To upgrade the Spindler battery, I would modify the `SpindlerBattery` class's `service_life` attribute to 3 instead of 2

```
class SpindlerBattery(Battery):
    def __init__(self):
        super().__init__(brand="Spindler", capacity=75, service_life=3)
```

To add tire servicing criteria, I would modify the `CarFactory` class to include a method for checking tire wear and servicing the tires if necessary:

```
class CarFactory:
    def __init__(self, engine_factory, battery_factory):
        self.engine_factory = engine_factory
        self.battery_factory = battery_factory

    def make_car(self, car_model, tire_wear):
        engine = self.engine_factory.build_engine(car_model)
        battery = self.battery_factory.build_battery(car_model)

        if car_model == "Thovex":
            return Car(engine=engine, battery=battery, tires=CarriganTires())

        if car_model == "Hakkitak":
            return Car(engine=engine, battery=battery, tires=OctoprimeTires())

        if car_model == "Snowy":
            return Car(engine=engine, battery=battery, tires=CarriganTires())

    def check_tire_wear(self, tires, tire_wear):
        total_wear = sum(tire_wear)
        if isinstance(tires, CarriganTires):
            return any(wear >= 0.9 for wear in tire_wear)
        elif isinstance(tires, OctoprimeTires):
```

```
        return total_wear >= 3
    else:
        raise ValueError("Unknown tire type")
```

I added a `check_tire_wear` method that takes in the `Tires` instance and the tire wear array, and returns `True` if the tires should be serviced and `False` otherwise. The method checks the tire wear according to the criteria given in the prompt and returns the appropriate boolean value.

To test these changes, I would write unit tests for the `SpindlerBattery` and `check_tire_wear` methods in the `CarFactory` class. Here is an example of a test case for `check_tire_wear`:

```
def test_check_tire_wear():
    cf = CarFactory(MockEngineFactory(), MockBatteryFactory())
    assert cf.check_tire_wear(CarriganTires(), [0.9, 0.8, 0.7, 0.6]) == True
    assert cf.check_tire_wear(CarriganTires(), [0.8, 0.7, 0.6, 0.5]) == False
    assert cf.check_tire_wear(OctoprimeTires(), [0.9, 0.8, 0.7, 0.6]) == False
    assert cf.check_tire_wear(OctoprimeTires(), [0.7, 0.6, 0.5, 0.4]) == True
```

This test case checks that the `check_tire_wear` method correctly returns `True` or `False` for each combination of `Tires` and tire wear array according to the criteria given in the prompt.