CSE 1322L – Assignment 4 (Spring 2025)

Introduction

In this assignment, you will write a program for a gas station which is able to provide gas to gas-powered vehicles and electricity for electric vehicles. We will specify a simple standard which all vehicles and the gas station follow, ensuring that both parties are fully compatible with one another.

You will be required to submit a UML diagram, along with your code. Check the last pages for details.

Requirements

The features described below must be in your program:

- ElectricEngine interface
  - **double rechargeBattery(double)**: Recharges the battery using the argument provided. If the argument provided would make the battery go over its maximum capacity, simply set the current charge to match the maximum capacity. The method should then return how much, in kW, was actually recharged.

    e.g.: If an ElectricEngine is currently at 15kW out of 20kW and rechargeBattery(3) is called, its current charge should go to 18kW and the method would return 3.

    e.g.: If an ElectricEngine is currently at 15kW out of 20kW and rechargeBattery(15) is called, its current charge should go to 20kW (maximum capacity) and the method would return 5 (how much was actually charged).

  - **double getMaxBattery()**: Returns the battery's maximum capacity, in kW
  - **double getCurrentCharge()**: Returns the current available charge, in kW
- GasEngine interface
  - **double refuelTank(double)**: Refuels the tank by the argument provided. If the argument provided would make the tank go over its maximum capacity, simply set the current volume to match the maximum capacity. The method should then return how much, in gallons, was actually refueled.

    e.g.: If a GasEngine is currently at 15 gals out of 20 gals and refuelTank(3) is called, its current volume should go to 18 gals and the method would return 3.

e.g.: If a GasEngine is currently at 15 gals out of 20 gals and refuelTank(15) is called, its current volume should go to 20 gals (maximum capacity) and the method would return 5 (how much was actually refueled).

  - **double getTankCapacity()**: Returns the amount of fuel the car can hold, in gallons
  - **double getCurrentVolume()**: Returns the current amount of fuel, in gallons
- Car class
  - Is abstract
  - Has 2 fields: id (integer), nextId (static integer initialized at 0)
  - **Car()**: Initializes id with the contents of nextId, incrementing nextId by 1.
  - Id has a getter
- GasCar class
  - Is a subclass of Car
  - Implements GasEngine
  - Has two double fields: tankCapacity and currentVolume
  - **GasCar(double, double)**: Uses the first argument to initialize tankCapacity, and the second one to initialize currentVolume.
  - An override of toString(), which returns the following string (where X is the Car's id, Y is its current fuel volume, and Z is its tank capacity:

```
Car: X | Current Fuel: Y/Z gals
```

- ElectricCar class
  - Is a subclass of Car
  - Implements ElectricEngine
  - Has two double fields: maxBattery and currentCharge
  - **ElectricCar(double, double)**: Uses the first argument to initialize maxBattery, and the second one to initialize currentCharge.
  - An override of toString(), which returns the following string (where X is the Car's id, Y is its current charge, and Z is its battery capacity:

```
Car: X | Current Charge: Y/Z kW
```

- HybridCar class
  - Is a subclass of Car
  - Implements GasEngine and ElectricEngine
  - Has four double fields: maxBattery, currentCharge, tankCapacity and currentVolume
  - **HybridCar(double, double, double, double)**: Uses the first argument to initialize maxBattery, the second one to initialize currentCharge, the third one to initialize tankCapacity, and the last one to initialize currentVolume

- An override of toString(), which returns the following string (where X is the Car's id, Y is its current charge, Z is its battery capacity, A is its current volume, and B is its tank capacity:

```
Car: X | Current Charge: Y/Z kW | Current Fuel: A/B gals
```

