Lab0

Design a set of Java classes to emulate a Robot Controller. The Robot Controller sends an encoded integer to the Robot called an opcode. See the technical details below. Use Object Design principles to design the Controller and the opcode class.

See this link for an Overview of Object Oriented design:

http://javarevisited.blogspot.com/2012/03/10-object-oriented-design-principles.html

The class is designed to set the appropriate commands for the Robot to execute. The lab consists of using the Robot Controller to send the commands to rotate the Robot using each of the four rotations.

Opcode (encoded integer)

The Robot executes the bit commands in an opcode. An opcode is encoded integer, with this encoding:

Bit 0:   0=off, 1=on
Bit 1:   1=slow
Bit 2:   1=fast
Bit 3:   1=clockwise
Bit 4:   1=counter clockwise
Bit 5:   1=rotate 90 degrees* (mutually exclusive)
Bit 6:   1=rotate 180 degrees* (mutually exclusive)
Bit 7:   1=rotate 270 degrees* (mutually exclusive)
Bit 8:   1=rotate 360 degrees* (mutually exclusive)
* Bits 5 thru 8: only one can be on

Robot Interface

Constructor: Initializes opcode integer to zero
Example: Robot robot = new Robot();

Set: Turns on one of the bits 0 thru 8 (returns true if successful, false otherwise)
Example: robot.Set(1<<3); // turns on bit 3
robot.Set(8); // turns on bit 3
These examples are based on the concept of "magic numbers". Magic numbers don't exercise the principals of object oriented programming techniques. The Set function uses the encoded integer to set commands for the robot.

Execute: Executes the integer opcode and resets to zero based on these rules:

* Bit 1 must be on (set to 1)
* Either the slow or fast bit (but not both) must be on (set to 1)
* Either the clockwise or counter clockwise (but not both) must be on (set to 1)
* Only one of the bits 5 thru 8 must be on (set to 1)
* If any of these bitcodes is not on, the opcode command is invalid
  Example: robot.Execute();