# Interpreter Architectural Style

Group Name: CJ2
Members:

- Sung Wook An (sw2an)
- Weifeng Jiang (w48jiang)
- Bo Liu (b69liu)
- Jinsea Park (j54park)
- Sawim Zargar (sazargar)

### Q1:- What is interpreter architectural style? What are its components and connectors? (define) (Jinsea)

The interpreter is an architectural style that is suitable for applications in which the most appropriate language or machine for executing the solution is not directly available. The style consists of a few components which are a program that we are trying to run, interpreter that we are trying to interpret, current state of the program and the interpreter and finally memory component to hold the program, the current state of the program and the current state of the interpreter. The connectors for interpreter architectural style is procedure calls to communicate between the components and direct memory accesses to access memory.

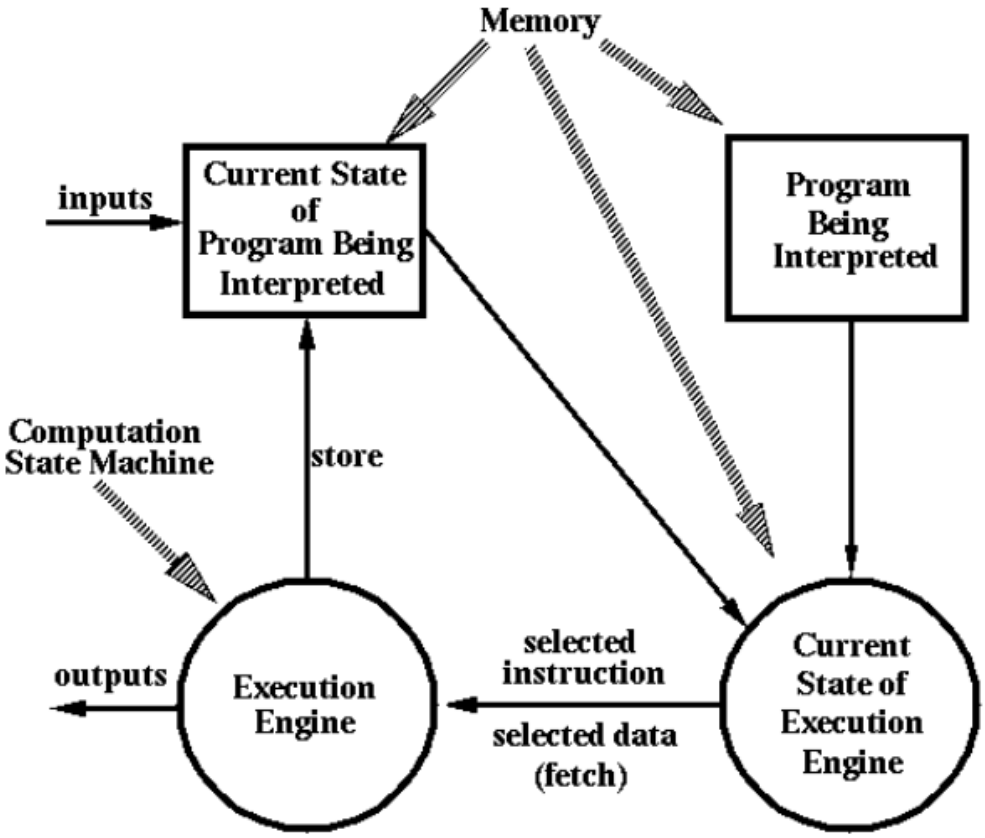
### Q2: - Impose specific topological constraints? (diagram) (Weifeng Present)

The interpreter has 4 compositions:

- An interpreter engine: finishes the interpreter work
- Data store field: contains the pseudo code
- A data structure: records the current state of the interpreter engine
- Another data structure: records the progress of the interpreted source code

Input: Input to the interpreted program component is sent to the program state, where it is read by the program running on the interpreter

Output: Program output is placed in the program state, where it can result in output from the interpreted program component's interface




---

**Q3: - Most applicable to specific kinds of problems? (Calvin)**
Programming Language Compilers: Java Smalltalk
Rule Based Systems: Prolog Coral
Scripting Languages: Awk Perl
Micro coded machine: Implement machine code in software.
Cash register / calculator: Emulate a clever chip using a cheap one.
Database plan: The database engine interprets the plan.
Presentation package : Display a graph, by operating on the graph.

**Q4: - Engender specific kinds of change resilience? (advantage) (Sawim)**
There are some distinct advantages of using an interpreter architecture style:

- By having the behaviour of the system defined by a custom language or data structure, software development becomes easier
- This facilitates the portability and flexibility of application or languages across various platforms
- Allows us to simulate non-implementable hardware which keeps costs of hardware affordable
- As each line is interpreted, the results of the execution are visible which makes debugging easier
- Errors are caught as they happen since the interpreter stops when it can't interpret a line

- Interpreting syntax is also faster and uses less memory than compiling syntax and then executing it as the latter requires creating and storing the machine code which is then executed whereas the interpreter just deals with the source code itself

**Q5: - Have any specific negative behaviours? (disadvantage) (Sawim)**

There are some distinct disadvantages to using an interpreter architecture style:

- Since there is no intermediate language conversion step which is stored to be executed, there is no option to optimize the code at a lower level as is done in a compiler
- Extra level of indirection make executing code using an interpreted slower than executing compiled code - this is because the interpreter must evaluate the source code every time whereas the compiled code is in machine code and can be executed very fast

**Q6: - Support/inhibit specific NFPs?(Caleb)**

Portability and flexibility of application or languages across various platforms
Virtualization. Machine code intended for one hardware architecture can be run on another using a virtual machine, which is essentially an interpreter
Behaviour of system defined by a custom language or data structure; makes software easier to develop and understand.
Supports dynamic change(Efficiency)
the interpreter usually just needs to translate the code being worked on to an intermediate representation (or not translate it at all), thus requiring much less time before the changes can be tested.
“Sandbox” security
An interpreter or virtual machine is not compelled to actually execute all the instructions the source code it is processing. In particular, it can refuse to execute code that violates any security constraints it is operating under.
For example, JS-Interpreter is a sandboxed JavaScript interpreter written in JavaScript. It allows for execution of arbitrary JavaScript code line by line. Execution is completely isolated from the main JavaScript environment. Multiple instances of the JS-Interpreter allow for multi-threaded concurrent JavaScript without the use of Web Workers.

**EXAMPLE:**

In our example, we’ll first start by quickly explaining how a compiled style works. Then to illustrate the interpreted style we’ll quickly identify each other as the different components. We’ll walk through a code example to show how each line is interpreted and how by changing part of the syntax, the interpreter and output are affected.