- Driver class:
  - Creates an arraylist of Cars called "cars", which starts out empty.
  - Creates a double called "balance". It will be used to keep track of how much money the gas station has generated.
  - It must then implement the menu options below, in a loop:
    1. **Add gas car:** Creates a GasCar object and places it in cars. Its tank capacity and current amount should both be randomly generated integers. Tank capacity should be a value between 15 and 30 (both inclusive), while current amount should be a value between 2 and 14, both inclusive. Check the considerations below on how to achieve this.
    2. **Add electric car:** Creates an ElectricCar object and places it in cars. Its battery capacity and current change should both be randomly generated. The capacity should be a value between 40 and 100, both inclusive, while the current charge should be a value between 2 and 30, both inclusive.
    3. **Add hybrid car**: Creates a HybridCar object and places it in cars. All of its fields should be randomly generated according to the rules above, as appropriate.
    4. **Refuel all gas engines**: Completely refuels all Car objects in the arraylist that implement GasEngine. Be sure to keep track of the revenue generated, at $4 per gallon.
    5. **Recharge all electric engines**: Completely recharges all Car objects in the arraylist that implement ElectricEngine. Be sure to keep track of the revenue generated, at $0.20 per kW.
    6. **Refuel and recharge all vehicles**: Completely refuels and recharges all Car objects in the arraylist. Be sure to keep track of the revenue generated, as per the prices above.
    7. **Display all vehicles**: Calls and prints the toString() of all Cars
    8. **Dispatch all vehicles**: Clears the arraylist of all items
    9. **Quit**: Terminates the program

Deliverables

- Assignment4.java (driver)

- Car.java
- ElectricCar.java
- GasCar.java
- HybridCar.java
- ElectricEngine.java
- GasEngine.java
- UML.pdf (UML diagram)

Considerations

- This assignment may seem intimidating, but that's just because of the number of things you have to do in the driver. Don't forget that you will get partial credit for partial work.
- Feel free to add any helper methods which you believe are necessary, but know you will be not graded for them.
- Despite what the deliverables say above, you can place all of your classes in a single file
    - You will notice that placing all of your code in a single file, while still valid, will make your single file very large and potentially harder to navigate. In general, you will want to place different classes on different files.
- For variables of type double, you will **not** lose points due to rounding errors. If the sample output below says "1.00001" and yours says "1.00002", you will still get full credit **as long as your logic is correct.** Similarly, you do **not** need to round off your numbers to a specific number of decimal places. If the sample output says "2.75" and yours says "2.747155482733", you will still get full credit **as long as your logic is correct.**
- Recall that, because most of your classes implement interfaces, all methods specified in the interfaces must be present as concrete methods in the classes that implement them.
- When you retrieve a Car from the arraylist, you will need to check what its specific instance is. This is because you will first need to cast that instance to the appropriate interface before you can call the methods that the interface specify. If don't check the instance, your cast may fail, crashing your program.
- You can generate random numbers by first importing the Random library, at the top of your driver:

```
import java.util.Random;
```

- You can then create a Random object and call its randInt(int) method.

```
Random r = new Random();
r.nextInt(10);
```

- Note that randInt(int) will generate a number between 0 (inclusive) and the argument passed (non-inclusive).

```
r.nextInt(5); // Generates either 0, 1, 2, 3, or 4
```

- Note that we can adjust the upper and lower bounds of our randomly generated number by adding to its result.

```
r.nextInt(5) + 2; // Generates either 2, 3, 4, 5, or 6
```

Sample Output (user input in red)

```
[The Refueler Station]
1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 1

Vehicle 0 has parked at pump

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
```

```
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 2

Vehicle 1 has parked at charger

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 3

Vehicle 2 has parked at pump-charger

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
```

```
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 3

Vehicle 3 has parked at pump-charger

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 2

Vehicle 4 has parked at charger

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 1
```

```
Vehicle 5 has parked at pump

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 7

Car: 0 | Current Fuel: 4.00/21.00 gals
Car: 1 | Current Charge: 9.00/98.00kW
Car: 2 | Current Charge: 6.00/99.00kW | Current Fuel: 13.00/20.00 gals
Car: 3 | Current Charge: 13.00/83.00kW | Current Fuel: 7.00/15.00 gals
Car: 4 | Current Charge: 15.00/99.00kW
Car: 5 | Current Fuel: 3.00/17.00 gals

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
8: Dispatch all vehicles
```

```
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 4

All gas vehicles refueled. Current balance is $184.00

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 7

Car: 0 | Current Fuel: 21.00/21.00 gals
Car: 1 | Current Charge: 9.00/98.00kW
Car: 2 | Current Charge: 6.00/99.00kW | Current Fuel: 20.00/20.00 gals
Car: 3 | Current Charge: 13.00/83.00kW | Current Fuel: 15.00/15.00 gals
Car: 4 | Current Charge: 15.00/99.00kW
Car: 5 | Current Fuel: 17.00/17.00 gals

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
```

