I chose the observer and strategy design patterns. For the observer part I created two new interfaces one called subject which contains methods that RaceWeather, the concrete subject class, will use to register, remove observers, and broadcast weather updates to them. The observer interface is implemented by the RaceCar class as to be updated in runtime for any weather changes. For the strategy part I have created an interface called DrivingBehavior which is implemented in two separate classes called CautiousDriving and FastDriving which has a driveCar() method which determines speed based on the current weather status and returns an integer collected by the DrivingBehavior object in the RaceCar class and added to the distance travelled. This way any new behaviors can simply implement the DrivingBehavior interface without any changes being made to RaceCar. Also, the two behaviors are not observers of RaceWeather since they don’t need to be updated every time the weather changes but only when they are needed by using the getWheather method of RaceWeather. The Race class is completely autonomous and works with the other classes by initialising their objects so if more cars are added to the race no problems will be caused. VideoLink: <https://youtu.be/VYxv_virxow>