Page Replacement – OPTIMAL-13

```
import java.util.*;


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


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


        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();


        int[] frame = new int[frames];
        Arrays.fill(frame, -1);


        int pageFaults = 0;


        for (int i = 0; i < n; i++) {
            int page = pages[i];
            boolean hit = false;


            // Check if page already exists
            for (int j = 0; j < frames; j++) {
                if (frame[j] == page) {
                    hit = true;
                    break;
    }
}

// If page fault occurs
if (!hit) {
    int replaceIndex = -1, farthest = -1;

    // Find a free frame first
    for (int j = 0; j < frames; j++) {
        if (frame[j] == -1) {
            replaceIndex = j;
            break;
        }
    }

    // If no free frame, apply optimal strategy
    if (replaceIndex == -1) {
        for (int j = 0; j < frames; j++) {
            int nextUse = Integer.MAX_VALUE;
            for (int k = i + 1; k < n; k++) {
                if (frame[j] == pages[k]) {
                    nextUse = k;
                    break;
                }
            }
            if (nextUse > farthest) {
                farthest = nextUse;
                replaceIndex = j;
            }
        }
    }
```

```
                frame[replaceIndex] = page;
                pageFaults++;
            }

            // Display frame status
            System.out.print("Frames: ");
            for (int f : frame) System.out.print((f == -1 ? "-" : f) + " ");
            System.out.println();
        }

        System.out.println("\nTotal Page Faults = " + pageFaults);
    }
}
```