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;

private int lastAllocatedIndex; // Track the last allocated index

public MemoryManager(List<MemoryBlock> memoryBlocks) {

this.memoryBlocks = memoryBlocks;

this.lastAllocatedIndex = 0; // Initialize to the start

}

public void nextFit(String processName, int requestSize) {

// Start searching from the last allocated index

for (int i = lastAllocatedIndex; i < memoryBlocks.size(); i++) {

MemoryBlock block = memoryBlocks.get(i);

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

block.isAllocated = true;

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

lastAllocatedIndex = (i + 1) % memoryBlocks.size(); // Update the last allocated index

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

return; // Exit after allocation

}

}

// If no suitable block found, wrap around and check from the beginning

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

MemoryBlock block = memoryBlocks.get(i);

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

block.isAllocated = true;

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

lastAllocatedIndex = (i + 1) % memoryBlocks.size(); // Update the last allocated index

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

return; // Exit after allocation

}

}

// If still no suitable block found

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 NextFit {

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.nextFit(processName, requestSize);

memoryManager.displayMemoryStatus();

}

scanner.close();

}

}

-----OUTPUT Nextfit---------

student@student:~$ javac NextFit.java

student@student:~$ java NextFit

Enter the number of memory blocks: 3

Enter size of memory block 1: 450

Enter size of memory block 2: 50

Enter size of memory block 3: 800

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

Enter the size of memory request: 570

| p1 | 570 | Allocated |

Memory Blocks Status:

| Block Size | Process |

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

| 450 | Free |

| 50 | Free |

| 800 | p1 |

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

Enter the size of memory request: 10

| p2 | 10 | Allocated |

Memory Blocks Status:

| Block Size | Process |

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

| 450 | p2 |

| 50 | Free |

| 800 | p1 |

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

Enter the size of memory request: 60

| p3 | 60 | Not Allocated |

Memory Blocks Status:

| Block Size | Process |

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

| 450 | p2 |

| 50 | Free |

| 800 | p1 |