6. Refuel and recharge all vehicles

7. Display all vehicles

8: Dispatch all vehicles

9. Quit

Gas: $4.00/gal | Electricity: $0.20/kW

Enter option: **5**

All electric vehicles recharged. Current balance is $251.20

1. Add gas car

2. Add electric car

3. Add hybrid car

4. Refuel all gas engines

5. Recharge all electric engines

6. Refuel and recharge all vehicles

7. Display all vehicles

8: Dispatch all vehicles

9. Quit

Gas: $4.00/gal | Electricity: $0.20/kW

Enter option: **7**

Car: 0 | Current Fuel: 21.00/21.00 gals

Car: 1 | Current Charge: 98.00/98.00kW

Car: 2 | Current Charge: 99.00/99.00kW | Current Fuel: 20.00/20.00 gals

Car: 3 | Current Charge: 83.00/83.00kW | Current Fuel: 15.00/15.00 gals

Car: 4 | Current Charge: 99.00/99.00kW

Car: 5 | Current Fuel: 17.00/17.00 gals

1. Add gas car

2. Add electric car

```
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 8

All vehicles have left.

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 1

Vehicle 6 has parked at pump

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
```

```
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 2

Vehicle 7 has parked at charger

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 3

Vehicle 8 has parked at pump-charger

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
```

7. Display all vehicles

8: Dispatch all vehicles

9. Quit

Gas: $4.00/gal | Electricity: $0.20/kW

Enter option: 7

Car: 6 | Current Fuel: 7.00/30.00 gals

Car: 7 | Current Charge: 6.00/79.00kW

Car: 8 | Current Charge: 2.00/71.00kW | Current Fuel: 12.00/27.00 gals

1. Add gas car

2. Add electric car

3. Add hybrid car

4. Refuel all gas engines

5. Recharge all electric engines

6. Refuel and recharge all vehicles

7. Display all vehicles

8: Dispatch all vehicles

9. Quit

Gas: $4.00/gal | Electricity: $0.20/kW

Enter option: 6

All vehicles refueled and recharged. Current balance is $431.60

1. Add gas car

2. Add electric car

3. Add hybrid car

4. Refuel all gas engines

5. Recharge all electric engines

6. Refuel and recharge all vehicles

```
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 7

Car: 6 | Current Fuel: 30.00/30.00 gals
Car: 7 | Current Charge: 79.00/79.00kW
Car: 8 | Current Charge: 71.00/71.00kW | Current Fuel: 27.00/27.00 gals

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 8

All vehicles have left.

1. Add gas car
2. Add electric car
3. Add hybrid car
4. Refuel all gas engines
5. Recharge all electric engines
6. Refuel and recharge all vehicles
```

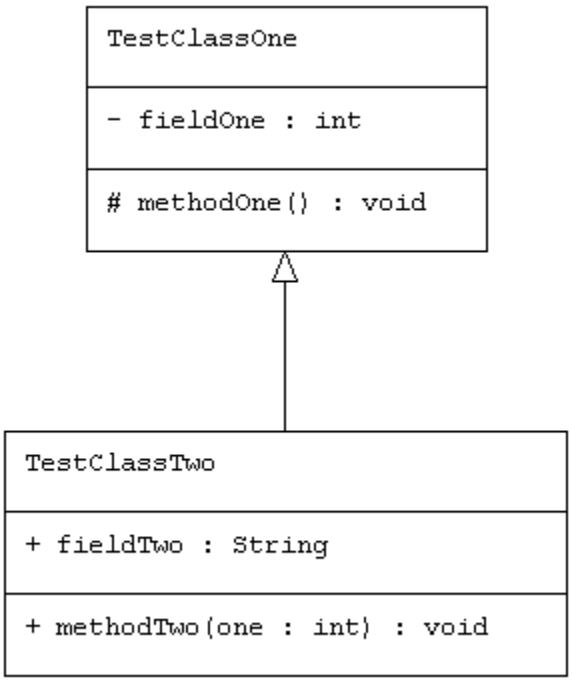
```
7. Display all vehicles
8: Dispatch all vehicles
9. Quit
Gas: $4.00/gal | Electricity: $0.20/kW
Enter option: 9

We've made a total of $431.60 today
Shutting off...
```

UML Diagram

Your UML diagram must be submitted in its own file, preferably as a PDF; image files are also acceptable. Note that if your grader is unable to open the file or if they are unable to verify that your diagram is correct due to poor formatting, it will not be graded.

