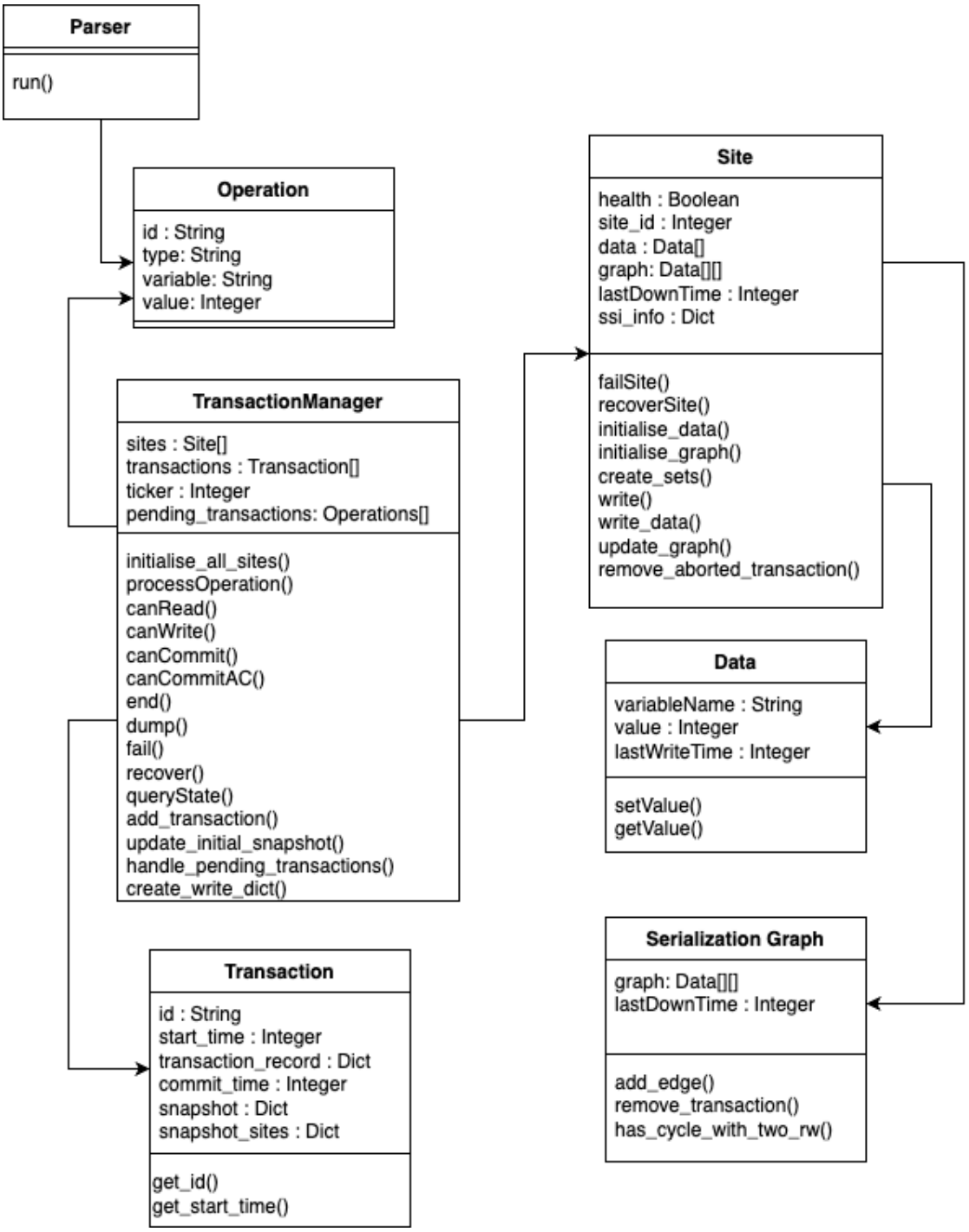
CSCI-GA 2434 Fall 2024
Sarthak Khandelwal sk12184
Anand Trehan at6404
December 6, 2024

---

1. **Design Diagram**



2. **Description**

- Parser
  - This Module is responsible for parsing the user inputs from file. Its output is the operation object which is consumed by the transaction manager.
- Operation
  - This module stores information about each operation that must be completed. Instances of Operation get created by the parser and they stores parameters like transaction number, transaction type, variable and value. This is used by the transaction manager for further processing.
- Transaction
  - Similar to operation this module stores transaction ID and start time. It also stores a record of all its operations and a snapshot of the data as of its start time. Instances of Transaction get created by the transaction manager when it encounters a "begin()" input.
- Transaction Manager
  - This is the module that links all components together. Firstly it stores all the instances of site and transaction and has their initialisation methods. Each instance of operation gets processed using the processOperation() method which creates transaction objects.
  - For begin(), it creates a new instance of Transaction and stores it. This can be referenced by other operations later on.
  - For read operations we have the canRead() method which determines if a read is possible and if so returns the value of the read otherwise it either aborts or makes the transaction wait.
  - For write operations we have the canWrite() checks if a write is possible and if not either aborts or makes it wait.
  - When there is an end() operation, the TM calls the canCommit() method, which checks each site for an RW edge and if all sites are good then the commit goes through. It also calls canCommitAC() which checks if the commit may need to be aborted because of available copies reasons.
  - fail() and recover() are responsible to bring down and bring up sites. Upon recover() we also update snapshots and handle pending transactions.
  - handle_pending_transactions() reprocesses all the operations that are in waiting state.
  - update_initial_snapshot() picks up all available values and stores it in the transaction snapshot for read operations
  - dump() returns the committed values of all copies of all variables at all sites, sorted per site with all values per site in ascending order by variable.
- Site
  - In addition to storing the variables and their values, each site has a health flag to indicate whether it's down or not, a serialization graph with all active operations affecting that site and each site also stores the last time it had failed.
  - failSite() and recoverSite() are responsible to update the health flags and lastDownTime for each site.

  - create_sets() checks each transactions record and separates out the read and write operations into sets.
  - updateGraph() is responsible to adding/removing edges in the serialization graph and to decide if a transaction can be added to a graph or not.
  - Write() is called when a transaction is green-lit to be committed. It updates the value of the variable and updates the graph.
  - remove_aborted_transactions() is called when a transaction is either aborted or if it's completed and can be removed from the graphs at all sites.
- Data
  - Each data point in site has a name, value and it's last committed time.
- Serialization Graph
  - Each site has a serialization graph object. It has methods to add edges, detect a cycle with 2 rw edges and remove a tranaction i.e remove a vertex.