M02 - Vehicles by Inheritance

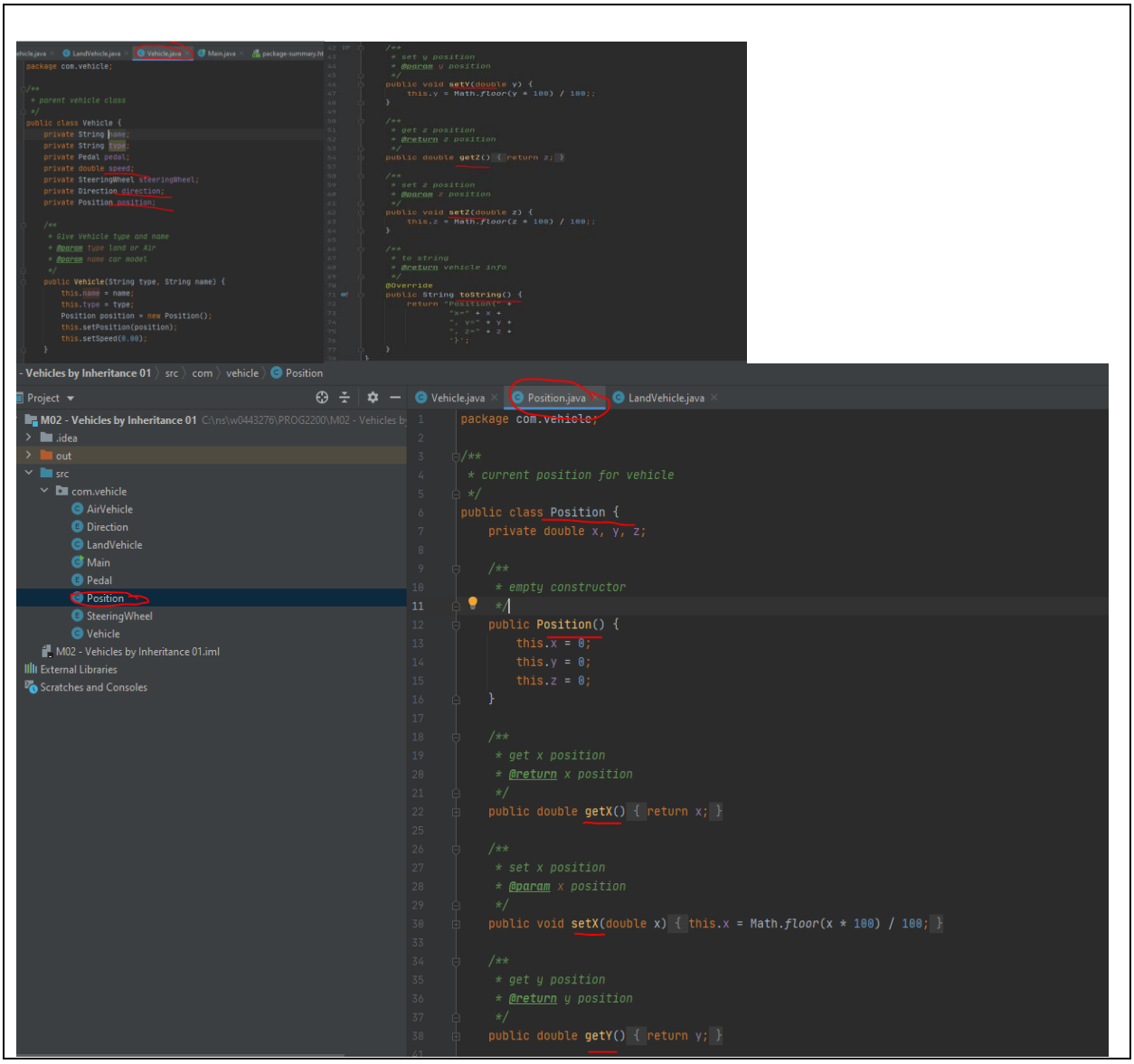
YiChen_2020_10_15

3D Land and Air Vehicles (Rendered in text, no 3D Display)

Use Classes to make a Class Hierarchy: Write the following class hierarchy in Java (you may alter these controls, provided the vehicle has proper movability):