The picture below shows what a class should look like in your diagram. Specifically:

- The box representing the class must be divided into 3 sections: name, fields, and behaviors
- The name must match the class name exactly
- The class members (fields and behaviors) must be preceded by the appropriate accessibility symbol (+ for public, - for private, and # for protected)
- A field must be listed as "name : type" (note the colon)
- A behavior must be listed as "name(name : type) : return_type"
- If two classes have a parent-child relationship, there must be an arrow pointing from the child to the parent



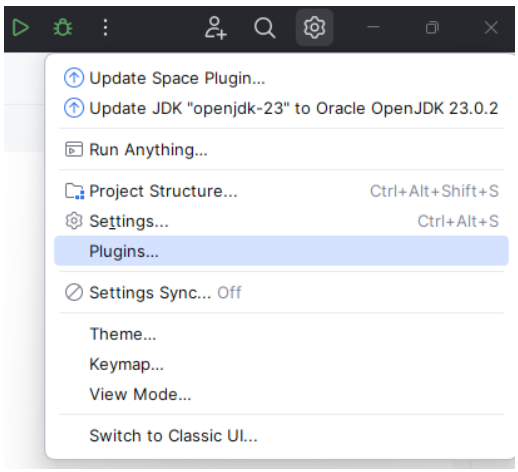
There are no restrictions on which program you use to generate the diagram. In fact, you could even hand-draw your diagram and upload a picture of it along with your code. **You are strongly discouraged from doing this.** Drawing your diagram manually, either by hand or by using any program besides the one listed below increases the chances of you missing something, which will lose you points. It's best to just write your code, and then have a program generate the diagram for you based on your code.

The next pages explains how to install and use the recommended UML diagram generator.

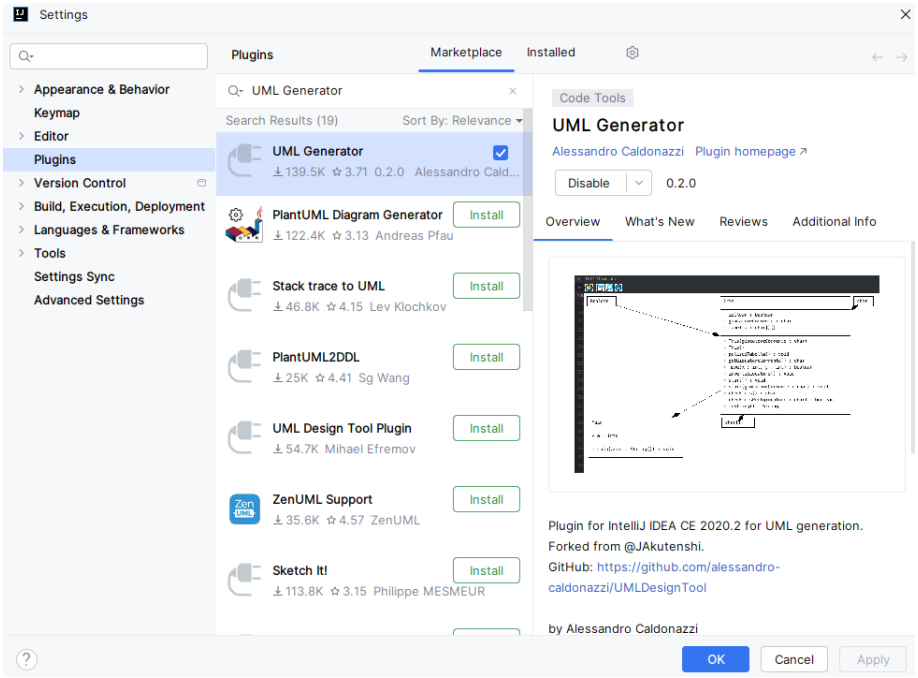
UML Generator

The recommended UML diagram generator is called “UML Generator”, developed by Alessandro Caldonazzi. Instructions on installing and using it can be found below:

- With IntelliJ open, open up your list of plugins by clicking the gear icon at the top right, followed by “Plugins”
    - If you haven’t updated IntelliJ in a while, the gear icon might be an arrow pointing upwards

