Documentation for Java Classes in Multi-Core Process Scheduler

1. Process.java

Class: Process
Description: Represents a process with a unique ID, instructions, burst time, program counter, and memory bounds.

Methods:

- Process createProcess(int processID, List<String> instructions, int programCounter, int[] memoryBounds, int burstTime): Creates a new process.

- void decreaseBurstTime(int n): Decreases the burst time by a specified amount.

- int getProcessID(): Returns the process ID.

- List<String> getInstructions(): Returns the list of instructions.

- int getProgramCounter(): Returns the current program counter.

- int[] getMemoryBounds(): Returns the memory bounds.

- int getBurstTime(): Returns the burst time.

- String getNextInstruction(): Retrieves the next instruction to execute.

- boolean isComplete(): Checks if the process is complete.

Usage Example:

```
List<String> instructions = Arrays.asList("assign x = 5", "print x");
int[] memoryBounds = {0, 100};
Process process = Process.createProcess(1, instructions, 0, memoryBounds,
50);
process.decreaseBurstTime(10);
System.out.println("Process ID: " + process.getProcessID());
```

2. SlaveCore.java

Class: SlaveCore
Description: Represents a slave core responsible for executing processes.

Methods:

- int getCoreID(): Returns the core ID.

- boolean isBusy(): Checks if the core is busy.

- void executeProcess(Process process, Runnable onComplete): Executes a process asynchronously.

- private void executeInstruction(String instruction): Executes a single instruction.

- private String categorizeInstruction(String instruction): Categorizes the instruction type.

- private void handleAssignment(String instruction): Handles assignment instructions.

- private void handlePrint(String instruction): Handles print instructions.

- private void handleArithmetic(String instruction): Handles arithmetic operations.

Usage Example:

```
SharedMemory sharedMemory = new SharedMemory();
SlaveCore slaveCore = new SlaveCore(1, sharedMemory);
List<String> instructions = Arrays.asList("assign x = 5", "print x");
Process process = Process.createProcess(1, instructions, 0, new int[]{0,
100}, 50);
slaveCore.executeProcess(process, () -> System.out.println("Process
completed"));
```

3. MasterCore.java

Class: MasterCore
Description: Controls the overall execution of processes using slave cores.

Methods:

- MasterCore(List<File> programFiles): Constructor that initializes shared memory, slave cores, and processes.

- private List<Process> parsePrograms(List<File> programFiles): Parses program files to create processes.

- void startExecution(): Starts the execution and scheduling of processes.

Usage Example:

```
List<File> programFiles = List.of(new File("program_1.txt"), new
File("program_2.txt"));
MasterCore masterCore = new MasterCore(programFiles);
masterCore.startExecution();
```

4. ExecutionLogger.java

Class: ExecutionLogger
Description: Logs the execution status of processes.

Methods:

- ExecutionLogger(): Constructor that initializes the log list.

- void AddLog(String status): Adds a log entry.

- void printLogs(): Prints all log entries.

- List<String> getLogs(): Returns the list of logs.

Usage Example:

```
ExecutionLogger logger = new ExecutionLogger();
logger.AddLog("Process started");
logger.printLogs();
```

5. InstructionParser.java

Class: InstructionParser
Description: Parses instructions from a file to create processes.

Methods:

- InstructionParser(File inputFile): Constructor that takes an input file.

- List<Process> parseInstructions(): Parses the instructions from the file.

- private int calculateBurstTime(List<String> instructions): Calculates the burst time based on instructions.

- public String categorizeInstruction(String instruction): Categorizes the instruction type.

Usage Example:

```
File file = new File("program.txt");
InstructionParser parser = new InstructionParser(file);
List<Process> processes = parser.parseInstructions();
for (Process process : processes) {
    System.out.println("Process ID: " + process.getProcessID());
}
```

6. Scheduler.java

Class: Scheduler
Description: Manages the scheduling of processes across slave cores.

Methods:

- Scheduler(List<Process> processes, SlaveCore slaveCore1, SlaveCore slaveCore2): Constructor that initializes the process queue and slave cores.

- boolean isQueueEmpty(): Checks if the process queue is empty.

- void startScheduling(): Starts the scheduling of processes.

Usage Example:

```
List<Process> processes = new ArrayList<>();
SlaveCore core1 = new SlaveCore(1, new SharedMemory());
SlaveCore core2 = new SlaveCore(2, new SharedMemory());
Scheduler scheduler = new Scheduler(processes, core1, core2);
scheduler.startScheduling();
```

7.SharedMemory.java

Class: SharedMemory
Description: Manages shared memory using a concurrent hash map to store and retrieve variables.

Methods:

-SharedMemory(): Constructor that initializes the memory map.

- synchronized Object read(String name): Reads a variable's value from memory.

- synchronized void write(String name, Object value): Writes a variable's value to memory.

- synchronized void remove(String name): Removes a variable from memory.

- synchronized String getState(): Returns the current state of the memory map.

Usage Example:

```
SharedMemory sharedMemory = new SharedMemory();
sharedMemory.write("x", 10);
System.out.println("Value of x: " + sharedMemory.read("x"));
sharedMemory.remove("x");
System.out.println("Shared Memory State: " + sharedMemory.getState());
```