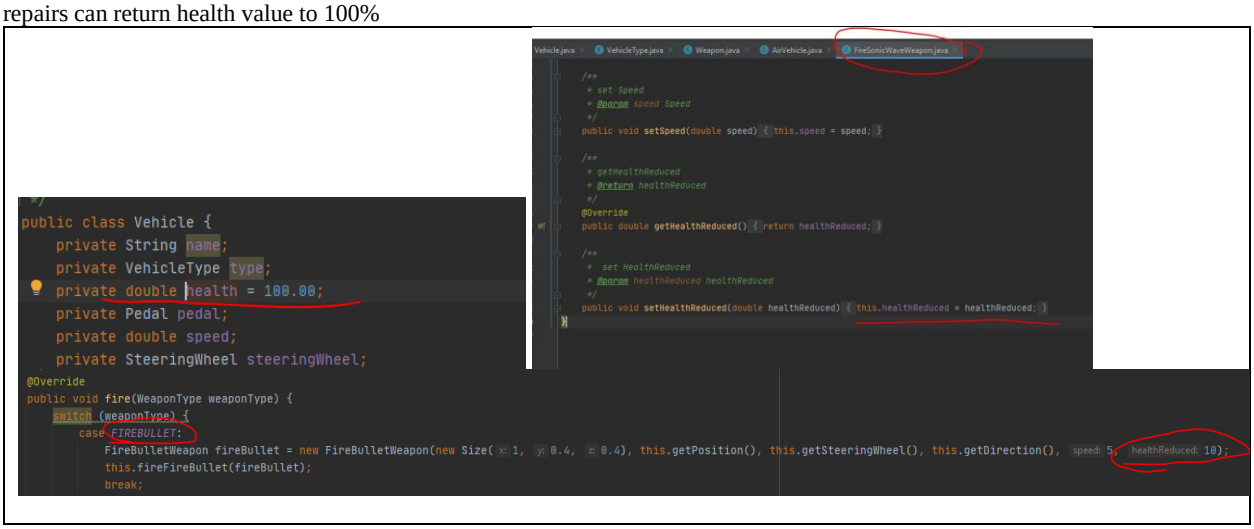
M03 - Weapons by Interface

YiChen_2020_10_25

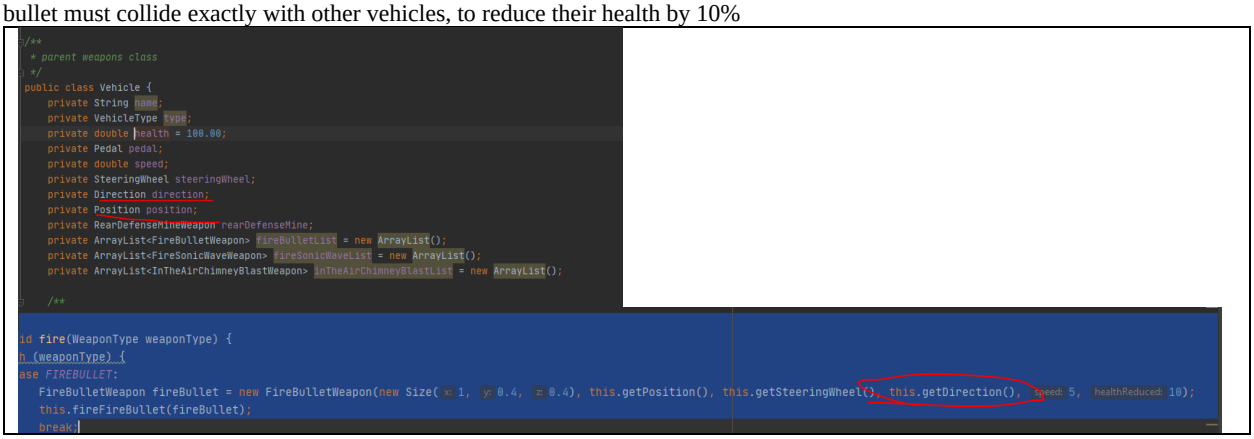
Weapons by Interface

Use Interfaces to create the following actions that a vehicle may take:

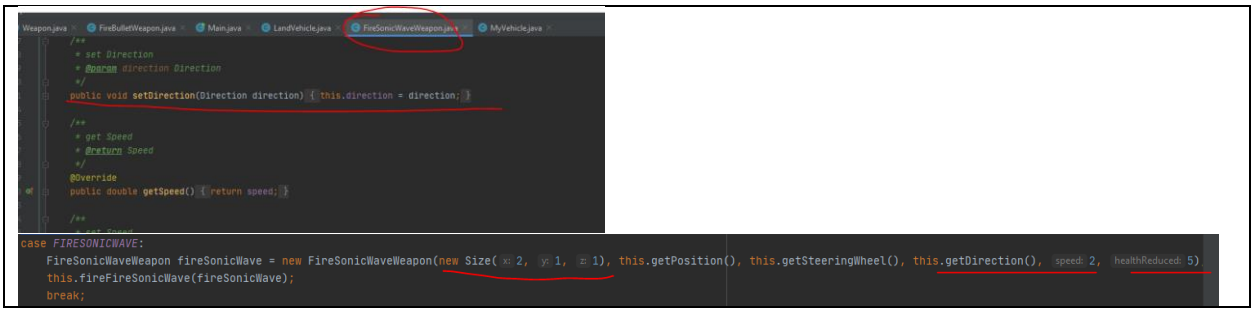
- **Health**:
  - object has a health value of 100%
  - health value can be decremented
  - repairs can return health value to 100%



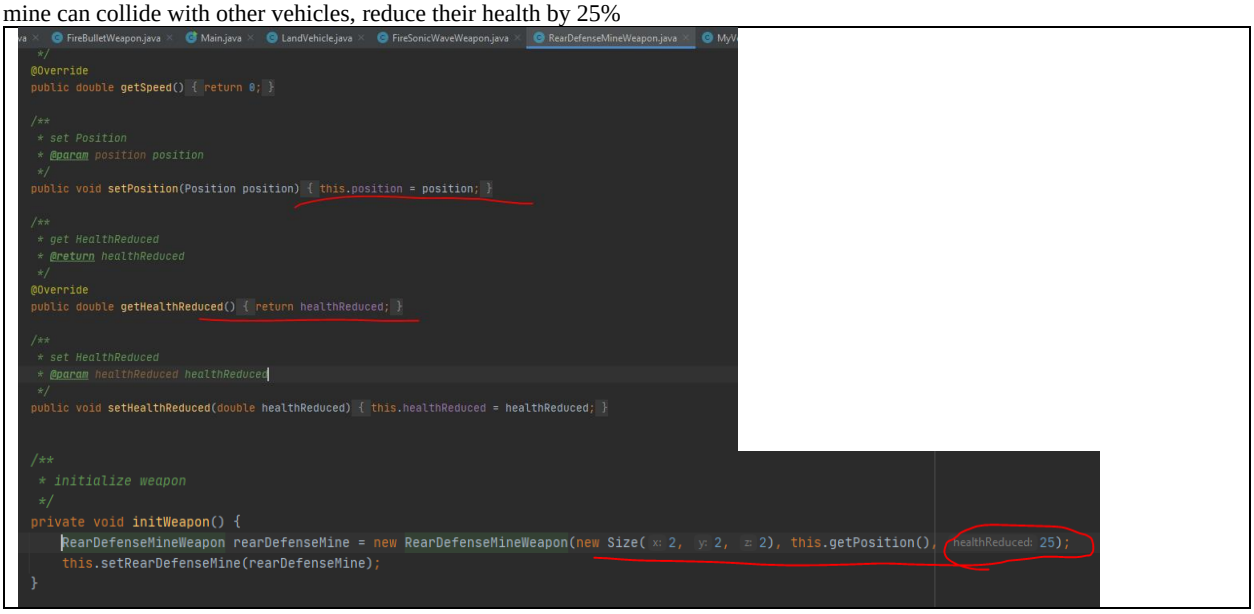
- **Fire bullet**:
  - bullet goes in same direction as vehicle
  - bullet travels fast and is small (you pick speed, size)
  - bullet must collide exactly with other vehicles, to reduce their health by 10%



- **Fire sonic wave:**
  - sonic goes in same direction as vehicle
  - sonic wave travels slow, and is large (you pick speed, size)
  - sonic wave can collide with other vehicles, reduce their health by 5%

