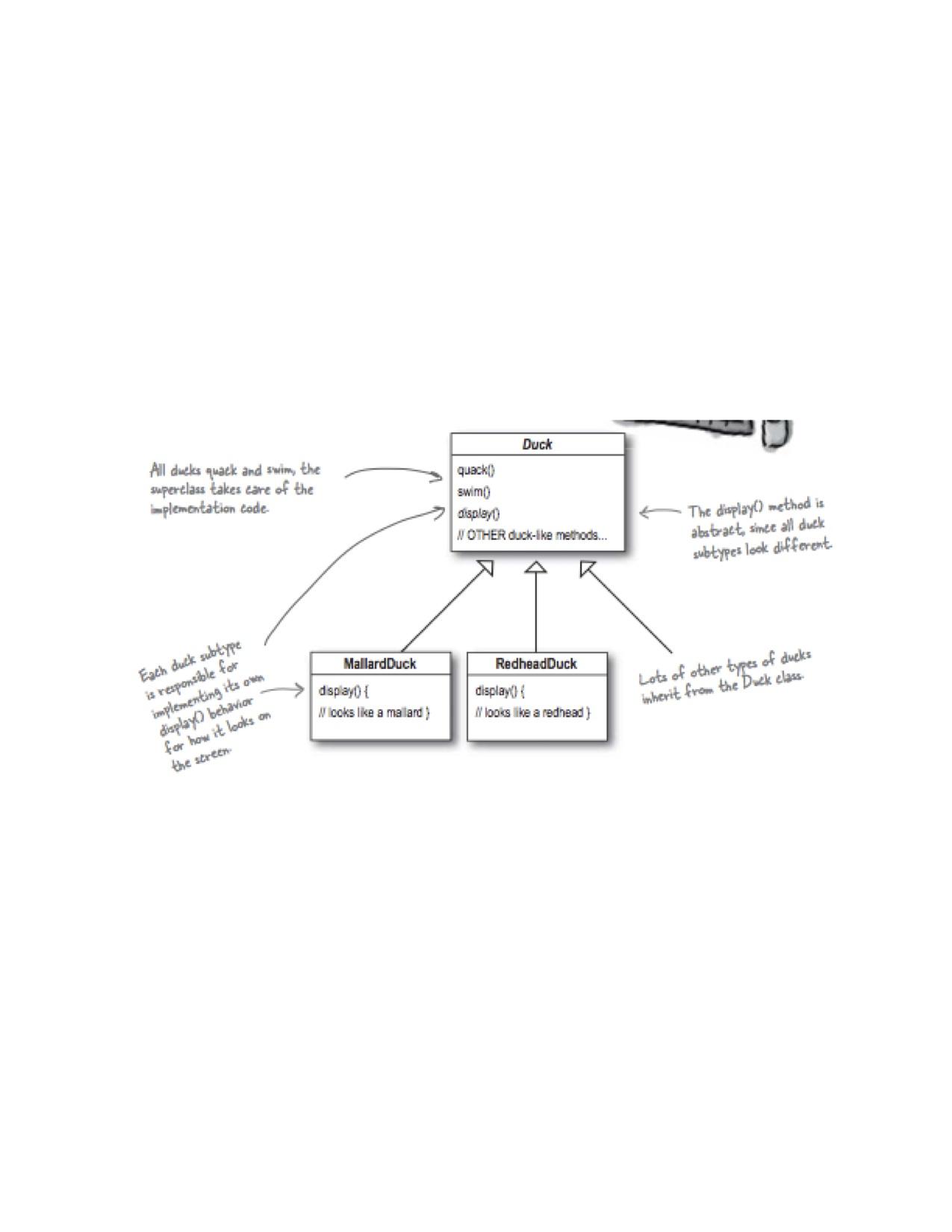
Task 2 :-



1) Do you see any issues with the Duck class and its sub classes below ?

Ans :- No, as per above figure can’t see any issue with the duck class and it’s sub class.

As per duck class is a abstract class because of one method is abstract which Is display() method and duck class extends into two different class which Is mallardDuck and RedheadDuck

While extends that display() methods need to implement on extended class. As per requirement.

Sudo code like

Abstract Class Duck {

Public void Quack(){}

Public void swim(){}

Public void Display();

}

Class mallardDuck extends Duck {

@Override

Public void Display(){//need to implements

}

}

Class RedheadDuck extends Duck {

@Override

Public void Display(){

//need to implements

}

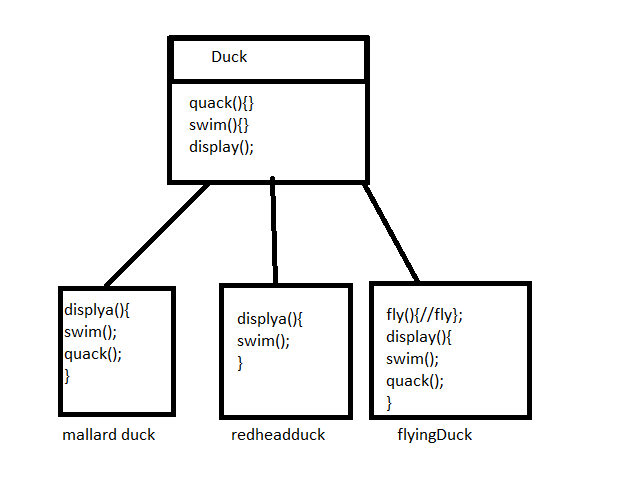
}

2) Ducks usually don’t fly. But what if I want to add a new duck “FlyingDuck” that can fly. How will this sub class look like ? Could you write a class diagram for both parent Duck and FlyingDuck Class ?