Vehicle class:

has position, speed, direction



Land Vehicle Class (sub-class of Vehicle)

has Gas Pedal (% pressed)

has brake Pedal (% pressed),

has steering wheel (left, center, right)

```java
package com.vehicle;

public class LandVehicle extends Vehicle {
    private double levelSpeed = 50; // 50%

    public LandVehicle(String type, String name) {
        super(type, name);
        this.setSpeed(levelSpeed);
    }

    /**
     * move vehicle according to its speed, steering wheel and direction
     * @param pedal
     * @param gasPedalSpeed
     * @param brakePedalSpeed
     * @param propellerSpeed
     * @param steeringWheel
     * @param direction
     */
    @Override
    public void move(Pedal pedal, double gasPedalSpeed, double brakePedalSpeed, double propellerSpeed, SteeringWheel steeringWheel, Direction direction) {
        if (pedal.equals(Pedal.GAS)) {
            this.setSpeed(this.getSpeed() * (1 + gasPedalSpeed / 100));
        } else if (pedal.equals(Pedal.BRAKE)) {
            double currentSpeed = this.getSpeed() * (1 - brakePedalSpeed / 100);
            currentSpeed = currentSpeed > 0 ? currentSpeed : 0;
            this.setSpeed(currentSpeed);
        }
        this.setPedal(pedal);
        this.setDirection(direction);
        this.setSteeringWheel(steeringWheel);

        if (steeringWheel.equals(SteeringWheel.LEFT)) {
            double x = this.getPosition().getX() + (this.getSpeed() / 100) * -1;
            this.getPosition().setX(x);
            this.getPosition().setY(y);
        } else if (steeringWheel.equals(SteeringWheel.RIGHT)) {
            double x = this.getPosition().getX() + (this.getSpeed() / 100) * 1;
            this.getPosition().setX(x);
        }
    }
}
```

Air Vehicle Class (sub-class of Vehicle)

has height

has propeller speed; ... use % speed (50% => level speed, more means faster, less means slower)

```java
public class AirVehicle extends Vehicle {
    private double levelSpeed = 50;  // 50%
    private double levelHeight = 10; // 10km

    /**
     * AirVehicle  type  name
     * @param type type
     * @param name name
     */

    public AirVehicle(String type, String name) {
        super(type, name);
        this.getPosition().setZ(levelHeight);
        this.setSpeed(levelSpeed);
    }

    /**
     * move vehicle according to its speed, steering wheel and direction
     * @param pedal GAS BRAKE
     * @param gasPedalSpeed gasPedalSpeed
     * @param brakePedalSpeed brakePedalSpeed
     * @param propellerSpeed propellerSpeed
     * @param steeringWheel LEFT CENTER RIGHT
     * @param direction  direction
     */
    @Override
    public void move(Pedal pedal, double gasPedalSpeed, double brakePedalSpeed, double propellerSpeed, SteeringWheel steeringWheel, Direction di
        this.setSpeed(this.getSpeed() * (1 + propellerSpeed / 100));
        this.setDirection(direction);
        this.setSteeringWheel(steeringWheel);

        if (steeringWheel.equals(SteeringWheel.LEFT)) {
            double x = this.getPosition().getX() + (this.getSpeed() / 100) * -1;
```