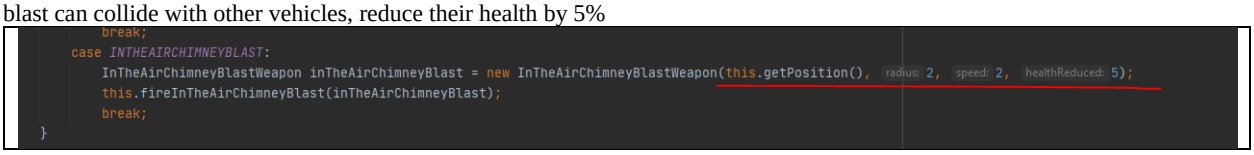


- **Rear defense mine**
  - mine is stationary, at the place vehicle left it.
  - mine is large (you pick size)
  - mine can collide with other vehicles, reduce their health by 25%



- **In-the-Air chimney blast**

  - blows a area directly above with a straight up blast
  - blast starts in this vehicles location, going up wit a radius that you choose.
  - blast can collide with other vehicles, reduce their health by 5%



```java
/**
 * class InTheAirChimneyBlastWeapon
 *blows a area directly above with a straight up blast
 * blast starts in this vehicles location, going up wit a radius that you choose.
 * blast can collide with other vehicles, reduce their health by 5%
 */
public class InTheAirChimneyBlastWeapon extends Weapon implements WeaponInterface {
    private Position position;
    private double radius;
    private double speed;
    private double healthReduced;

    /**
     * InTheAirChimneyBlastWeapon
     * @param position position
     * @param radius radius
     * @param speed speed
     * @param healthReduced healthReduced
     */
    public InTheAirChimneyBlastWeapon(Position position, double radius, double speed, double healthReduced) {
        this.position = position;
        this.radius = radius;
        this.speed = speed;
        this.healthReduced = healthReduced;
    }
```

- Weapons:
  - Connect one vehicle to the keyboard and/or mouse for firing weapons (or get weapons to fire at random times)
  - process a weapons strike on a vehicle ... decrements health, changes directions, whatever you want.

  Write a Java program that creates 3 land vehicles and 3 air vehicles, and a loop performing:

```java
package com.weapons;

/**
 * class Weapon collect all weapons
 */
public class Weapon {
    private String name;
    private Vehicle vehicle;

    /**
     * getName
     * @return name
     */
    public String getName() { return name; }

    /**
     * setName
     * @param name name
     */
    public void setName(String name) { this.name = name; }

    /**
     * getVehicle
     * @return weapons
     */
    public Vehicle getVehicle() { return this.vehicle; }

    /**
     * set Vehicle
     * @param vehicle weapons
     */
    public void setVehicle(Vehicle vehicle) { this.vehicle = vehicle; }
}
```

- Give the vehicle random settings for speed, direction, and all other settings (wings up, ...whatever)

