CS101- Algorithms and Programming I

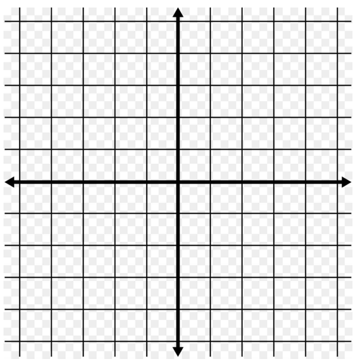
Lab 07

Lab Objectives: Classes and Objects

- ❏ For all labs in CS 101, your solutions must conform to these CS101 style guidelines (rules!)
- ❏ Remember to include **javadoc comments** for each class and method.
- ❏ Create a Lab07 workspace (i.e. the folder H:\private\cs101\lab07).

---

a. Create a new project Lab07. Our objects will be in the 10x10 coordinate system where the base for robots is in the very middle with coordinates (0,0) in the application. See the following.



Note: The words “scout robot” and “robot” will be used interchangeably.
Create a class `ScoutRobot` with the following properties and the methods:

- **Constant Data Members:**
  - `VELOCITY`: velocity of the scout robots (2 units/time)
  - `CONSUMED_ENERGY_PER_UNIT`: energy consumed by a robot per unit (5)

- **Static Data Members:**
  - `scoutCounter:`initial value is 100 and counts the number of scout robots created.
- **Instance Data Members:**
  - `robotID:` stores the id of the scout robot (example: “Scout-100”) (String)
  - `xCoordinate:` stores the x-coordinate of scout robot in the coordinate system (int)
  - `yCoordinate:` stores the y-coordinate of scout robot in the coordinate system (int)

    - `direction:` stores the direction the scout robot is facing at the moment. (North = 0, East = 1, South = 2, West = 3) (int)
    - `maxBattery:` stores the maximum amount of battery the scout robot can have (double)
    - `batteryLeft:` stores the amount of battery left of the scout robot (double)
    - `condition:` stores the condition of the scout robot that indicates whether it needs helps from a mechanic or not (boolean)
- **Methods:**
    - **Constructor**
        - Takes x-coordinate, y-coordinate, and maximum amount battery that a scout robot can have (all integers)
        - Initializes the variables using appropriate set methods.

    - **Accessor Methods**
        - get methods for the instance data members.

    - **Mutator Methods**
        - `setRobotID` - private method that sets the scout robot id using the name ("Scout") plus the current scout robot counter (example: Scout-105 if 5 scout robots have been created)
        - Complete the other set methods for the rest of the instance data members
    - **Other Methods**
        - `goToLocation():` instance method that takes an integer x-coordinate and an integer y-coordinate as parameters. Returns true if the scout robot is able to go to specified point, false otherwise. Update `condition` variable accordingly. However, while going to some point in the coordinate system, the scout robots cannot cross over the base, which is on (0, 0). In some cases you may have pass around the base if needed. Also keep in mind that the objects are on 10x0 coordinate system. You should validate the parameters. Prints the minimal path the scout robots goes during the move. At the end, print the time taken and battery left for the scout robot. See the sample run to fully understand what you are supposed to do. Please consider all cases and implement accordingly.
        - `callMechanic():` instance method that takes a Mechanic as a parameter and if the scout robot does not have enough battery for a move asked by the base, the scout robot calls the mechanic. Print a statement that shows a mechanic has been called and call the `fixRobot()` method of Mechanic class via the parameter Mechanic object to simulate the process of the mechanic's fixing the robot.
        - `toString():` returns a String representation of a scout robot. See sample output for format details.

        - `compareRobots():` instance method that takes a ScoutRobot as a parameter, and compares the ScoutRobots. If the target object is closer to the base, returns 1, further distance returns -1, same distance returns 0.
        - You may write helper methods if necessary.

b. Create another class `Mechanic` inside the same project with the following properties and the methods:

- **Static Data Members:**
    - `mechanicCounter:` initial value is 10 and counts the number of mechanics created.
- **Instance Data Members:**
    - `mechanicID:` stores the id of the mechanic (example: "Mechanic-10") (String)
    - `xCoordinate:` stores the x-coordinate of scout robot in the coordinate system (int)
    - `yCoordinate:` stores the y-coordinate of scout robot in the coordinate system (int)
- **Methods:**
    - **Constructor**
        - Takes no parameters
        - Initializes instance data members using set methods appropriately

    - **Accessor Methods**
        - get methods for the instance data members.

    - **Mutator Methods**
        - `setMechanicID` - private method that sets the scout robot id using the name ("Mechanic") plus the current mechanic counter (example: Mechanic-15 if 5 scout robots have been created)
        - Complete the other set methods for the rest of the instance data members

    - **Other Methods**
        - `teleportation():` instance method that takes integer x-coordinate and y-coordinates as parameters. As the name of the method implies, the mechanic teleports to the specified point using set methods. Also keep in mind that the objects are on 10x0 coordinate system. You should validate the parameters.
        - `fixRobot():` instance method that takes ScoutRobot as a parameter. Checks the condition of the scout robot, if it is no

good, mechanic teleports to the current point of the scout robot; fills up the battery and changes the condition of the robot. It, then, teleports back to the base. Either way, print a statement that explains the current situation. See the sample run.

- `toString()`: returns a String representation of a scout robot. See sample output for format details.

c. Create an application class RobotApp which does the following:
    - *Note that* in the main method you are acting as the base, giving commands etc.
    - Create 2 ScoutRobot and 1 Mechanic objects.
    - Display the objects.
    - First ScoutRobot has the following points to discover:
      (2,3) -> (-2,0) -> (3,0) -> (0,5)
    - Second ScoutRobot has the following points to discover:
      (3,-4) -> (0,3) -> (0,-4) -> (1,2)
    - At some point, scout robots' conditions are going to be no good, which means they will have no enough battery left to do next move. So, make proper method calls to call for a mechanic.
    - At the end, compare the distances of robots to the base. Print the proper statement.
    - See the sample run in the `output.txt` file for printing formats.