has steering wheel for wings with straight/left/right and pull steering wheel back/forth for down/up direction

```java
        if (steeringWheel.equals(SteeringWheel.LEFT)) {
            double x = this.getPosition().getX() + (this.getSpeed() / 100) * -1;
            if (x < -10 || x > 10) {
                x = 0;
            }
            this.getPosition().setX(x);
        } else if (steeringWheel.equals(SteeringWheel.CENTER)) {
            double y = this.getPosition().getY() + (this.getSpeed() / 100) * 1;
            if (y > 100) {
                y = 0;
            }
            this.getPosition().setY(y);
        } else if (steeringWheel.equals(SteeringWheel.RIGHT)) {
            double x = this.getPosition().getX() + (this.getSpeed() / 100) * 1;
            if (x < -10 || x > 10) {
                x = 0;
            }
            this.getPosition().setX(x);
        }

        if (direction.equals(Direction.UP)) {
            double z = this.getPosition().getZ() + (this.getSpeed() / 100) * 1;
            if (z < 5 || z > 15) {
                z = levelHeight;
            }
            this.getPosition().setZ(z);
        } else if (direction.equals(Direction.DOWN)) {
            double z = this.getPosition().getZ() + (this.getSpeed() / 100) * -1;
            if (z < 5 || z > 15) {
```

```
x=0.52, y=0.0, z=9.47, name=Bombardier, steeringwheel=RIGHT, speed=52.5%, direction=DOWN
x=1.08, y=0.0, z=9.47, name=Bombardier, steeringwheel=RIGHT, speed=56.7%, direction=STRAIGHT
x=1.08, y=0.54, z=10.01, name=Bombardier, steeringwheel=CENTER, speed=54.99%, direction=UP
x=1.58, y=0.54, z=9.5, name=Bombardier, steeringwheel=RIGHT, speed=50.04%, direction=DOWN
x=1.58, y=1.04, z=9.5, name=Bombardier, steeringwheel=CENTER, speed=50.04%, direction=STRAIGHT
x=1.58, y=1.51, z=9.97, name=Bombardier, steeringwheel=CENTER, speed=47.03%, direction=UP
x=2.04, y=1.51, z=9.5, name=Bombardier, steeringwheel=RIGHT, speed=46.55%, direction=DOWN
x=2.45, y=1.51, z=9.5, name=Bombardier, steeringwheel=RIGHT, speed=41.89%, direction=STRAIGHT
x=2.45, y=1.96, z=9.95, name=Bombardier, steeringwheel=CENTER, speed=45.66%, direction=UP
x=2.01, y=1.96, z=9.51, name=Bombardier, steeringwheel=LEFT, speed=43.03%, direction=DOWN
```

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

```java
/**
 * main method
 * @param args to call the main function in java. The main method is called if it's formal parameter matches that of an array of Strings.
 */
public static void main(String[] args) {
    Main main = new Main();
    System.out.println("----------------------------------------------------------------------------------------");
    main.moveLandVehicle( name: "Toyota");
    System.out.println("----------------------------------------------------------------------------------------");
    main.moveLandVehicle( name: "Volkswagen");
    System.out.println("----------------------------------------------------------------------------------------");
    main.moveLandVehicle( name: "Ford");
    System.out.println("----------------------------------------------------------------------------------------");
    main.moveAirVehicle( name: "Boeing");
    System.out.println("----------------------------------------------------------------------------------------");
    main.moveAirVehicle( name: "Airbus");
    System.out.println("----------------------------------------------------------------------------------------");
    main.moveAirVehicle( name: "Bombardier");
    System.out.println("----------------------------------------------------------------------------------------");
}

/**
```

```java
/**
 * move LandVihicle for 10 actions
 * @param name give name
 */
public void moveLandVehicle(String name) {
    Vehicle vehicle = new LandVehicle( type: "LandVehicle", name);
    for (int i = 0; i < 10; i++) {
        Random random = new Random();
        double randomGasPedalSpeed = random.nextInt( bound: 10);
        double randomBrakePedalSpeed = random.nextInt( bound: 10);
        int randomPedal = new Random().nextInt(Pedal.values().length);
        Pedal pedal = Pedal.values()[randomPedal];
        int randomSteeringWheel = new Random().nextInt(SteeringWheel.values().length);
        SteeringWheel steeringWheel = SteeringWheel.values()[randomSteeringWheel];
        vehicle.move(pedal, randomGasPedalSpeed, randomBrakePedalSpeed,  propellerSpeed: 0, steeringWheel, Direction.STRAIGHT);
        System.out.println(vehicle.toString());
        try {
            Thread.sleep( millis: 1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

```java
    /**
     * move AirVihicle for 10 actions
     * @param name give name
     */
    public void moveAirVehicle(String name) {
        Vehicle vehicle = new AirVehicle( type: "AirVehicle", name);
        for (int i = 0; i < 10; i++) {
            Random random = new Random();
            double propellerSpeed = random.nextInt( bound: 20) - 10;
            int randomSteeringWheel = new Random().nextInt(SteeringWheel.values().length);
            SteeringWheel steeringWheel = SteeringWheel.values()[randomSteeringWheel];
            int randomDirection = new Random().nextInt(Direction.values().length);
            Direction direction = Direction.values()[randomDirection];
            vehicle.move( pedal: null,  gasPedalSpeed: 0,  brakePedalSpeed: 0, propellerSpeed, steeringWheel, direction);
            System.out.println(vehicle.toString());
            try {
                Thread.sleep( millis: 1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

