

# 重庆大学

## 车载软件开发基础

课后实践 3



**2022至2023学年第1学期**

学号	姓名
E2020149	张俊杰
任课教师	刘骥
成绩	

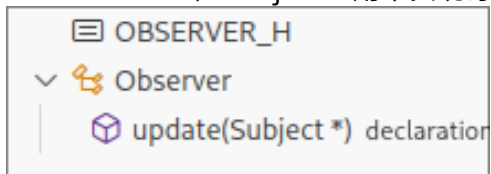
任务书	
任务内容	<p>请针对课后实践 2 的内容，补充多线激光雷达和底盘模块，使用订阅者模式，实现以下功能。</p> <ol style="list-style-type: none"><li>1、多线激光雷达作为主题方，发布主题，并接受订阅</li><li>2、底盘模块中增加观察者，向雷达模块进行订阅注册，并接收订阅通知。</li><li>3、当雷达更新障碍物状态时，底盘模块通过订阅者模式接收通知并执行对应操作。<ol style="list-style-type: none"><li>(1) 障碍我状态为“前方“，则底盘执行”后退“执行</li><li>(2) 障碍我状态为“右前方“，则底盘执行”左转“执行</li><li>(3) 障碍我状态为“左前方“，则底盘执行”右转“执行</li></ol></li></ol> <p>注意：</p> <ol style="list-style-type: none"><li>1、底盘执行指令，只需要在屏幕输出类似“左转。。。”文字即可。</li><li>2、障碍物状态可通过命令行方式输入。比如 1 代表前方障碍，2 代表左前方障碍，3 代表右前方障碍。</li></ol>
程序规范	<ol style="list-style-type: none"><li>(1) 所有程序代码采用 C++编写，使用 git 进行源代码管理；</li><li>(2) 类名、变量名、函数名应符合 C++的命名规范，并在代码中前后保持一致；</li><li>(3) 涉及面向对象的程序，例如自定义的类，应符合面向对象的设计原则；</li><li>(4) 正确使用头文件和源文件，自定义的头文件应符合头文件的编写原则，例如用条件宏定义确保头文件不被多次引用、不在头文件中进行类和函数的实现（模板除外）；</li></ol>
报告要求	<ol style="list-style-type: none"><li>(1) 报告至少应该包括程序设计、程序效果展示、总结分析 3 个部分；</li><li>(2) 程序设计描述组成程序的模块、类、函数以及他们之间的相互关系，若有算法，可以描述算法流程；</li><li>(3) 程序效果展示除了程序运行效果截图之外，应该有必要的文字说明；</li><li>(4) 总结分析可以分析实现的效果与理想情况的差异，分析导致这些差异的原因，切忌不要写成心得体会；</li><li>(5) 报告应该格式规范、排版整洁、少语病和错误。</li></ol>
作业提交	<ol style="list-style-type: none"><li>(1) 含有 git 仓库（有.git 目录）的完整源代码；</li><li>(2) 任务报告。</li></ol>
评分标准	<p>按照五级制打分，分为优秀、良好、中等、及格、不及格，各评分项占总成绩的比例为：</p> <ol style="list-style-type: none"><li>(1) 任务完成情况占评分的 60%；</li><li>(2) 报告占评分的 40%。</li></ol>

	评分老师根据各部分的完成情况，直接给出总成绩。
--	-------------------------

## 1. 程序设计

本程序由课后实践 2 的代码继承而来，添加了两个源文件，以及在 distribute.h 和 distribute.cpp 对 chassis、raidar 和 smartCar 增加了方法的属性。

以下分别是新增的两个文件：Observer.h 和 Subject.h 用来实现订阅者模式。

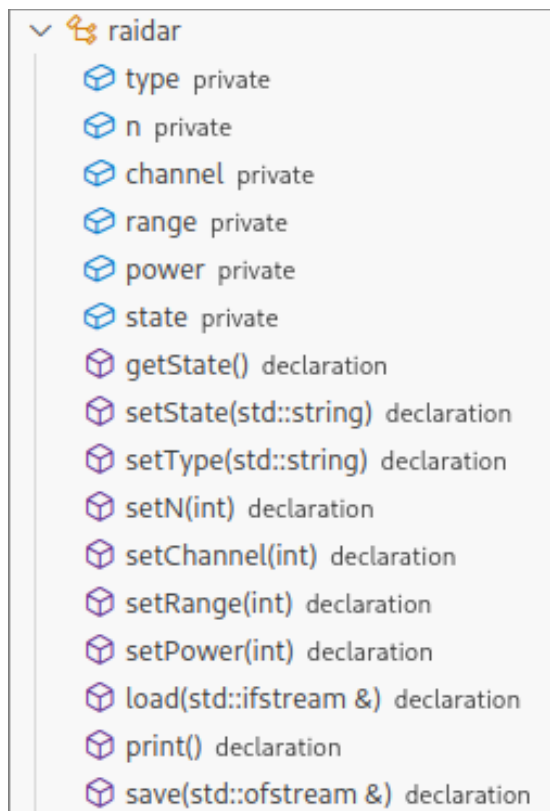


```
OBSERVER_H
Observer
update(Subject *) declaration
```