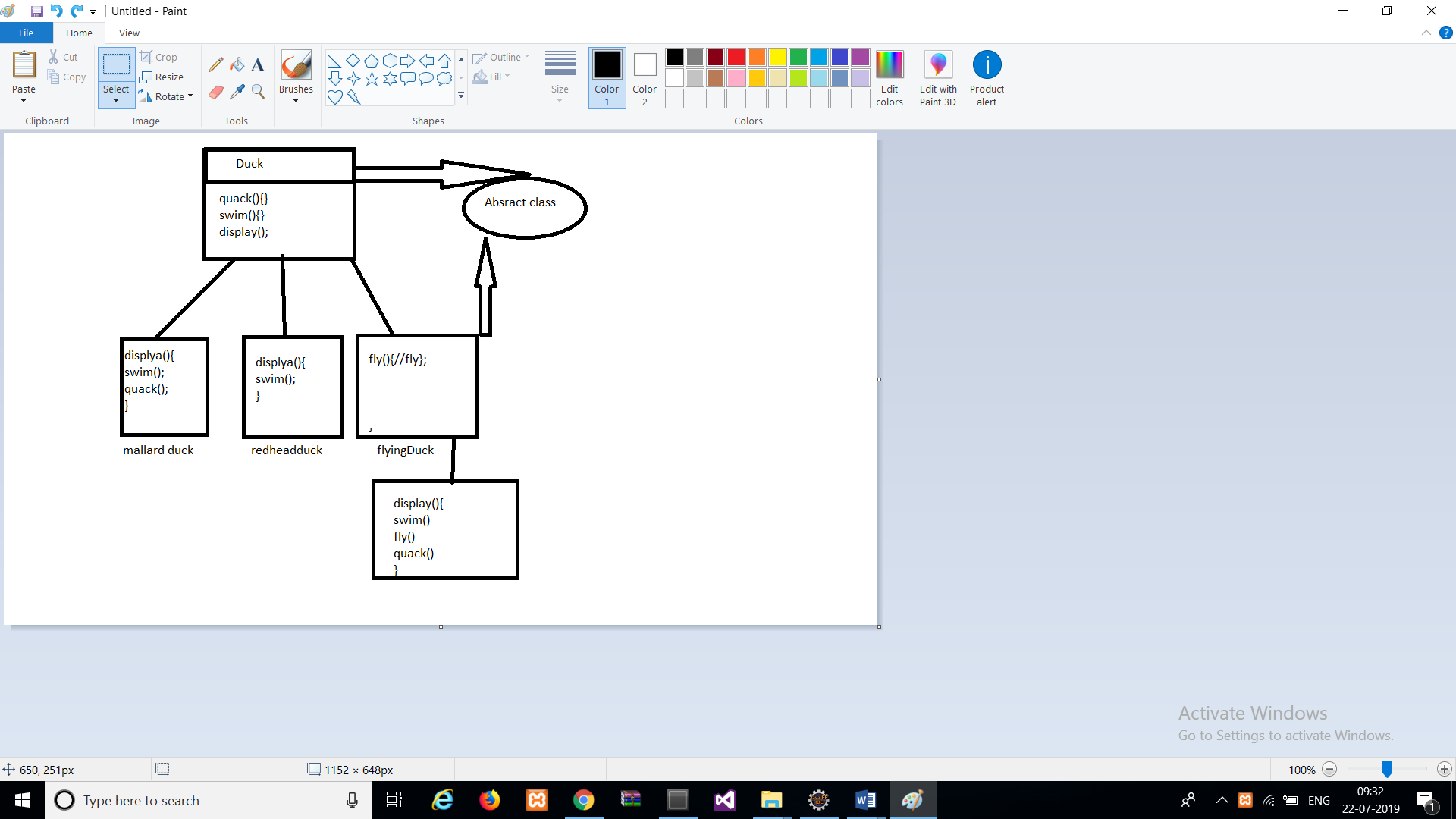
Ans:- as per above diagram duck has only two proertywhich is quack and swim. but in case of flying duck has one extra property which is fly . but flying duck also quack and swim also that is to be implements on sub class of duck.

That is two way to impelments

1)first method is directly implement one method fly() on extended duck class into another child class of flying duck .



2)extends one another abstract class and add another method which to be implements



3) Why using a interface to take out the flying and quacking behavior might be a bad programming design ?

Asn:- in term of interface it’s provide multiple inheritance so it’s a good programming practice to design a structure and it’s also helpful to the flying and quacking behaviour. Because we provide interface implementation in a class so it’s not given any ambiguity problem in implemented class.so it’s a good programming practice.

4) Why overriding inherited fly and quack methods can be a bad programming design ?

Ans : - NO. it’s not a bad programming practice. Because it’s advantage is Code Reusability.

So we can’t write same method again and again.

5) Please re-write this skeleton code or class diagram to introduce flying behaviour dynamically for birds that can fly and cant fly and birds that can quack to make different sounds using Strategy Pattern ?

Ans: -

