CS 3341 FOUNDATIONS OF COMPUTING: LAB 2 FLOWSCRIPT: A VISUAL DOMAIN SPECIFIC LANGUAGE FOR PROCESS FLOW

Overview: Design and develop a domain specific language for describing process flows called FlowScript, that utilizes the DOT Language (Graph Description Language) as its base and can describe the execution dependencies of jobs within the multithreaded job system from Lab 1.

Repo Link:

Due: October 23, 2023 @ 11:59pm.

Code Requirements (40%):

- Modify the Multithreaded Job System from Lab 1 to compile as a library with a simplified interface that exposes the following functionality to end user.
    - Get list of available job types
    - Create/Destroy/Status/Complete specific Jobs
        - This should be accomplished via string interface. All Job classes will be handled by the job system.
        - The user should not be required to pass objects (jobs, classes, etc) as inputs or receive them as outputs. All job types can be identified by a unique custom string identifier ("compile", "compileParse", etc)
        - All input/output for all job types will be restricted to JSON objects, ensure all jobs in system will comply with this requirement.
    - Start/Stop/Destroy Job System
    - "Register" new jobs type with job system
        - The only exception to JSON input/output requirement for job system interface.
        - This enables a user to create its own custom job, that inherits from base Job class
            - User must create a factory pattern style class that enables job system to create new instance of the custom job class. And register this interface with the Job System via the Register method.
            - User can then register the custom job with Job System so it can be utilized.
- Must create a shared library for the job system that exposes the full Job interface described above

- Write a test program that is capable of testing all features of Job System, for multiple jobs. Test program must load job system as a shared library.
- Utilize good OOP design practices within the Job System, consider design patterns as a guide in your final implementation, even if you modify the pattern to meet your specific design needs.
- ***CS 5393 Requirement***: Must enable the ability to have job execution be dependent on completion of a separate job. This can not be hard coded, user must be able to define the job dependencies at run time to job system.
  - For example
    - JobA and JobB are created,
    - User informs Job System that JobB is dependent on JobA being complete
    - JobB will only execute once JobA completes.
- ***EXTRA CREDIT***: Make the job system capable of being asynchronous and allowing user to register a callback function that will be executed once job is complete.

Report Requirements (60%):

- Your new language, FlowScript, should be an extension of the DOT language. It will enable users to design workflows and control flows visually, while integrating the capability to script conditions, loops, and function calls.
- Identify the Programming Language Elements you feel are needed to enable the creation of the FlowScript DSL to meet the languages' objective. Provide detailed specification of these elements, why they are needed and how they are represented (syntax and semantics) in FlowScript via the DOT Language. Not all elements must be implemented,
  - Possible language elements to consider:
    - Variables, data types, operators, control structures, functions/procedures, Objects/Classes, Exceptions
- FlowState should have the following features at a minimum
  - Define process, execution conditions
  - Define specific process, by name, that will be executed
  - Define dependency order for execution of processes
  - Enable ability to show/allow parallel execution of process
  - Enable conditional call of process based upon a previous condition
    - Output of condition could enable process to execute again (loop), or could conditionally call 1 of many processes based upon condition (i.e. switch statement)
- Use any elements of the DOT language (shapes, edges, subgraphs, attributes, etc) to enable all features needed to describe all requirements.
- Create detailed sample scripts that demonstrate FlowScript features
  - Examples should show both FlowScript and how (when interpreted), the execution process of Jobs from the Job System.

    - Utilize a DOT language visualizer to show the graphical representation of DOT language examples, along with the raw dot code.
- Provide an overview of your final design of FlowScript, discussing its features, programming paradigms and language and elements description
- Keep in mind the FlowScript will need to be parsed in later labs, and your definitions and explanations in this report will be all the information provided to enable the parsing of FlowScript to drive your JobSystem.
- ***EXTRA CREDIT***: write a standalone program (in any language) that will take raw FlowState code and visualize it using a DOT Language visualizer. Example here

All code in the "Code" folder, raw data files generated in "Data" and your final report in "Report" folder in the git repo. If your project fails the compile action upon final push to git repo, it will receive a 0. Be sure to resolve any issues with the compile action prior to deadline.