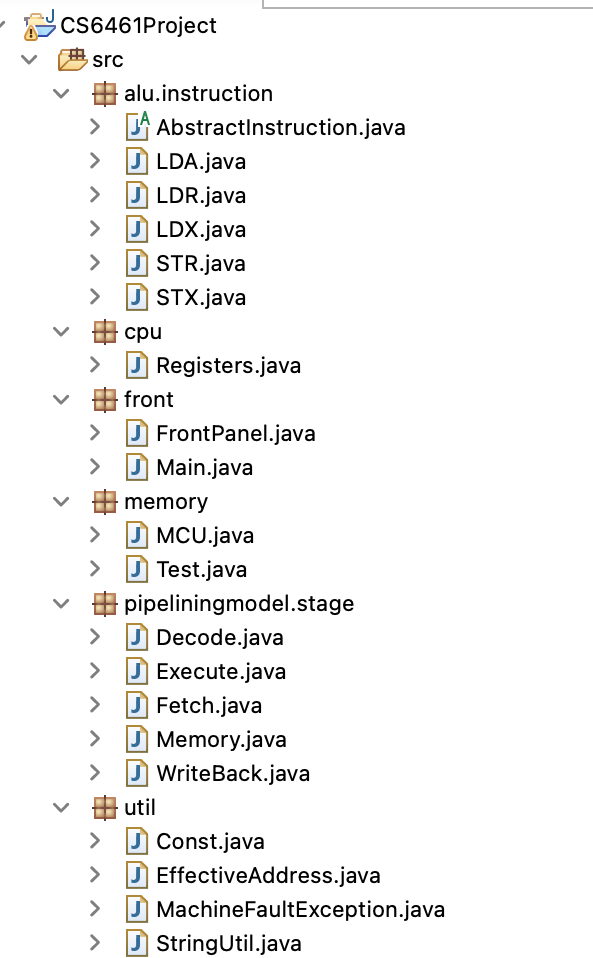
**Design Notes**

The file structure of the designed simulator as at the time of submission for the first phase is illustrated in figure 1 below. This is followed by a brief description of the constituent files.



**Figure 1: File structure for phase-1 of simulator project**

1. Src
   1. alu
      1. Instructions

* **AbstarctInstructions.java**

Contains class AbstractInstruction that is responsible for returning execution details of the machine. To ensure that all valid operations are proceeding expectedly.

* **LDA.java**

Containsclass LDA that facilitates execution of the LDA instruction. Constitutes methods such as execute and String getExecuteMessage() to ensure instruction execution.

* **LDR. Java**

Containsclass LDR that facilitates the execution of the instruction LDR. Constitutes methods such as execute and String getExecuteMessage() to ensure instruction execution.

* **LDX.java**

Containsclass LDX that facilitates the execution of the instruction LDX. Constitutes methods such as execute and String getExecuteMessage() to ensure instruction execution.

* **STR.java**

Containsclass STR that facilitates the execution of the instruction STR. Constitutes methods such as execute and String getExecuteMessage() to ensure instruction execution.

* **STX.java**

Containsclass STX that facilitates the execution of the instruction STX. Constitutes methods such as execute and String getExecuteMessage() to ensure instruction execution.

* 1. cpu
* **Registers.java** Containsclass Registers that Initialises all register variables. Constitutes functions that help in storage and retrieval of values stored in registers. Also has functions that help display register values in different base systems.
  1. Front
* **FrontPanel.java**

Has class FrontPanel that contains set up of the GUI of the simulator. Constitutes a constructor to set up initial state of the simulator. Also contain functions that execute when various front panel buttons are pressed to extract suitable functionality from the simulator for instance - function runInstructions(), function printConsole(), etc.

* **Main.java**

Containsclass Main that driver code to initialise an instance of the front panel of the simulator.

* 1. Memory
* **MCU.java**

Contains class MCU that declares the memory of the simulator in the form of array lists and constitutes functions that help in storing and fetching from their “memory” as implemented using arrays.

* **Test.java**

Made up of class Test that performs some trivial extra functions.

* 1. Pipeliningmodel
     1. Stage
* **Decode.java**

Made up of class Decode that given a particular instruction thread, decodes it into operations to be performed.

* **Execute.java**

Made up of class Execute that given an instruction, sets into motion the execution functions.

* **Fetch.java**

Has class Fetch to help import suitable values stored in buffers to be executed.

* **Memory.java**

Constitutes class Memory that facilitates memory manipulation while instruction is being executed.

* **WriteBack.java**

Constitutes class WriteBack that stores manipulated values into suitable locations.

* 1. Util:

Constitutes of classes such as const.java, EffectiveAddress.java, Machine FaultException.java etc.

1. Jar file: The file to be executed to launch the simulator