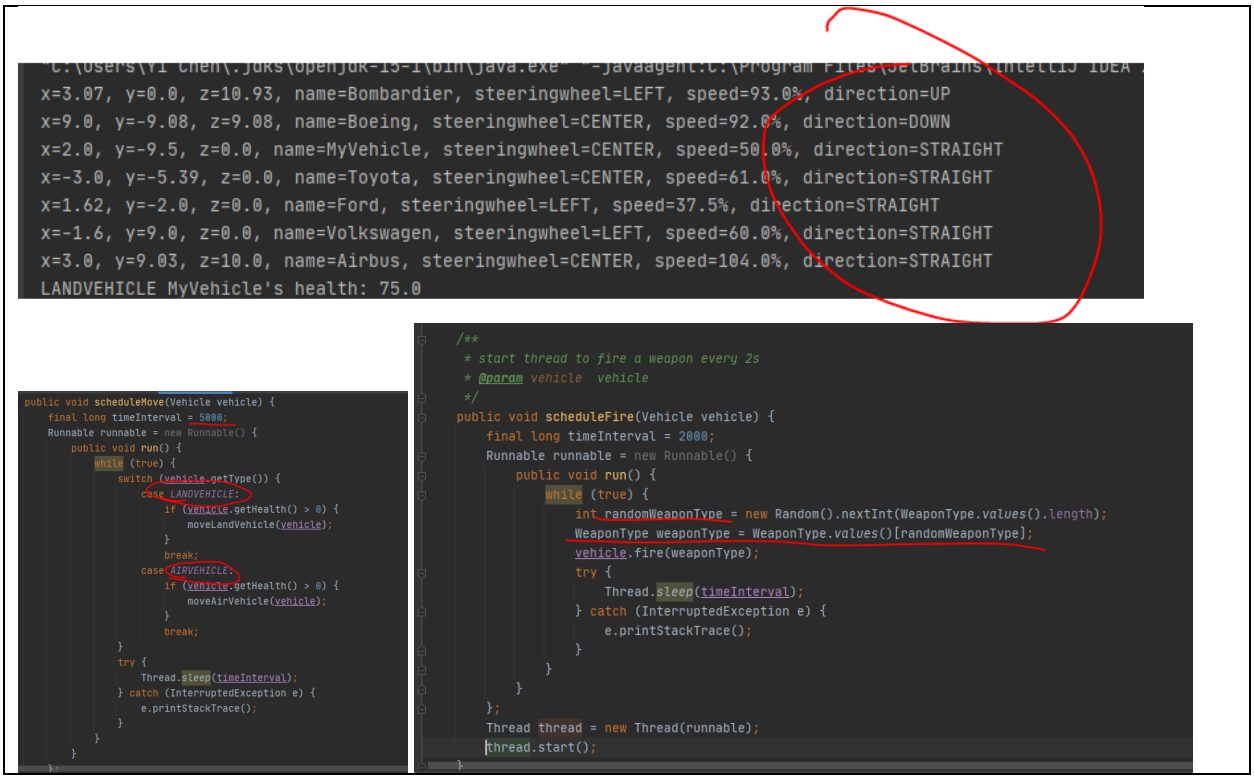
```java
/**
 * LandVehicle
 * @param type  LandVehicle
 * @param name  model
 */

public LandVehicle(VehicleType type, String name) {
    super(type, name);
    this.setSpeed(levelSpeed);
    Random random = new Random();
    double randomX = random.nextInt( bound: 21) - 10;
    double randomY = random.nextInt( bound: 21) - 10;
    this.setPosition(new Position(randomX, randomY,  z: 0));
    this.setSteeringWheel(SteeringWheel.CENTER);
    this.setDirection(Direction.STRAIGHT);
    this.initWeapon();
}

/**
 * initialize weapon
 */
private void initWeapon() {
    RearDefenseMineWeapon rearDefenseMine = new RearDefenseMineWeapon(new Size( x: 2,
```

- The vehicle is updated every second, and it changes position depending on the steering wheel, or propeller, or whatever.

```
x=3.07, y=0.0, z=10.93, name=Bombardier, steeringwheel=LEFT, speed=93.0%, direction=UP
x=9.0, y=-9.08, z=9.08, name=Boeing, steeringwheel=CENTER, speed=92.0%, direction=DOWN
x=2.0, y=-9.5, z=0.0, name=MyVehicle, steeringwheel=CENTER, speed=50.0%, direction=STRAIGHT
x=-3.0, y=-5.39, z=0.0, name=Toyota, steeringwheel=CENTER, speed=61.0%, direction=STRAIGHT
x=1.62, y=-2.0, z=0.0, name=Ford, steeringwheel=LEFT, speed=37.5%, direction=STRAIGHT
x=-1.6, y=9.0, z=0.0, name=Volkswagen, steeringwheel=LEFT, speed=60.0%, direction=STRAIGHT
x=3.0, y=9.03, z=10.0, name=Airbus, steeringwheel=CENTER, speed=104.0%, direction=STRAIGHT
LANDVEHICLE MyVehicle's health: 75.0
```