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

```java
    /**
     * move AirVihicle for 10 actions
     * @param name give name
     */
    public void moveAirVehicle(String name) {
        Vehicle vehicle = new AirVehicle( type: "AirVehicle", name);
        for (int i = 0; i < 10; i++) {
            Random random = new Random();
            double propellerSpeed = random.nextInt( bound: 20) - 10;
            int randomSteeringWheel = new Random().nextInt(SteeringWheel.values().length);
            SteeringWheel steeringWheel = SteeringWheel.values()[randomSteeringWheel];
            int randomDirection = new Random().nextInt(Direction.values().length);
            Direction direction = Direction.values()[randomDirection];
            vehicle.move( pedal: null,  gasPedalSpeed: 0,  brakePedalSpeed: 0, propellerSpeed, steeringWheel, direction);
            System.out.println(vehicle.toString());
            try {
                Thread.sleep( millis: 1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

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

```java
public void moveLandVehicle(String name) {
    Vehicle vehicle = new LandVehicle( type: "LandVehicle", name);
    for (int i = 0; i < 10; i++) {
        Random random = new Random();
        double randomGasPedalSpeed = random.nextInt( bound: 10);
        double randomBrakePedalSpeed = random.nextInt( bound: 10);
        int randomPedal = new Random().nextInt(Pedal.values().length);
        Pedal pedal = Pedal.values()[randomPedal];
        int randomSteeringWheel = new Random().nextInt(SteeringWheel.values().length);
        SteeringWheel steeringWheel = SteeringWheel.values()[randomSteeringWheel];
        vehicle.move(pedal, randomGasPedalSpeed, randomBrakePedalSpeed,  propellerSpeed: 0, steeringWheel, Direction.STRAIGHT);
        System.out.println(vehicle.toString());
        try {
            Thread.sleep( millis: 1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