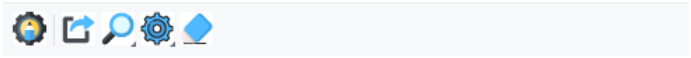


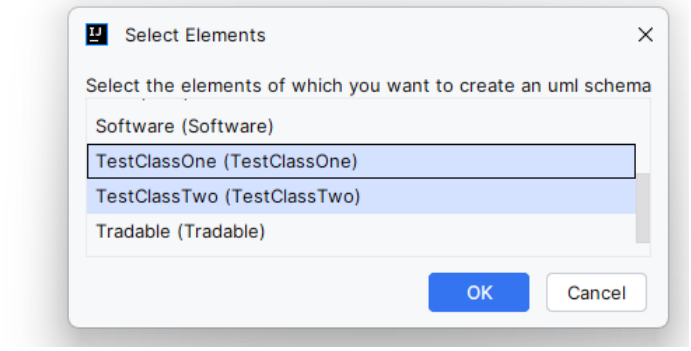
- While in the Plugins menu on the left, select “Marketplace” at the top
- Search for “UML Generator”, developed by Alessandro Caldonazzi



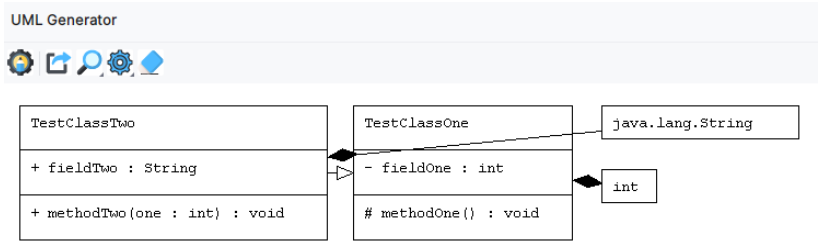
- Click on install. Once it’s done, you may be asked to restart IntelliJ. Please do so.

- On the far right, you will now see a UML button. Clicking on it will show you a blank window with some buttons
- To generate your diagram, click the gear icon with the pencil in the middle (the first button). Then, select all the classes that are part of your diagram using the control key. Finally, click ok.

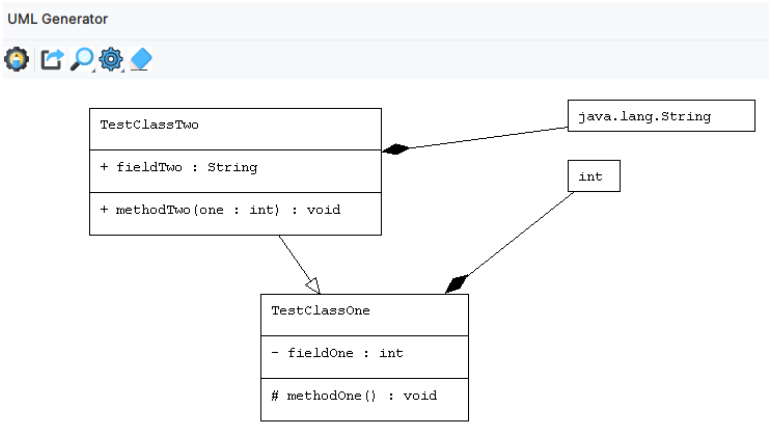




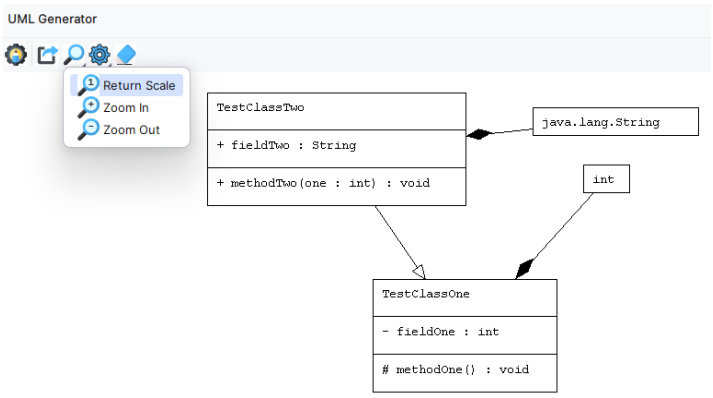
- Your initial diagram will likely look disorganized.



- You can drag its entities with your mouse, to make your diagram more legible



- You can zoom in and out using the magnifier glass button. You can pan around the image using the scroll bars at the right and bottom



- Once your diagram is presentable, click save (the second button). Select a place to save, and give your diagram a name. **Do not use the name provided in the name box, as that will make your diagram fail to save**.
- Don't forget to submit your diagram along with your code to Gradescope.