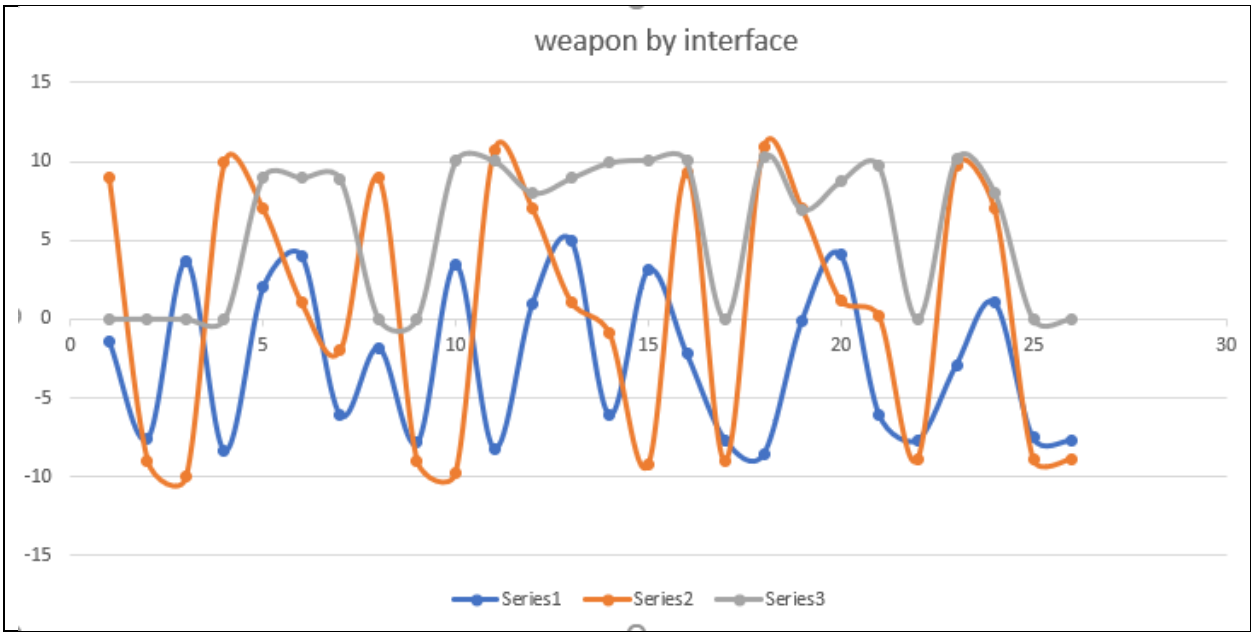
```java
/**
 * start a thread to move weapons every 5s
 * @param vehicle
 */
public void scheduleMove(Vehicle vehicle) {
    final long timeInterval = 5000;
    Runnable runnable = new Runnable() {
        public void run() {
            while (true) {
                switch (vehicle.getType()) {
                    case LANDVEHICLE:
                        if (vehicle.getHealth() > 0) {
                            moveLandVehicle(vehicle);
                        }
                        break;
                    case AIRVEHICLE:
                        if (vehicle.getHealth() > 0) {
                            moveAirVehicle(vehicle);
                        }
                        break;
                }
                try {
                    Thread.sleep(timeInterval);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }
    };
    Thread thread = new Thread(runnable);
    thread.start();
}
```

- Every 10 seconds the controls are randomly nudged in one direction or another, to simulate someone at the controls. (display the change in text output)...***this includes random new firing of weapons above***.



- The vehicle rebounds from outer boundaries so it stays within your viewing area.

