Name: Antoine J. Smith Jr.

Assignment: Project 3

Course: CSIS 3810

Scope/Overview:

* Introduction
  + The purpose of this project is to design a simulated Bandwidth Cache that has ten destination addresses and interacts with ten distinct decision threads. These ten threads are competing with each other for access to all ten slots within the cache.
* Solution
  + A semaphore with one permit was used to control each threads’ access to the Bandwidth Cache. Additionally, the two primitive data types that were implemented are acquire and release. Which would obtain and release the lock for a smooth data flow within the system.
* Criteria Establishment
  + All ten threads will perform an interaction with the Bandwidth Cache in every given round. Furthermore, each thread pings an initial address ranging from 1 – 100 in order to locate a position within the Cache. If this method fails, then there is a backup procedure, which is to then ping an address ranging from 1 – 10. Then, if this fails, there will either be an eviction and replacement of a slot within the Cache or to fill up an empty spot within the Cache. Also, during any stage of this process, and a successful search was made, then both the address and its bandwidth value will then be displayed to the user inside of the command line.
* Criteria Validation
  + The process was tested using print statements all throughout the program. This was very useful in order to see what exactly was occurring inside of the cache at each stage and by seeing each threads interaction method chosen while interacting with cache.

Data Design:

* Semaphores (Java Built-in tool)
  + Term
    - Abstract data type used to control access to a resource
      * In this project, the resource is the Bandwidth Cache
  + Discussion
    - As previously stated, to control the threads access to the Cache
* Addresses
  + This data type in the program was defined as its own integer array of 10 addresses that contain address values of 1 – 100
  + Defined this way because Java does not have a built-in tuple functionality.
* BandwidthCache
  + This data type in the program was defined as the Bandwidth Cache, which is a integer array. Additionally, the address array is indexes corresponds to the Bandwidth Cache is indexes within its array.
  + Therefore, whenever one of the threads would query an address between 1 – 100, it would search for that address within the address array. Once it is found that corresponding index will then be used inside of the Bandwidth Cache array to return the Bandwidth Value for display to the user.
* Random & Math.Random (Java Built in Function)
  + This built-in function was very useful for giving random address numbers, bandwidth values, decision thread queries, and on how many addresses and values will start off inside of the cache at the very beginning of the program/system.
* Thread.sleep (Java Built in Function)
  + Useful for pauses within the system.
* Rounds
  + Useful for knowing how many overall interactions have been completed between the threads and the cache.
* Output
  + This is all of the return values to the user. Such as the following:
    - Bandwidth Values
    - Bandwidth Addresses
    - Decision Thread Being Implemented
    - Current Round
    - Instructions on how the Addresses are defined in terms of if it has been initialized and if that spot is empty or not.

Architectural Design:

* Initialization
  + Partially initializes the Bandwidth cache, so all methods within the program are able to be executed.
    - Therefore, a certain number of addresses are initialized & bandwidth values
* Decision Threads
  + Initial
    - All ten threads are initialized with the integer value of 0
  + Then
    - All of threads query a random integer between 1 – 100
      * Success 🡪 Returns Address and Bandwidth Value
      * Failed 🡪 Queries an integer between 1 – 10
        + Success

Returns Address and Bandwidth Value

* + - * + Failed

Fills in an empty slot

Or evicts a random slot

* Termination
  + Shutdown after final round

Classes and Object Design:

* Main Class (Class)
  + This is the application presented to the user
* BandwidthCacheSystem (Class)
  + All of the logic is located within this class
* DecisionThreadsQueries (Method)
  + Used to have all ten threads query an address between 1 – 100
* BandwidthValue (Method)
  + Returns a random value between 1 - 1000 to be put in the bandwidth cache as a bandwidth value
* DecisionThreadBackupQuery (Method)
  + Returns a random value between 1 – 10 to be used as the new query value for searching inside of the bandwidth cache
* AddressCacheInitialization (Method)
  + Initializes some of the Bandwidth Cache is values and addresses
* UpdateAddress (Method)
  + Returns a new address value between 1 – 100 to update the address of a particular slot in the cache
* EvictionIndex (Method)
  + Returns a random index between 0 – 9 to be used for evicting a particular slot within the cache
* DecisionThreadQueryFound (Method)
  + Returns an index between 0 and 9 if the query was a success. Otherwise, returns a negative 1.
* EmptySlotinCache (Method)
  + Same as DecisionThreadQueryFound Method 🡪 However, it is looking to see if a slot is empty
* ThreadBandwidthInteraction (Method)
  + Used for the thread and cache interactions.

Test:

* Making sure every thread interacts with the cache during each round properly. Used several print statements to verify this.

Command (Running Code from Command Line):

* Run the Following:
  + Javac \*.java
    - Directory 🡪 /company
  + Java com/company/Main.java
    - Directory 🡪 /src