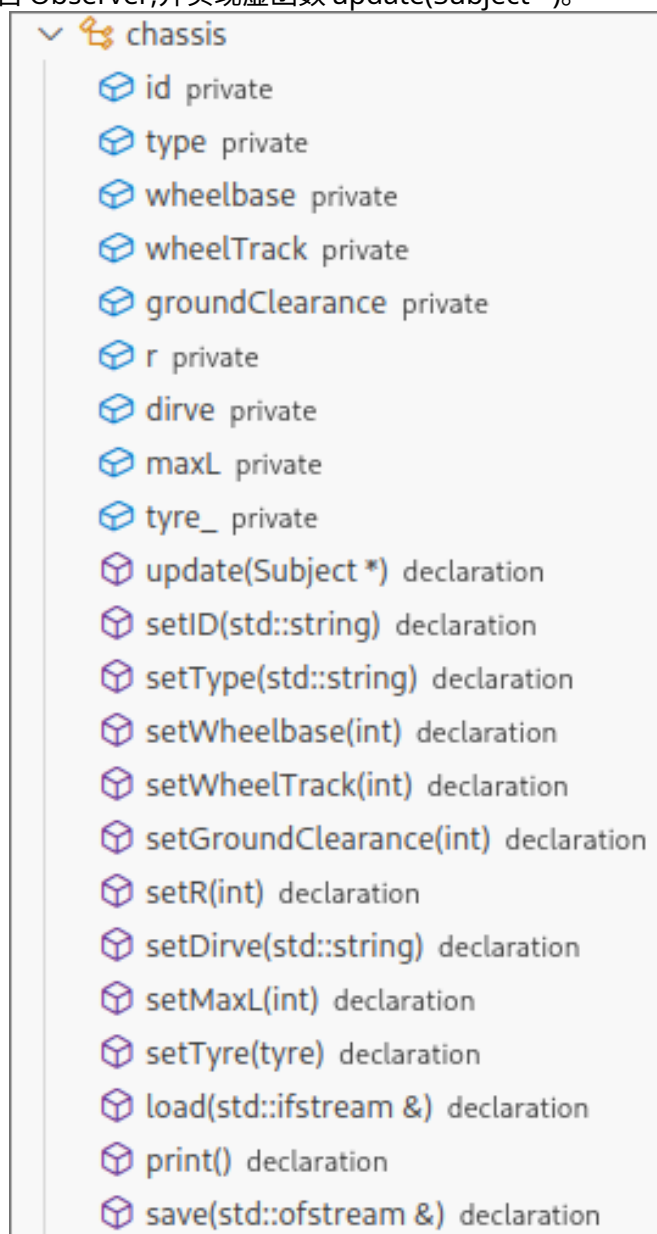
```
SUBJECT_H
Subject
attach(Observer *)
detach(Observer *)
Notify()
getState() declaration
setState(string) declaration
_observers protected
```

以下是 raidar 类新增继承自 Subject,并实现虚函数 getState()和 setState(string)，并为此添加了新的成员变量 state。