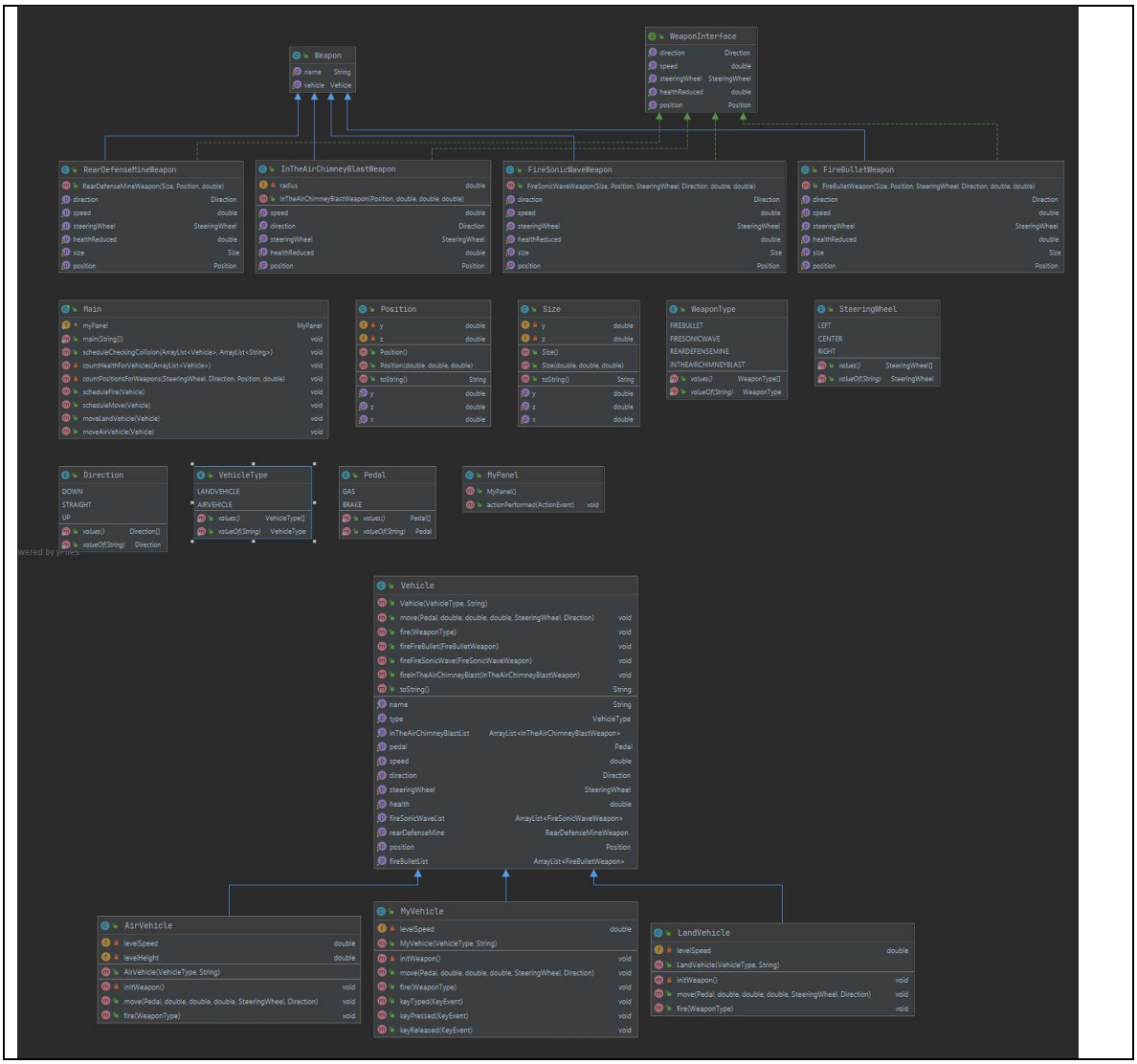
- Text output shows the location each second, in x,y,z format, capable of being loaded into a spreadsheet.



