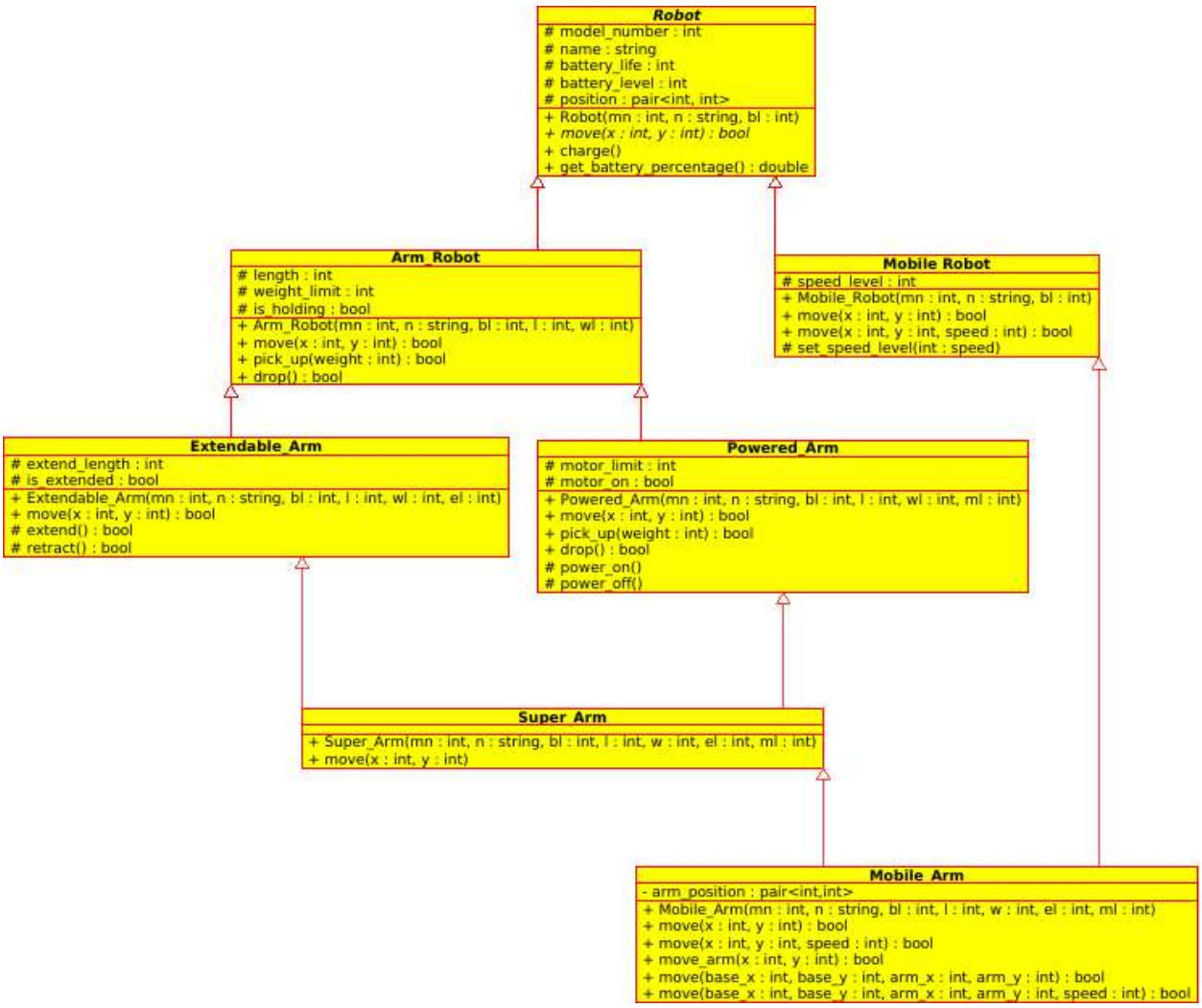
CSE 1325-001 Homework #7 – Multiple Inheritance and Polymorphism

In this homework assignment, you will be extending your Homework 6. You may use your existing HW6 solution, or the supplied HW6 sample solution (coming soon).

See the below UML Diagram



Part 1: Changes from HW6 UML that's not a new class.

