# CS102 - Algorithms and Programming II
# Lab Programming Assignment 4
# Fall 2016

**ATTENTION:**
- Feel free to ask questions on Moodle on the Lab Assignment Forum.
- Compress all of the Java program source files (.java) files into a single zip file.
- The name of the zip file should follow the below convention:
  **CS102_SecX_Asgn4_YourSurname_YourName.zip**
- Replace the variables “YourSurname” and “YourName” with your actual surname and name and X with your Section id (1, 2 or 3).
- Upload the above zip file to Moodle by the deadline before the lab (if not significant points will be taken off). You will get a chance to update and improve your solution by consulting to the TA during the lab. You will resubmit your code once you demo your work to the TA.

**GRADING WARNING:**
- Please read the grading criteria provided on Moodle.

**Q1 [100 p.]** In this lab, you will implement a simple text-based game where you are an air force commander and you are defending your border from enemy vehicles trying to pass through. To implement this game, you will need the following class hierarchy:

**Destructible -** An interface that has methods with the following signatures:
- `boolean isDestroyed()`
- `void takeDamage(double damage)`

**Difficulty -** An interface that has methods with the following signatures:
- `void setDifficulty(int difficulty)`
- `int getDifficulty()`

For the difficulty values, you should use 1 for Easy, 2 for Medium and 3 for Hard.

**Movable -** An interface that has methods with the following signatures:
- `void move()` (note that objects will only be able to move in the -y direction)
- `Point getLocation()` (Use the Point class in `java.awt.*`)

**EnemyVehicle -** An abstract class that implements **Movable** and **Destructible**. This class should also have the following methods:

- `EnemyVehicle(int minX, int maxX, int y)`: this constructor should initialize the position of this object, such that its x coordinate will be between `minX` and `maxX`, and its y coordinate will be `y`.
- `int getDistanceToBorder()`
- `abstract String getType()`
- All methods from **Movable** and **Destructible** interfaces.

This class should also have the following instance variable:

- `final int BASE_SPEED = 3` : this variable should be used in implementing the `move()` method

**Helicopter -** A class that extends **EnemyVehicle**. The difference between Helicopter and EnemyVehicle is that Helicopter objects should increase their speed after each move. This class should also have the following methods:

- `Helicopter(int minX, int maxX, int y)`
- `void move()` : this method should be overriden since we want Helicopter objects to move faster after each turn.
- `String toString()`
- Any method that needs to be implemented from the superclass.

**Tank -** A class that extends **EnemyVehicle**. The difference between the Tank and EnemyVehicle is that Tank objects should take less damage from attacks. This class should also have the following methods:

- `Tank(int minX, int maxX, int y)`
- `void takeDamage(double damage)` : this method should be overriden since we want Tank objects to take less damage from attacks.
- `String toString()`
- Any method that needs to be implemented from the superclass.

**Player -** A class that will be used to attack EnemyVehicle objects. This class should have the following methods:

- A default constructor.
- `void attack(int x, int y, ArrayList<EnemyVehicle> enemies)` : this method should check if there are any EnemyVehicle

objects in a circular area whose center is `(x, y)` and radius is `BOMB_RADIUS`, and if so, call their `takeDamage()` methods.

This class should also have the following instance variables:

- **`final double`** `BOMB_RADIUS = 2.0`
- **`final double`** `DAMAGE = 50.0`

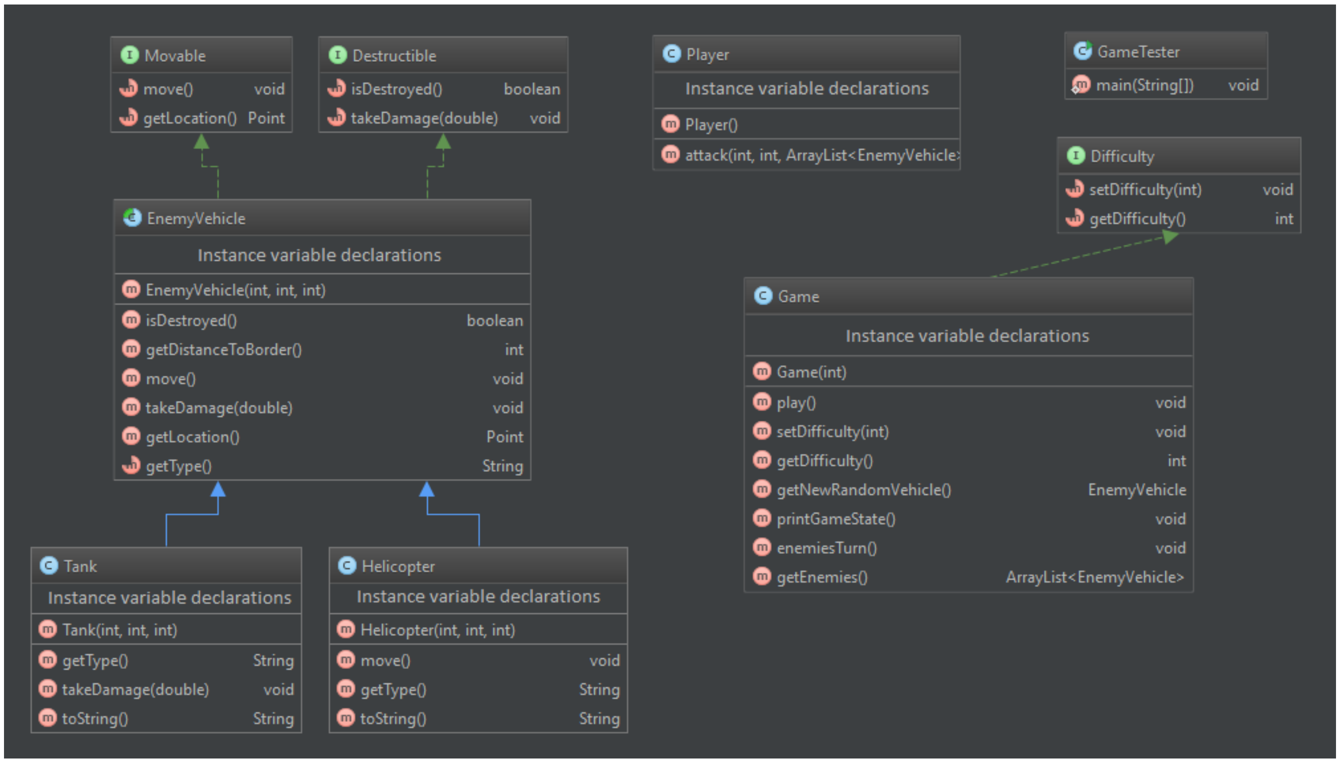
**Game -** A class that implements **Difficulty**. This class should have the following methods:

- `Game(`**`int`**` difficulty)` : this constructor should set the difficulty and initialize any instance variables you may have. This method should also create the Player and a number of EnemyVehicle objects (2 for Easy, 4 for Medium, 6 for Hard). The initial x coordinates of the EnemyVehicle objects should be between 0-5 for Easy, 0-10 for Medium and 0-15 for Hard. Their y coordinates should be `STARTING_DISTANCE`.
- All methods from the **Difficulty** interface.
- **`void`** `play()` : this method is where the game will be played. It should ask for input from the user (x and y coordinates to bomb) and deal damage accordingly. Next, it should check if any EnemyVehicle objects were destroyed, and if so, replace them with a new EnemyVehicle object and increase the player’s score. After that, it should make every enemy move and again replace them with a new EnemyVehicle if they crossed the border (i.e. if their y ≤ 0). This should continue in a loop until the number vehicles passing the border reaches `MAX_NO_OF_PASSING_ENEMIES`. You should also print the current state of the game in the beginning of each turn. After the loop ends, this method should print the final score of the player.
- `ArrayList<EnemyVehicle> getEnemies()`
- The following private methods to help you with the `play()` method:
    - `EnemyVehicle getNewRandomVehicle()`: this method should return a new Tank or Helicopter object with equal probability.
    - **`void`** `enemiesTurn()`: this method should check each enemy if they are destroyed or have passed the border after moving and replace them with new vehicles.
    - **`void`** `printGameState()`

Finally, you need to implement a **GameTester** class to test the classes you have implemented. In the main method, you should ask the user for a difficulty level, create a Game object accordingly, and start the game.

You are given a class diagram with this assignment that should help you with the class hierarchy. Note that the instance variables and visibility modifiers are not denoted in this diagram. If you need it, the legend of this class diagram can be found at https://www.jetbrains.com/help/idea/2016.2/symbols.html

You are also given a class named GamePlotter, which, given a Game object, plots the vehicles and their locations on the console. You can use this class to help you visualize the current state of the game. This is optional.



Movable
move() void
getLocation() Point
Destructible
isDestroyed() boolean
takeDamage(double) void
Player
Instance variable declarations
Player()
attack(int, int, ArrayList<EnemyVehicle
GameTester
main(String[]) void
Difficulty
setDifficulty(int) void
getDifficulty() int
EnemyVehicle
Instance variable declarations
EnemyVehicle(int, int, int)
isDestroyed() boolean
getDistanceToBorder() int
move() void
takeDamage(double) void
getLocation() Point
getType() String
Game
Instance variable declarations
Game(int)
play() void
setDifficulty(int) void
getDifficulty() int
getNewRandomVehicle() EnemyVehicle
printGameState() void
enemiesTurn() void
getEnemies() ArrayList<EnemyVehicle>
Tank
Instance variable declarations
Tank(int, int, int)
getType() String
takeDamage(double) void
toString() String
Helicopter
Instance variable declarations
Helicopter(int, int, int)
move() void
getType() String
toString() String