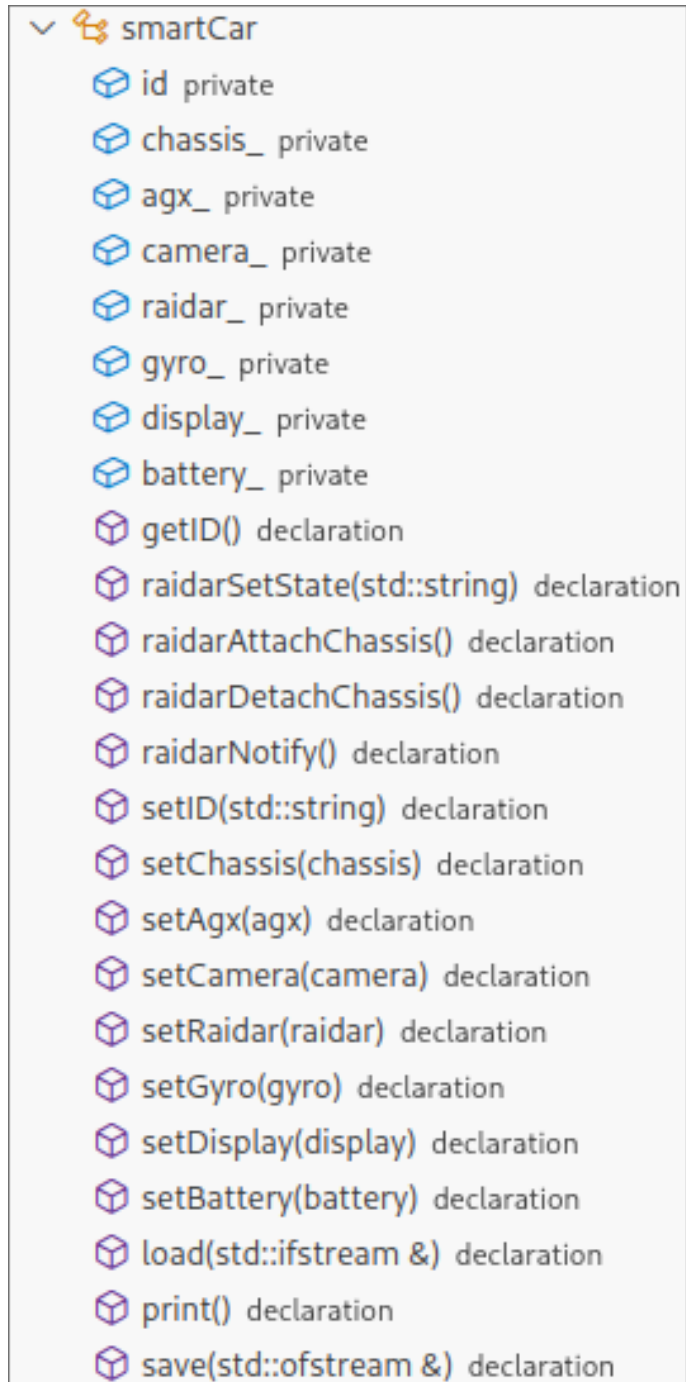


```
raidar
type private
n private
channel private
range private
power private
state private
getState() declaration
setState(std::string) declaration
setType(std::string) declaration
setN(int) declaration
setChannel(int) declaration
setRange(int) declaration
setPower(int) declaration
load(std::ifstream &) declaration
print() declaration
save(std::ofstream &) declaration
```

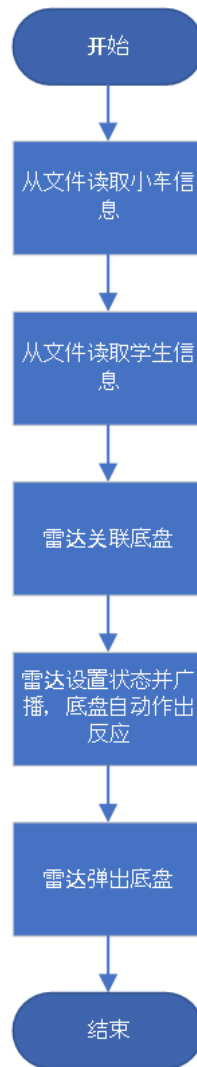
以下是 chassis 类新增继承自 Observer,并实现虚函数 update(Subject \*)。



由于各个类中的成员变量都被 `private` 属性保护了起来，外部无法访问，所以我在 `smartCar` 中添加了新方法使得可以调用雷达和底盘新增的函数，并且在同一辆车上 `attach` 也是符合现实世界和直觉得。以下是 `smartCar` 类实现函数 `raidarSetState()`、`raidarAttachChassis()`、`raidarDetachChassis()`和 `raidarNotify()`。



下图所示是程序的基本流程图，其中读取小车信息的模块是程序中的 readCar(path)函数，读取学生信息模块是 readStu(path)函数，雷达关联底盘，雷达设置状态并广播，底盘自动作出反应，雷达弹出底盘。



## 2. 程序效果展示

如下图所示，当雷达遇到障碍在前左右这三种情况时，会按照预定的策略让底盘作出反应。

```
• [zjj@eieArch build]$ /home/zjj/Documents/eie/soft3/build/eie_3_soft
The obstacle is front of me, then the chassis performs back.
The obstacle is left of me, then the chassis performs right.
The obstacle is right of me, then the chassis performs left.
```

## 3. 总结和分析

本程序基本遵守了声明和实现规范完成了代码，并已上传至 github (<https://github.com/WhateverMO/eie/tree/master/soft3>)，遵守了驼峰命名法，声明和实现按文件分开，且没有直接全局使用命名空间 std，规范了小车信息储存和学生信息储存的格式并且使用文件夹来管理各个不同的组件，文件结构简单易懂；虽然程序可以正常运行，但是没有对 smartCar 调用这些情况时的函数作出很好的优化，比较随意，都需很可能需要改动升级，更新。