- Robot
  - Robot is an Abstract Class, meaning there is at least one Abstract Function. An Abstract Function is a function that is declared but not defined. In C++, an abstract function is a pure virtual function.
  - move is now a pure virtual function.
- Mobile_Robot
  - Typo from HW6 has been corrected

    - speed_level and set_speed_level() are now protected
- Extendable_Arm
    - extended_length, is_extended, extend(), and retract() are now protected
- Powered_Arm
    - motor_limit, motor_on, power_on(), power_off() are now protected

Part 2: Super_Arm

Super_Arm inherits from Extendable_Arm and Powered_Arm. There are no variables.

The Constructor calls the base class' constructer. Move will now take into account if it is extended and if the motor is on. So now it will take 1 battery unit to move 1 distance, 2 if moving and holding an object, 2 if moving and extended, 3 if moving, holding an object, and extended, 4 if moving, holding an object, and motor is on, 5 if moving, holding an object, motor is on, and arm is extended.

Part 3: Mobile_Arm

Mobile_Arm inherits from Super_Arm and Mobile_Robot. There is one new variable to help differentiate the different between the base's position (the mobile robot position) and the arm's position (the end of the arm). position is the base's position. arm_position is the arm's position.

The Constructor calls the base class' constructer. There are now 5 move functions

- move(int, int) moves the base to the new coordinates at the current speed level. It just calls Mobile_Robot's move(int, int) method.
- move(int, int, int) move the base to the new coordinates at the new speed level. It just calls Mobile_Robot's move(int, int, int) method.
- move_arm(int, int) move the arm to the new coordinates. It just calls Super_Arm's move method.
- move(int, int, int, int) moves both the base and the arm to the new coordinates at the current speed level. The UML shows which input variables go to which. The arm's reach must be calculated from what would be the bases new position. The base's position should not update if the arm cannot reach its new position
- move(int, int, int, int, int) moves both the base and the arm to the new coordinates at the new speed level. The UML show which input variables go to which.

Part 4: Main without Polymorphism

Create a main method that creates a Super_Arm and Mobile_Arm.

For the Super_Arm, you must show the following four scenarios:

- Move to pick up an object that the arm does not need to be extended, pick up an object that the arm can pick up without the motor, move the object to a spot that the arm does not need to be extended, drop the object.
- Move to pick up an object that the arm does not need to be extended, pick up an object that the arm can pick up without the motor, move the object to a spot that the arm does need to be extended, drop the object.

- Move to pick up an object that the arm does not need to be extended, pick up an object that the arm can only pick up with the motor, move the object to a spot that the arm does not need to be extended, drop the object.
- Move to pick up an object that the arm does not need to be extended, pick up an object that the arm can only pick up with the motor, move the object to a spot that the arm does need to be extended, drop the object.

For Mobile_Arm, you must show a success and unsuccessful case for each move method. One of the unsuccessful cases must be because the arm cannot reach from the base's new position.

Part 5: Main with Polymorphism

You will make a second main file (abc1234_main_two.cpp). In this file, you will create a main function that contains a list of Robots. Make sure at least one of each robot is in the list. Each robot must be able to move to a new position (move both base and arm if a mobile robot), pick up an object (if it is an arm), move to a new position (move both base and arm if a mobile robot), drop the object (if it is an arm). If it's an Arm_Robot, you also must try to move to a position that is too far away and pick up an object that is too heavy. If it's an Extendable Arm, you must also move to a position that it has to extend to and try to pick up an object that is too heavy. If it's a Powered Arm, you must try to pick up an object that requires the motor.

Feel free to add the "string type" field to each class to differentiate between the objects like we did in class.

For compiling this main, you have to use the same make file to compile from part 4. (Hint: you can type more than just make in terminal. Think back to HW 3.)

Bonus (5pts):

Replace the string type field with an enumeration telling which class is which.

Deliverables:

You will submit your code and screenshots via Blackboard. You will upload a zip file, named "abc1234_HW7zip", 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_Super_Arm.h and abc1234_Super_Arm.cpp
  - abc1234_Mobile_Arm.h and abc1234_Mobile_Arm.cpp
  - abc1234_main1.cpp
  - abc1234_main2.cpp
  - makefile

    - abc1234_main1.png (or multiple if multiple screenshots were taken). These screenshots will be picture of your code running in terminal.
    - abc1234_main2.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_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_Super_Arm.h and abc1234_Super_Arm.cpp
    - abc1234_Mobile_Arm.h and abc1234_Mobile_Arm.cpp
    - abc1234_main2.cpp
    - makefile
    - abc1234_main.png (or multiple if multiple screenshots were taken). These screenshots will be picture of your code running in terminal.
    - abc1234_main2.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)
    - Any extra files needed for the enumeration

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