Practical No : 07
Name : Priya Sanjay Nandewar
Roll No : 13266
Batch : B3

Program :

```
import java.util.*;

class PageReplacement {

    static void fifo(int pages[], int capacity) {
        Set<Integer> memory = new HashSet<>();
        Queue<Integer> fifoQueue = new LinkedList<>();
        int pageFaults = 0, pageHits = 0;

        for (int page : pages) {
            if (!memory.contains(page)) {
                if (memory.size() == capacity) {
                    int oldest = fifoQueue.poll();
                    memory.remove(oldest);
                }
                memory.add(page);
                fifoQueue.add(page);
                pageFaults++;
            } else {
                pageHits++;
            }
            System.out.println("FIFO Memory State: " + fifoQueue);
        }

        double hitRatio = (double) pageHits / pages.length;
        double faultRatio = (double) pageFaults / pages.length;

        System.out.println("FIFO Page Faults: " + pageFaults);
        System.out.println("FIFO Page Hits: " + pageHits);
        System.out.printf("FIFO Page Hit Ratio: %.2f\n", hitRatio);
        System.out.printf("FIFO Page Fault Ratio: %.2f\n\n", faultRatio);
    }

    static void lru(int pages[], int capacity) {
        Set<Integer> memory = new HashSet<>();
        Map<Integer, Integer> indexes = new HashMap<>();
        int pageFaults = 0, pageHits = 0;

        for (int i = 0; i < pages.length; i++) {
            int page = pages[i];
            if (!memory.contains(page)) {
                if (memory.size() == capacity) {
                    int lruPage = Collections.min(indexes.entrySet(), 
Map.Entry.comparingByValue()).getKey();
                    memory.remove(lruPage);
                    indexes.remove(lruPage);
                }
                memory.add(page);
                pageFaults++;
            } else {
                pageHits++;
            }
            indexes.put(page, i);
            System.out.println("LRU Memory State: " + memory);
        }

        double hitRatio = (double) pageHits / pages.length;
        double faultRatio = (double) pageFaults / pages.length;

        System.out.println("LRU Page Faults: " + pageFaults);
        System.out.println("LRU Page Hits: " + pageHits);
        System.out.printf("LRU Page Hit Ratio: %.2f\n", hitRatio);
        System.out.printf("LRU Page Fault Ratio: %.2f\n\n", faultRatio);
    }

    static void optimal(int pages[], int capacity) {
        Set<Integer> memory = new HashSet<>();
        int pageFaults = 0, pageHits = 0;

        for (int i = 0; i < pages.length; i++) {
            int page = pages[i];
            if (!memory.contains(page)) {
                if (memory.size() == capacity) {
                    int farthest = -1, pageToReplace = -1;
                    for (int memPage : memory) {
                        int j;
                        for (j = i + 1; j < pages.length; j++) {
                            if (pages[j] == memPage) break;
                        }
                        if (j == pages.length) {
                            pageToReplace = memPage;
                            break;
                        }
                        if (j > farthest) {
                            farthest = j;
                            pageToReplace = memPage;
                        }
                    }
                    memory.remove(pageToReplace);
                }
                memory.add(page);
                pageFaults++;
            } else {
                pageHits++;
            }
            System.out.println("Optimal Memory State: " + memory);
        }

        double hitRatio = (double) pageHits / pages.length;
        double faultRatio = (double) pageFaults / pages.length;
        System.out.println("Optimal Page Faults: " + pageFaults);
        System.out.println("Optimal Page Hits: " + pageHits);
        System.out.printf("Optimal Page Hit Ratio: %.2f\n", hitRatio);
        System.out.printf("Optimal Page Fault Ratio: %.2f\n\n", faultRatio);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.print("Enter number of pages: ");
        int n = sc.nextInt();
        int[] pages = new int[n];
        System.out.println("Enter page reference string:");
        for (int i = 0; i < n; i++) {
            pages[i] = sc.nextInt();
        }

        System.out.print("Enter number of frames: ");
        int capacity = sc.nextInt();

        System.out.println("\n--- FIFO Page Replacement ---");
        fifo(pages, capacity);

        System.out.println("--- LRU Page Replacement ---");
        lru(pages, capacity);

        System.out.println("--- Optimal Page Replacement ---");
        optimal(pages, capacity);

        sc.close();
    }
}
```

```
OUTPUT :=== Enter number of pages: 12
Enter page reference string:
1 3 0 3 5 6 3 4 2 1 3 0
Enter number of frames: 4

--- FIFO Page Replacement ---
FIFO Memory State: [1]
FIFO Memory State: [1, 3]
FIFO Memory State: [1, 3, 0]
FIFO Memory State: [1, 3, 0]
FIFO Memory State: [1, 3, 0, 5]
FIFO Memory State: [3, 0, 5, 6]
FIFO Memory State: [3, 0, 5, 6]
FIFO Memory State: [0, 5, 6, 4]
FIFO Memory State: [5, 6, 4, 2]
FIFO Memory State: [6, 4, 2, 1]
FIFO Memory State: [4, 2, 1, 3]
FIFO Memory State: [2, 1, 3, 0]
FIFO Page Faults: 10
FIFO Page Hits: 2
FIFO Page Hit Ratio: 0.17
FIFO Page Fault Ratio: 0.83

--- LRU Page Replacement ---
LRU Memory State: [1]
LRU Memory State: [1, 3]
LRU Memory State: [0, 1, 3]
LRU Memory State: [0, 1, 3]
LRU Memory State: [0, 1, 3, 5]
LRU Memory State: [0, 3, 5, 6]
LRU Memory State: [0, 3, 5, 6]
LRU Memory State: [3, 4, 5, 6]
LRU Memory State: [2, 3, 4, 6]
LRU Memory State: [1, 2, 3, 4]
LRU Memory State: [1, 2, 3, 4]
LRU Memory State: [0, 1, 2, 3]
LRU Page Faults: 9
LRU Page Hits: 3
LRU Page Hit Ratio: 0.25
LRU Page Fault Ratio: 0.75

--- Optimal Page Replacement ---
Optimal Memory State: [1]
Optimal Memory State: [1, 3]
Optimal Memory State: [0, 1, 3]
Optimal Memory State: [0, 1, 3]
Optimal Memory State: [0, 1, 3, 5]
Optimal Memory State: [0, 1, 3, 6]
Optimal Memory State: [0, 1, 3, 6]
Optimal Memory State: [0, 1, 3, 4]
Optimal Memory State: [0, 1, 2, 3]
Optimal Memory State: [0, 1, 2, 3]
Optimal Memory State: [0, 1, 2, 3]
Optimal Memory State: [0, 1, 2, 3]
Optimal Page Faults: 7
Optimal Page Hits: 5
Optimal Page Hit Ratio: 0.42
Optimal Page Fault Ratio: 0.58
```