NUCLEAR POWER PLANT

Statement of Work:

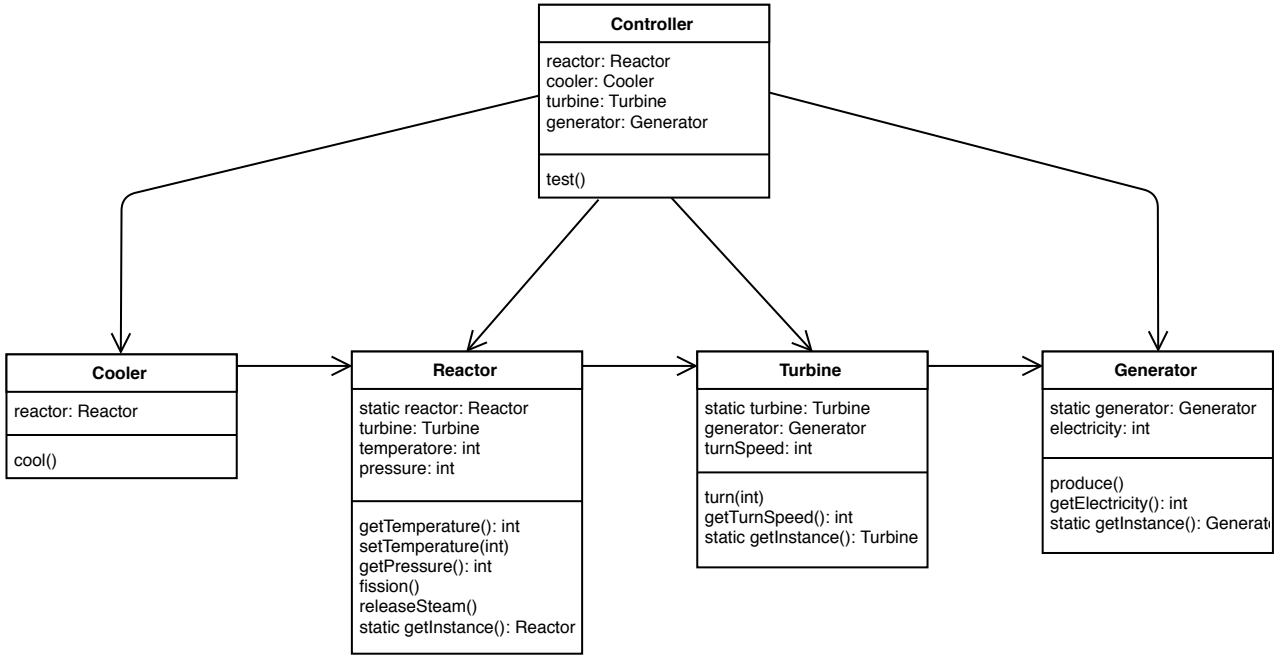
In this project, there is a nuclear power plant with a nuclear reactor, turbine, generator and the cooler of the nuclear reactor. There is also a control unit to test the reactor. The controller gives information about the test and has access to the cooler, reactor, turbine and the generator. When the chain reaction starts, the reactor cannot prevent overheating by itself. The cooler should interfere to the reactor and keep it in an optimum temperature. Since the temperature is increasing, the water pressure in the reactor is also increases. When the pressure is enough to turn the turbine, the reactor releases some amount of steam to turn the turbine. When the turbine has enough turn speed for the generator, it triggers the generator and the generator starts producing electricity. If the generator produces decent amount of electricity in a given time, then the reactor passes the test successfully. Design issue is that, there is only one reactor. In addition to this, that reactor is accessed by two different components. Same thing is also valid for turbine and the generator.

Design Patterns:

Singleton pattern is useful to solve this problem.

Since there should be exactly one reactor, turbine and generator, we can ensure them only have one instance and provide a global access point to them. To make a class singleton, we make the constructor private. Therefore only that class can make an instance of it. We create a unique instance of the class. To provide a global access, we can use a class method that returns the instance of the class.

UML:


Controller
reactor: Reactor
cooler: Cooler
turbine: Turbine
generator: Generator
test()
Cooler
reactor: Reactor
cool()
Reactor
static reactor: Reactor
turbine: Turbine
temperatore: int
pressure: int
getTemperature(): int
setTemperature(int)
getPressure(): int
fission()
releaseSteam()
static getInstance(): Reactor
Turbine
static turbine: Turbine
generator: Generator
turnSpeed: int
turn(int)
getTurnSpeed(): int
static getInstance(): Turbine
Generator
static generator: Generator
electricity: int
produce()
getElectricity(): int