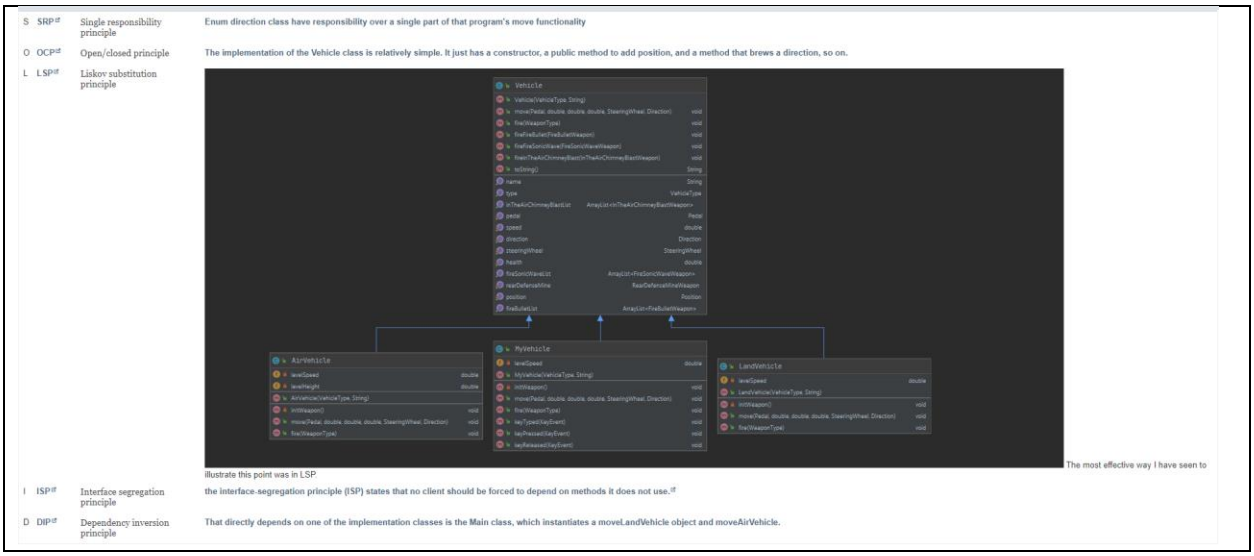
- The physics of the movement need not be perfect, but left and right would change the x, y coordinates, and height would change the z coordinate (or y coordinate?) in the proper general direction.

```
x=3.08, y=-9.23, z=10.11, name=Volkswagen, steeringwheel=CENTER, speed=34.33%, direction=STRAIGHT
x=-2.18, y=9.33, z=10.04, name=Ford, steeringwheel=CENTER, speed=33.9%, direction=STRAIGHT
x=-7.67, y=-9.0, z=0.0, name=MyVehicle, steeringwheel=RIGHT, speed=18.49%, direction=STRAIGHT
x=-8.55, y=10.92, z=10.25, name=Toyota, steeringwheel=LEFT, speed=59.61%, direction=STRAIGHT
x=-0.07, y=7.6, z=6.94, name=Airbus, steeringwheel=LEFT, speed=107.91%, direction=DOWN
x=4.06, y=1.19, z=8.74, name=Boeing, steeringwheel=LEFT, speed=98.82%, direction=STRAIGHT
x=-6.08, y=0.17, z=9.71, name=Bombardier, steeringwheel=CENTER, speed=109.08%, direction=STRAIGHT
```

Install ObjectAid in your eclipse to produce a class diagram of your homework, showing the classes used, and their relationship.

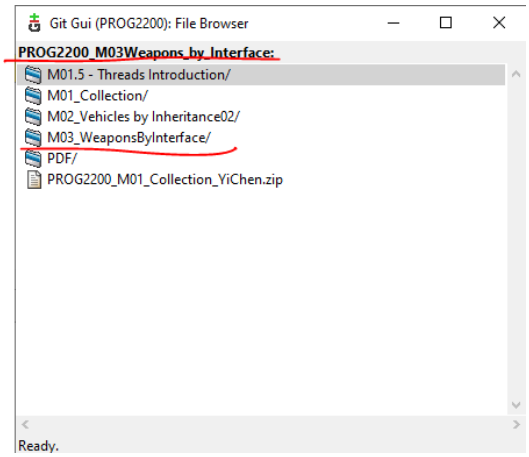


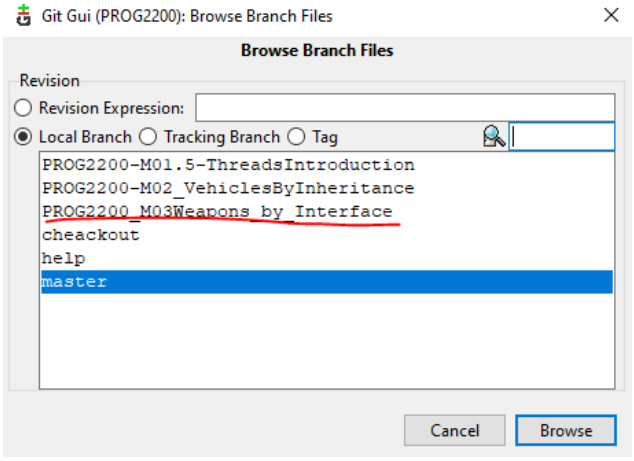
Copy the table below into your package-level JavaDoc documentation, and PDF. For every concept, describe how you implemented, or could implement it.



Submit Artifacts for marking:

- You will be asked to explain your code in class during a code review. Be prepared to demonstrate your code, and answer questions.
- your code into the git server, using a branch labeled PROG2200-Mxx (where xx is the module number)





- Submit a simple PDF with code and running output (no TOC, paragraphs, ...)
- Submit a movie (MP4) of you explaining your code and running your code.