* CcdpEngineMain,
  + starts and a JSON file can be passed to start processing
  + Creates a CommandInfo object with the executor jar file as well as the executor classname
  + Instantiates a new CommandExecutorInfo with that information and that is what is passed to the Scheduler
  + Starts the scheduler and passes any Thread that needs to be executing at startup
* CcdpMesosSecheduler
  + Uses the factory to create all the different objects required to run the framework
  + Starts the un-assigned resources specified in the configuration property “min.number.free.agents”
  + Adds all the Threads requests using the addTask(request) method
  + Gets the request session id. If exists and don’t have resources allocated it takes it from the free VMs and starts another free one. If free does not have one then it starts it
  + If the thread does not have a session-id allocated it assigns the public one
  + Once all the resources allocation takes place, the abstract method addRequest() implemented by the child class is invoked
* CcdpRemoteScheduler