import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

class MemoryBlock {

int size;

boolean isAllocated;

String processName; // To store the name of the allocated process

MemoryBlock(int size) {

this.size = size;

this.isAllocated = false;

this.processName = ""; // Initialize with empty string

}

}

class MemoryManager {

private List<MemoryBlock> memoryBlocks;

public MemoryManager(List<MemoryBlock> memoryBlocks) {

this.memoryBlocks = memoryBlocks;

}

public void worstFit(String processName, int requestSize) {

MemoryBlock worstBlock = null;

// Find the worst fit block

for (MemoryBlock block : memoryBlocks) {

if (!block.isAllocated && block.size >= requestSize) {

if (worstBlock == null || block.size > worstBlock.size) {

worstBlock = block;

}

}

}

// Allocate the worst block found

if (worstBlock != null) {

worstBlock.isAllocated = true;

worstBlock.processName = processName; // Assign process name to the block

System.out.printf("| %-10s | %-10d | %-10s |\n", processName, requestSize, "Allocated");

} else {

System.out.printf("| %-10s | %-10d | %-10s |\n", processName, requestSize, "Not Allocated");

}

}

public void displayMemoryStatus() {

System.out.println("\nMemory Blocks Status:");

System.out.printf("| %-10s | %-12s |\n", "Block Size", "Process");

System.out.println("|------------|--------------|");

for (MemoryBlock block : memoryBlocks) {

System.out.printf("| %-10d | %-12s |\n", block.size, block.isAllocated ? block.processName : "Free");

}

}

}

public class WorstFit {

public static void main(String[] args) {

List<MemoryBlock> memoryBlocks = new ArrayList<>();

Scanner scanner = new Scanner(System.in);

// Input memory block sizes

System.out.print("Enter the number of memory blocks: ");

int numBlocks = scanner.nextInt();

for (int i = 0; i < numBlocks; i++) {

System.out.print("Enter size of memory block " + (i + 1) + ": ");

int size = scanner.nextInt();

memoryBlocks.add(new MemoryBlock(size));

}

MemoryManager memoryManager = new MemoryManager(memoryBlocks);

while (true) {

System.out.print("\nEnter process name (e.g., p1) or 'exit' to quit: ");

String processName = scanner.next();

if (processName.equalsIgnoreCase("exit")) break;

System.out.print("Enter the size of memory request: ");

int requestSize = scanner.nextInt();

memoryManager.worstFit(processName, requestSize);

memoryManager.displayMemoryStatus();

}

scanner.close();

}

}

---------OUTPUT WorstFit--------

student@student:~$ javac WorstFit.java

student@student:~$ java WorstFit

Enter the number of memory blocks: 3

Enter size of memory block 1: 90

Enter size of memory block 2: 400

Enter size of memory block 3: 230

Enter process name (e.g., p1) or 'exit' to quit: p1

Enter the size of memory request: 115

| p1 | 115 | Allocated |

Memory Blocks Status:

| Block Size | Process |

|------------|--------------|

| 90 | Free |

| 400 | p1 |

| 230 | Free |

Enter process name (e.g., p1) or 'exit' to quit: p2

Enter the size of memory request: 300

| p2 | 300 | Not Allocated |

Memory Blocks Status:

| Block Size | Process |

|------------|--------------|

| 90 | Free |

| 400 | p1 |

| 230 | Free |

Enter process name (e.g., p1) or 'exit' to quit: p3

Enter the size of memory request: 100

| p3 | 100 | Allocated |

Memory Blocks Status:

| Block Size | Process |

|------------|--------------|

| 90 | Free |

| 400 | p1 |

| 230 | p3 |

Enter process name (e.g., p1) or 'exit' to quit: p4

Enter the size of memory request: 115

| p4 | 115 | Not Allocated |

Memory Blocks Status:

| Block Size | Process |

|------------|--------------|

| 90 | Free |

| 400 | p1 |

| 230 | p3 |