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)

```java
/**
 * move AirVihicle for 10 actions
 * @param name give name
 */
public void moveAirVehicle(String name) {
    Vehicle vehicle = new AirVehicle( type: "AirVehicle", name);
    for (int i = 0; i < 10; i++) {
        Random random = new Random();
        double propellerSpeed = random.nextInt( bound: 20) - 10;
        int randomSteeringWheel = new Random().nextInt(SteeringWheel.values().length);
        SteeringWheel steeringWheel = SteeringWheel.values()[randomSteeringWheel];
        // int randomDirection = new Random().nextInt(Direction.values().length);
        int randomDirection = i % 3;
        Direction direction = Direction.values()[randomDirection];
        vehicle.move( pedal: null,  gasPedalSpeed: 0,  brakePedalSpeed: 0, propellerSpeed, steeringWheel, direction);
        System.out.println(vehicle.toString());
        try {
            Thread.sleep( millis: 1000);
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

Unnamed

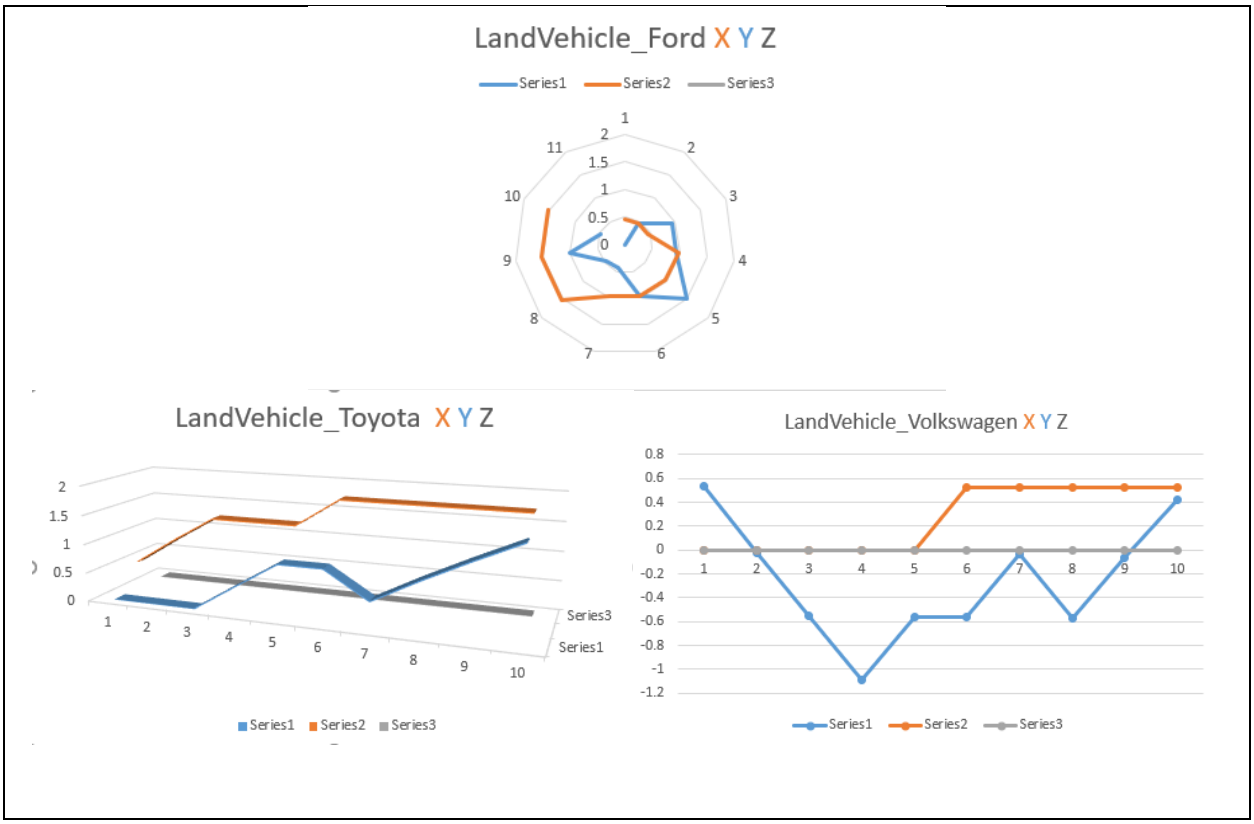
```
--------------------------------------------------------------------------------
