Exercises module 15 – OO Design

15.1 – Aggregation

1. We are given this car class:

```
public class Car {
    private Tire[] wheels;
    private Door[] doors;
    private float weight;

    public Car (float weight, Tire[] wheels, Door[] doors) {
        this.weight = weight;
        this.wheels = wheels;
        this.doors = doors;
    }

    public float getWeight()
{
        //Implement logic here to get the total weight of
        //the car + all the doors and wheels
        return;
    }
}
```

2. Create a suitable **Tire** and **Door** class where each class have:
    a. A private weight field
    b. The weight is set via the constructor
    c. Each class have a public method named **getWeight** that returns a float with its own weight.

3. Complete the **getWeight** method in the Car class so that it returns the total weight of the car, tires and doors

4. Try to create a car with 4 doors (weight 150 each) and 4 tires (Weight 100 each) and the car itself should have a weight of 1000. Then the total weight returned by the car should be 2000.

```
Car car = new Car(1000,tires, doors);

System.out.println("Total car weight " + car.getWeight());
```

15.2 – Single responsibility principle

1. According to the single responsibility principle a class should only have one focus/purpose.

2. Our job is to refactor the code given below so that it better follows the single responsibility principle.

```
public class Bank {

    public void sendMoney(int amount, String fromAccount, String toAccount) throws
IOException {

        String logstr = "";

        //Calculate the transfer cost
        int paymentCost = 10;
        if (amount > 1000)
            paymentCost = 50;

        //Write to a log file for debugging purposes
        logstr = "Bank auditlog" + "\r\n";
        Files.write(Paths.get("./log.txt"), logstr.getBytes(),
StandardOpenOption.CREATE);
        logstr = "Doing a desposit of " + amount + " Sek from " + fromAccount + " to
" + toAccount + "\r\n";
        Files.write(Paths.get("./log.txt"), logstr.getBytes(),
StandardOpenOption.APPEND);
        logstr = "Payment cost" + paymentCost + "\r\n";
        Files.write(Paths.get("./log.txt"), logstr.getBytes(),
StandardOpenOption.APPEND);


        System.out.printf("Payment cost " + paymentCost);

        //save a receipt to disk
        String reciept = "Bank transfer reciept\r\n" +
                "\r\nDate: " + LocalDateTime.now() +
                "\r\nAmount:" + amount +
                "\r\nCost: " + paymentCost +
                "\r\nFrom account:" + fromAccount +
                "\r\nTo account:" + toAccount + "\r\n";
        Files.write(Paths.get("./confirmation.txt"), reciept.getBytes(),
StandardOpenOption.CREATE);

        //Doing the bank transfer somehow....
    }
}
```

3. Do the following:
    - Read up on the principle here https://en.wikipedia.org/wiki/Single_responsibility_principle
    - Refactor the code so that the calculation of the payment cost is handled in a private method inside the bank class
    - Refactor the code so that all the code that writes to the log.txt file is located in a separate class named **Logging**.
    - Refactor the code so that all the code that writes to the receipt (confirmation.txt) is located in a separate class named **Receipt**.

    As you should notice is that now the Bank class focus on the banking operations and the receipt/logging operations in handled in separate classes. Allowing each class to focus on one thing.

Questions and concepts to study further on your own:

- Association vs Aggregation vs Composition
- Single responsibility principle
- Separation of concerns https://en.wikipedia.org/wiki/Separation_of_concerns
- Don't repeat yourself https://en.wikipedia.org/wiki/Don%27t_repeat_yourself