**题目：**

开发一个计算机操作系统的线程调度程序，要求实现时间片调度和抢占调度这2种调度算法，支持Windows、Unix和Linux这3个操作系统，并且将来有可能会增加新的调度算法和支持新的操作系统，请选择恰当的设计模式解决该问题，画出类图，写出关键代码。(不考虑线程调度算法的具体代码实现)

**解答：**

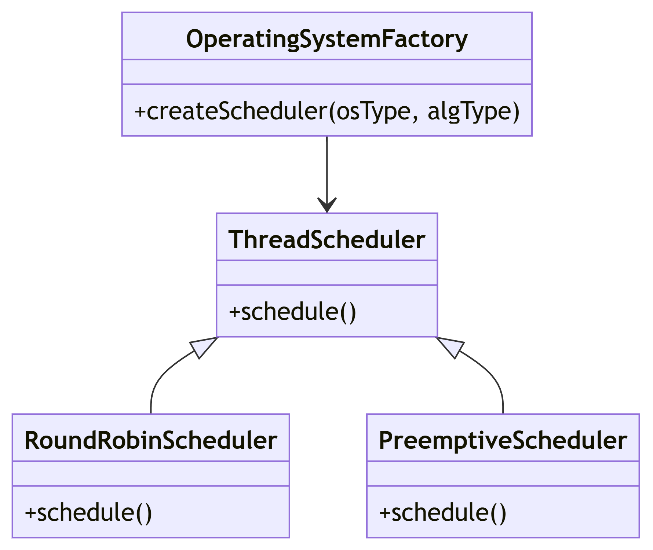
* 设计思路：

1. 可以使用**策略模式**来解决线程调度算法的扩展问题，同时使用**工厂模式**来创建适应不同操作系统的调度程序对象。
2. 策略模式将不同的调度算法封装为不同的策略类，客户端可以在运行时选择使用哪种调度算法。
3. 工厂模式根据不同的操作系统，生成相应的调度器实例。将操作系统和调度算法解耦，便于将来扩展新的操作系统和新的调度算法。

* 模式：

**策略模式**和**工厂模式**

* 类图：



|  |
| --- |
| * 核心代码： |
| from abc import ABC, abstractmethod  *# 调度器的抽象类*  class ThreadScheduler(ABC):      @abstractmethod      def schedule(self):          pass  *# 时间片调度算法*  class RoundRobinScheduler(ThreadScheduler):      def schedule(self):          print("Using Round-Robin Scheduling Algorithm")  *# 抢占式调度算法*  class PreemptiveScheduler(ThreadScheduler):      def schedule(self):          print("Using Preemptive Scheduling Algorithm")  *# 操作系统工厂类*  class OperatingSystemFactory:      @staticmethod      def createScheduler(osType: str, algType: str)-> ThreadScheduler:          if osType == 'Windows':              print("Creating Windows Scheduler")          elif osType == 'Unix':              print("Creating Unix Scheduler")          elif osType == 'Linux':              print("Creating Linux Scheduler")          if algType == 'RoundRobin':              return RoundRobinScheduler()          elif algType == 'Preemptive':              return PreemptiveScheduler()          else:              raise ValueError("Unknown scheduling algorithm")  *# 客户端代码*  def client\_code(osType: str, algType: str):      scheduler = OperatingSystemFactory.createScheduler(osType, algType)      scheduler.schedule()  *# 测试：Windows下使用时间片调度*  client\_code('Windows', 'RoundRobin')  *# 测试：Linux下使用抢占调度*  client\_code('Linux', 'Preemptive') |