x=0.92, y=0.0, z=9.08, name=Boeing, steeringwheel=RIGHT, speed=92.0%, direction=DOWN
x=0.92, y=0.91, z=9.08, name=Boeing, steeringwheel=CENTER, speed=91.08%, direction=STRAIGHT
x=0.92, y=1.9, z=10.07, name=Boeing, steeringwheel=CENTER, speed=99.27%, direction=UP
x=-0.17, y=1.9, z=8.98, name=Boeing, steeringwheel=LEFT, speed=108.2%, direction=DOWN
x=0.86, y=1.9, z=8.98, name=Boeing, steeringwheel=RIGHT, speed=103.87%, direction=STRAIGHT
x=0.86, y=2.94, z=10.02, name=Boeing, steeringwheel=CENTER, speed=104.9%, direction=UP
x=-0.24, y=2.94, z=8.92, name=Boeing, steeringwheel=LEFT, speed=109.09%, direction=DOWN
x=-0.24, y=4.11, z=8.92, name=Boeing, steeringwheel=CENTER, speed=117.81%, direction=STRAIGHT
x=0.9, y=4.11, z=10.06, name=Boeing, steeringwheel=RIGHT, speed=114.27%, direction=UP
x=0.9, y=5.33, z=8.83, name=Boeing, steeringwheel=CENTER, speed=122.26%, direction=DOWN
--------------------------------------------------------------------------------
```

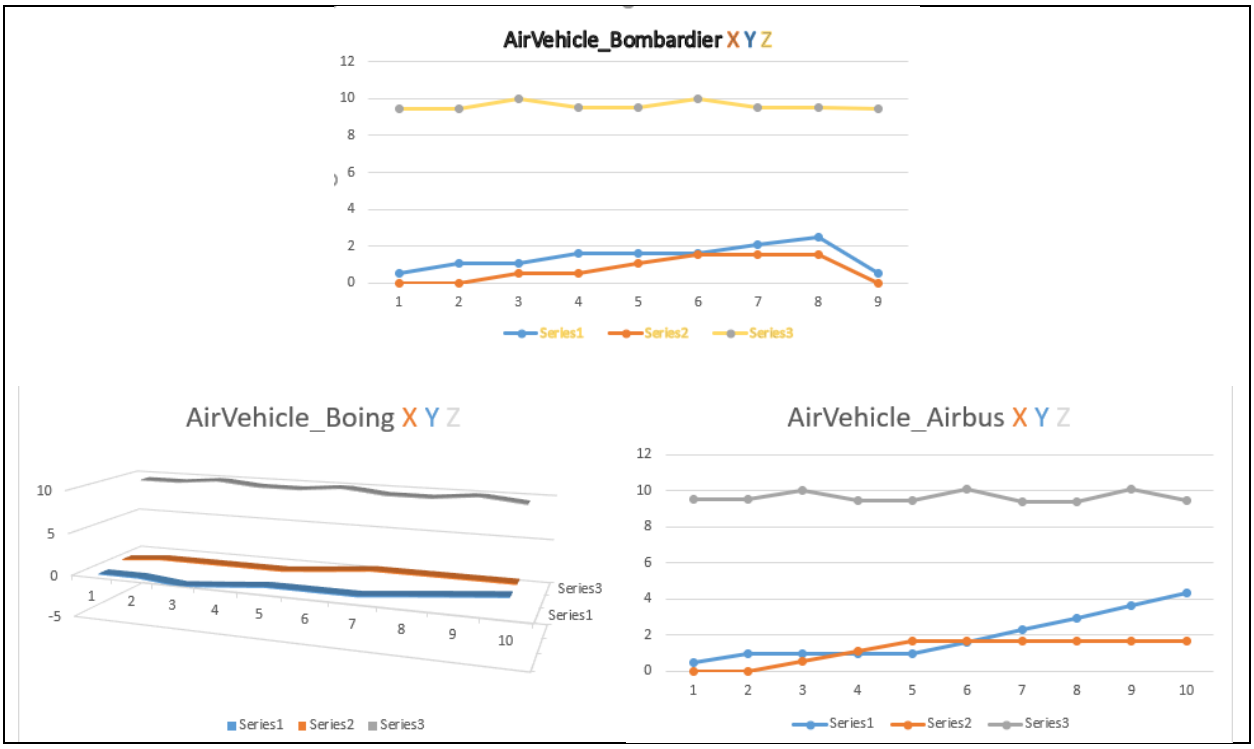
The vehicle rebounds from outer boundaries so it stays within your viewing area.

