, n : string, bl : int, l : int, wl : int, el : int)
+ move(x : int, y : int) : bool
- extend() : bool
- retract() : bool
Powered_Arm
- motor_limit : int
- motor_on : bool
+ Powered_Arm(mn : int, n : string, bl : int, l : int, wl : int, ml : int)
+ move(x : int, y : int) : bool
+ pick_up(weight : int) : bool
+ drop() : bool
- power_on()
- power_off()

Part 1: Robot

For this part, you will create a robot class. The robot has a model number, a name, a battery life (an int representing how many "units" the battery will last for), and a position ((x,y) coordinate).

The constructor assigns all values to the input parameters, sets the position to (0,0), and sets the battery level to max. The move method changes the coordinate location of the robot. It also reduces the battery level by 1 unit for each distance of one it moves. The distance it moves can be calculated with the classic distance formula. Since battery life is represented as an int, you just round up the distance traveled. This method returns true if the move was successful and false if not, such as if the battery would die by doing the move. charge restores the battery level to full. get_battery_percentage returns the current battery level as a percent.

Part 2: Arm_Robot

Arm_Robot is derived from Robot. Think of an Arm robot as a Sawyer Robot from Rethink Robotics. It is a stationary robot, but the arm can move. Position and move now pertain to the end of the arm's position.

Three new variables are introduced. The length is how long the arm is. The weight limit is how heavy an object the arm can pick up. is_holding is whether or not the arm is holding an object.

The constructor assigns all values to the input parameters, sets the position to (0,0), sets the battery level to max, and sets the is_holding to false. Move acts the same as the parent class, but now each 1 distance takes an extra battery unit if it is holding an object (1 if moving, 2 if moving and holding an object). It can also not move to a location if the desired location is past the length of the arm. Consider the base of the arm is at (0,0). Pick_up will tell the robot to hold the object if it is under the weight limit. Pick_up also takes 1 unit of battery. It cannot pick up an object if it already holding an object. This method returns true if it was successful and false if not. Drop will tell the robot to drop the object. Drop also takes 1 unit of battery. It cannot drop an object if it not holding an object. This method returns true if the move was successful and false if not.

Part 3: Extendable_Arm

Extendable_Arm is derived from Arm_Robot. This robot can extend its arm further out than a regular Arm_Robot. If it can't reach an object, it will extend its arm further.

Two new variables are introduced. The extend_length is the distance that the arm can extend. Therefore, the total distance the robot can reach is now the length + extend_length. is_extended is whether or not the arm is extended.

The constructor assigns all values to the input parameters, set the position to (0,0), sets the battery level to max, sets is_holding to false, and sets is_extended to false. Move acts the same as Arm_Robot, but now each 1 distance takes an extra battery unit if the arm is extended (1 if moving; 2 if moving and holding an object; 2 if moving and extended; 3 if moving, holding and object, and extended). If the position is too far for the arm to reach, but can with the arm extended, the move() method will call extend(). If the arm can reach the position with the arm not extended, move() will call retract. Both extend and retract each take 1 battery unit. Extend can't extend if already extended, and retract can't retract if already retracted.

Part 4: Powered_Arm

Powered_Arm is derived from Arm_Robot. This robot can turn on a motor that can pick up heavier objects than an Arm_Robot.

Two new variables are introduced. The motor_limit is how much the arm can pick up when the motor is on. This this different than extend_length. extend_length was an additional amount to the arm length. motor_limit is the total amount the arm can lift when the motor is on, not an additional amount. Also, there is motor_on, which states whether or not the motor is on or not.

The constructor assigns all values to the input parameters, set the position to (0,0), sets the battery level to max, sets is_holding to false, and sets motor_on to false. Move acts the same as Arm_Robot, but now each 2 distance takes an extra battery unit if the motor is on (1 if moving; 2 if moving and holding an object; 3 if moving and the motor is on; 4 if moving, holding and object, and motor is on). Pick_up() will turn the motor on if the object is too heavy for the arm to lift on its own. Drop() will turn off the motor if the motor is on when the object is dropped. Turning on and off the motor does not cost battery.

Part 5: Mobile Robot

Mobile_Robot is derived from Robot. This robot can move around at three different speed levels.

One new variable is introduced. The speed level is how fast the robot will move. If it is a 1, it will move slowly. If it is a 2, it will move at an average speed. If it is a 3, it will move at a fast speed.

The constructor assigns all values to the input parameters, set the position to (0,0), and sets speed level to 1. Move with 3 parameters will set the speed level, then move the robot. Move with 2 parameters will move the robot at whatever the current speed level is at. Slow movement is 1 battery unit for every 1 distance. Average movement is 2 battery units for every 1 distance. Fast movement is 3 battery units for every 1 distance.

Part 6: Main

In your main function, you will create one robot of each type, each with a battery life of 100. Then the robots will do the following (Each must print out a statement whether the robot was able to complete the task or not. Also it must say which robot it was):

- Robot will move around until the battery drains, charge up, then move around till it drains again.
- Mobile robot will move around at three different speeds. It will also move once without not changing the speed.
- Arm robot will move, pick up an object that it can pick up, move it, and drop it. Then the Arm robot will attempt to pick up an object that is too heavy. Then the robot will attempt to move beyond the length of the arm.
- Extendable_Arm will move to pick up an object that it can pick up while retracted, move it, and drop it. Then the arm will move to pick up an object that it can only pick up while extended, move it to a spot that it can reach only while extended, and drop it. Then the robot will to pick up an object that it can pick up while retracted, move it to a spot it can only reach while extended, and drop it. Next, the robot will try to reach an object it can't reach even, while extended.
- Powered Arm will move to pick up an object that it can pick up without the motor, move it, and drop it. Then it will move to pick up an object that it can only pick up with the motor, move it, and drop it. Lastly it will move and attempt to pick up an object that is too heavy.

Bonus (10pts)

For the Mobile Robot, change the speed_level to be an object of an Enumerator Class instead of an int. Be sure to update the UML diagram to reflect this.

Deliverables

You will submit your code and screenshots via Blackboard. You will upload a zip file, named "abc1234_HW6zip", which contains 1 folder (2 if you did the bonus)

- full_credit
    - abc1234_Robot.h and abc1234_Robot.cpp
    - abc1234_Arm_Robot.h and abc1234_Arm_Robot.cpp
    - abc1234_Extendable_Arm.h and abc1234_Extendable_Arm.cpp
    - abc1234_Powered_Arm.h and abc1234_Powered_Arm.cpp
    - abc1234_Mobile_Robot.h and abc1234_Mobile_Robot.cpp
    - abc1234_main.cpp
    - makefile
    - abc1234_main.png (or multiple if multiple screenshots were taken). These screenshots will be picture of your code running in terminal.
    - Instructions for compiling and running your code (either in comments in blackboard or in a README file)
- bonus_1
    - abc1234_HW6_Class_Diagram.xmi
    - abc1234_Robot.h and abc1234_Robot.cpp
    - abc1234_Arm_Robot.h and abc1234_Arm_Robot.cpp
    - abc1234_Extendable_Arm.h and abc1234_Extendable_Arm.cpp
    - abc1234_Powered_Arm.h and abc1234_Powered_Arm.cpp
    - abc1234_Mobile_Robot.h and abc1234_Mobile_Robot.cpp
    - abc1234_main.cpp
    - makefile
    - abc1234_main.png (or multiple if multiple screenshots were taken). These screenshots will be picture of your code running in terminal.
    - Instructions for compiling and running your code (either in comments in blackboard or in a README file)

Full credit files named incorrectly result in a loss of 5 points each.