1.1 The SOLID violation that I found was the problem, of “Single Responsibility”
This is is triggered by the getBaddageAllowance() function, because the class is responsible to assign Passenger-types to the Passenger, and the calculation of how much Baggage Allowance a specific Passenger is allowed to take with him. This specific class would then do 2 thing, which breaks the SOLID rules. The Class should have one and only one responsibility.

The solution is to assign the calculation of the allowed baggage to a new class. So if there would be any new PassengerTypes it would be also easily expendable.

```
enum PassengerType{Economy, Flex,  Bussines}

class Baggage
{
    public double getBaggageAllowance(PassengerType passengerType)
    {
        switch (passengerType)
        {
            case Flex:
                return 32;
            case Economy:
                return 64;
            case Bussines:
                return 0;
        }
        return -1;
    }
}
public class Passenger {
    private PassengerType passengerType;

    public PassengerType getPassengerType() {
        return passengerType;
    }
    public void setPassengerType(PassengerType passengerType) {
        this.passengerType = passengerType;
    }
}
```