```java
@Override
public void move(Pedal pedal, double gasPedalSpeed, double brakePedalSpeed, double propellerSpeed, SteeringWheel steeringWheel, Direction direction) {
    if (pedal.equals(Pedal.GAS)) {
        this.setSpeed(this.getSpeed() * (1 + gasPedalSpeed / 100));
    } else if (pedal.equals(Pedal.BRAKE)) {
        double currentSpeed = this.getSpeed() * (1 - brakePedalSpeed / 100);
        currentSpeed = currentSpeed > 0 ? currentSpeed : 0;
        this.setSpeed(currentSpeed);
    }
    this.setPedal(pedal);
    this.setDirection(direction);
    this.setSteeringWheel(steeringWheel);

    if (steeringWheel.equals(SteeringWheel.LEFT)) {
        double x = this.getPosition().getX() + (this.getSpeed() / 100) * -1;
        if (x < -10 || x > 10) {
            x = 0;
        }
        this.getPosition().setX(x);
    } else if (steeringWheel.equals(SteeringWheel.CENTER)) {
        double y = this.getPosition().getY() + (this.getSpeed() / 100) * 1;
        if (y > 100) {
            y = 0;
        }
        this.getPosition().setY(y);
    } else if (steeringWheel.equals(SteeringWheel.RIGHT)) {
        double x = this.getPosition().getX() + (this.getSpeed() / 100) * 1;
        if (x < -10 || x > 10) {
            x = 0;
        }
        this.getPosition().setX(x);
    }
}
```

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





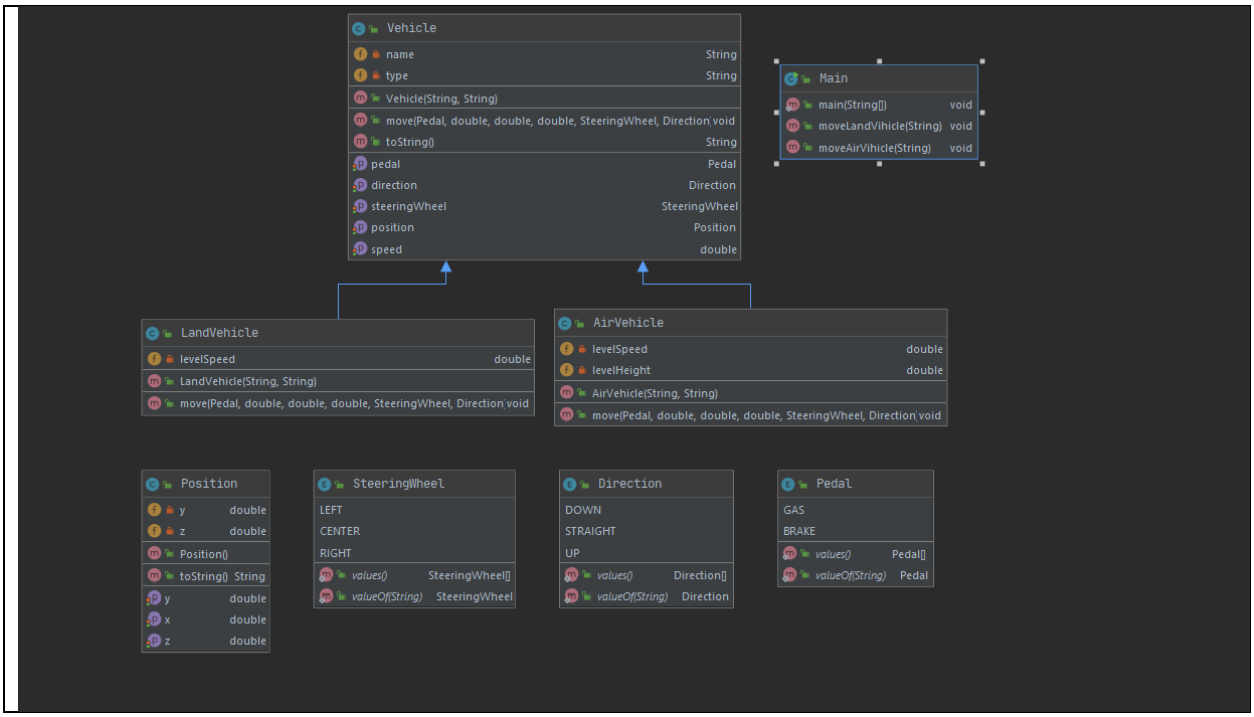
The physics of the movement need not be perfect, but left and right would change the x, y coordinates, and height would change the z coordinate in the proper general direction.

```
public AirVehicle(String type, String name) {
    super(type, name);
    this.getPosition().setZ(levelHeight);
    this.setSpeed(levelSpeed);
}
```

