任务：JProwler网络仿真架构重构重构的目标是使得整个仿真系统模块与模块之间的边界更清晰、实现模块与模块之间的松耦合，以及扩展保存网络仿真过程的功能。从而为下一阶段Flood协议仿真与实现打下基础。

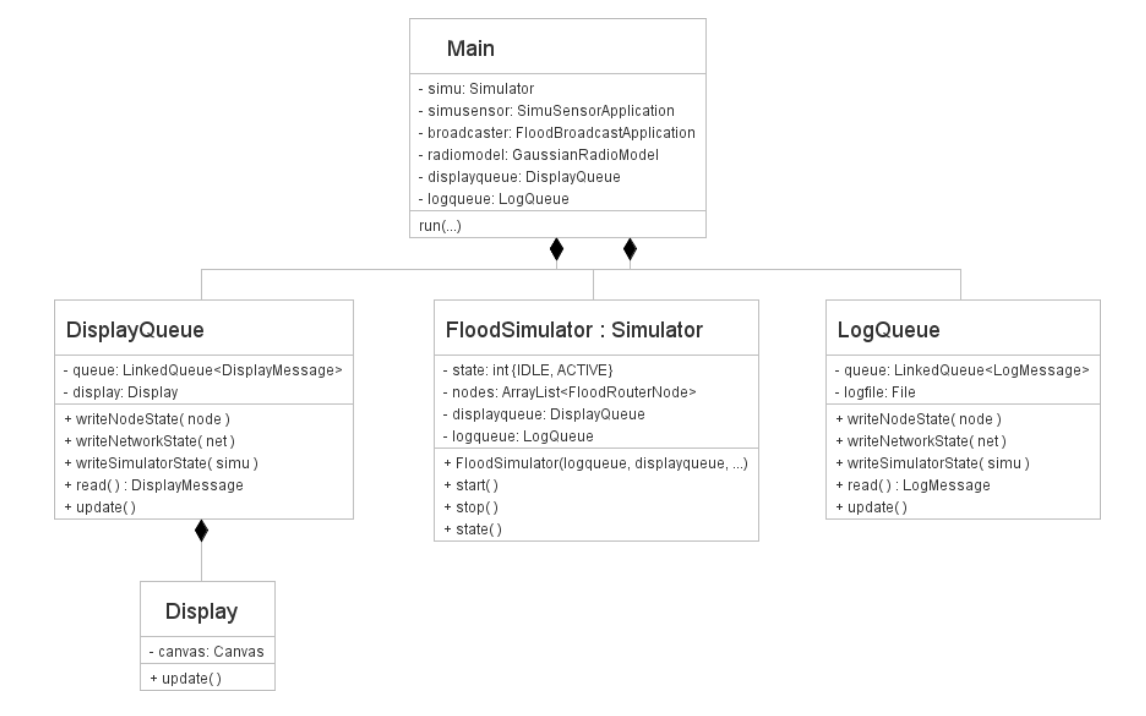
对开发Flooding协议仿真，本阶段任务并非必须，但是对避免可视化代码和仿真逻辑代码混在同一模块、提高后续开发效率、减少不必要的出错、提高开发质量很有好处。所以仍然推荐花0.5 - 1天时间完成本重构。重构时可以每做一点改进，即尝试运行。熟练的开发人员应能在0.5天内完成本次重构。

R1：在程序中增加一个 DisplayQueue类，实现Java的Queue或DeQue接口，可以从LinkedQueue或ConcurrentQueue继承或扩展得来。可以为DisplayQueue增加 writeNodeState, writeLinkState, writeNetworkState, writePacketState 等方法以代替Queue接口标准的push方法，以便于在仿真中调用。同时将存在于Node、Simulation等类中用于控制显示的代码，替换为对上述方法的调用，保持核心仿真模块职责的单一性。今后，凡是仿真引擎（核心仿真代码）需要显示时，一律采用将待显示数据推入DisplayQueue的做法，并禁止在仿真引擎中直接操纵Display类进行输出，以实现仿真过程可视化模块与仿真引擎的松耦合。

R2：在DisplayQueue中增加一个 flush 方法，在flush中读取DisplayQueue中的数据，并解释数据和更新Display显示。  
原先分散于Node等类中的直接操控Display类完成显示的代码统一收集到Monitor类中。（一个更直接更暴力的办法是将Monitor类和Display类合并，只扩展Display类）  
然后在主程序循环中增加对updateView的调用，以确保在Queue中有数据时，能够被Monitor类及时读取并显示。  
至此，重新调试JProwler，使之能像以前一样正常工作。

R3：在程序中增加一个 LogQueue 类（功能非常类似DisplayQueue），可以继承自LinkedQueue，并仿效DisplayQueue增加 writeNodeState, writeLinkState, writeNetworkState, writePacketState 等方法，便于在仿真过程中被仿真引擎代码调用，以实现仿真中各对象状态的保存（便于后续数据分析）。

R4：在LogQueue中增加一个flush方法，在flush中读取Queue中的数据，并解释数据和将其保存到文件中。  
后续可以写一段Python脚本来读取数据并进行统计。  
至此，重新调试JProwler，使之能像以前一样正常工作。



注：用于暂存显示用数据的DisplayQueue和暂存写入文件数据的LogQueue几乎完全相同，仅flush方法不同。这是为什么在课上讲的时候，只提到了一个队列且让其同时兼具DisplayQueue和LogQueue的功能。

R5：将目前 TestBroadcastNode 类中的main方法代码 extract 出来，放入一个独立的模块 Main 中，以保持 TestBroadcastNode 类职责的单一性和纯粹性。