```
if (steeringWheel.equals(SteeringWheel.LEFT)) {
    double x = this.getPosition().getX() + (this.getSpeed() / 100) * -1;
    this.getPosition().setX(x);
} else if (steeringWheel.equals(SteeringWheel.CENTER)) {
    double y = this.getPosition().getY() + (this.getSpeed() / 100) * 1;
    this.getPosition().setY(y);
} else if (steeringWheel.equals(SteeringWheel.RIGHT)) {
    double x = this.getPosition().getX() + (this.getSpeed() / 100) * 1;
    this.getPosition().setX(x);
}
```

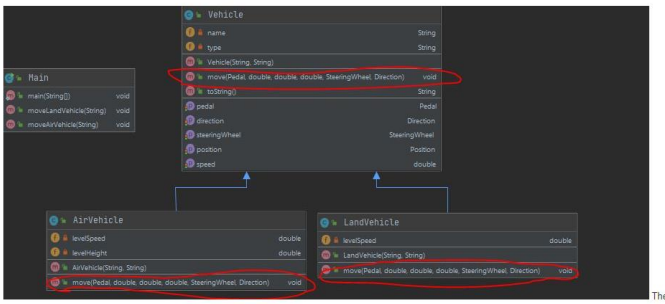
TEST: test with one vehicle at a time first.

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



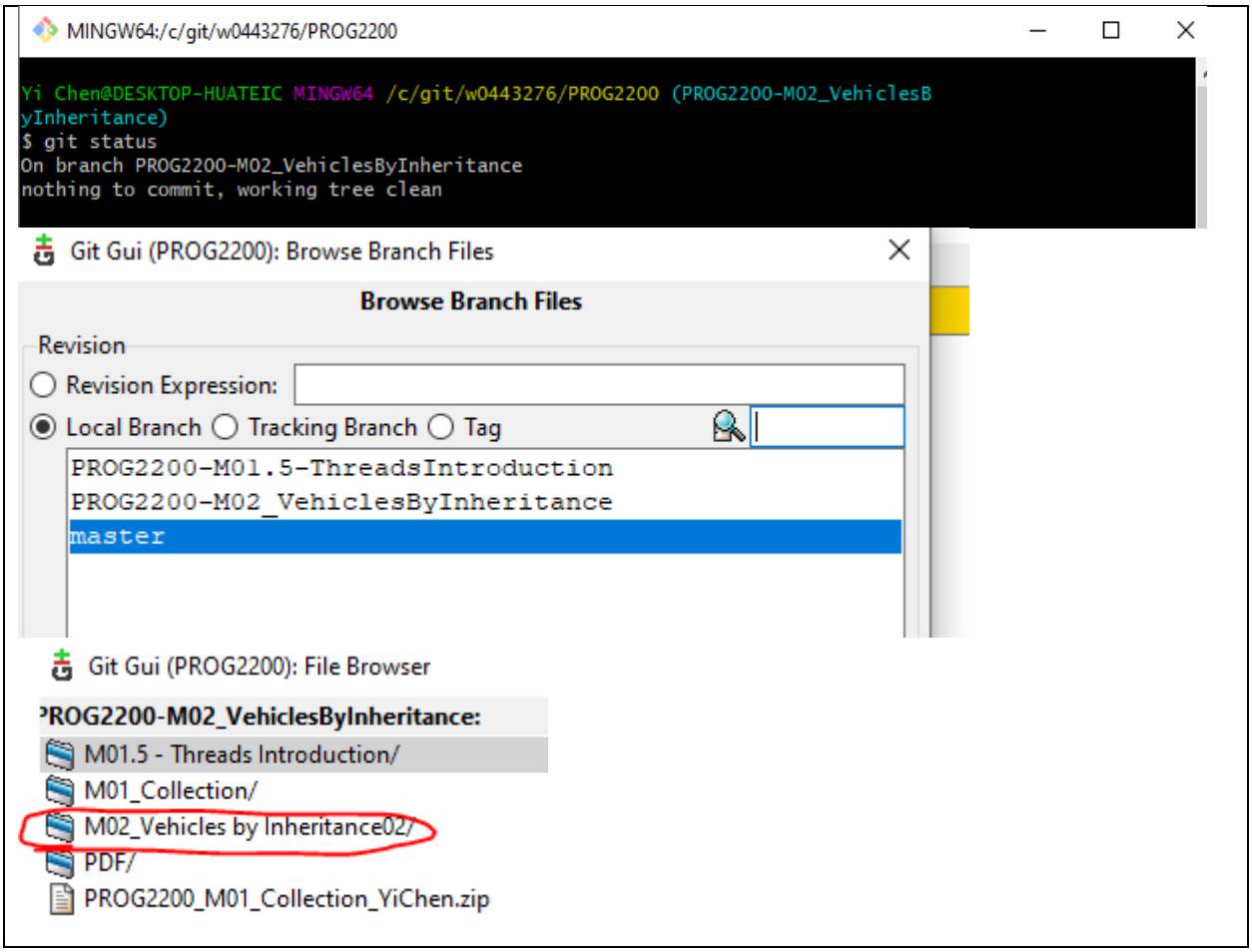
Copy the table below into your package-level JavaDoc documentation, and PDF. For at least one concept, describe how you implemented, or could implement it.

My Application of this concept

	acronym	Concept	My Application of this concept
S	SRP	Single responsibility principle	Enum direction class have responsibility over a single part of that program's move functionality
O	OCP	Open/closed principle	The implementation of the Vehicle class is relatively simple. It just has a constructor, a public method to add position, and a method that brews a direction, so on.
L	LSP	Liskov substitution principle	 The most effective way I have seen to illustrate this point was in LSP.
I	ISP	Interface segregation principle	NO ISP
D	DIP	Dependency inversion principle	That directly depends on one of the implementation classes is the Main class, which instantiates a moveLandVehicle object and moveAirVehicle.

Submit Artifacts for marking:

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.