Ass4.java :-

```
import java.util.*;

public class Ass4 {

    static Scanner sc = new Scanner(System.in);

    // Print frame contents
    static void printFrames(List<Integer> frames) {
        for (int f : frames) System.out.print(f + " ");
        System.out.println();
    }

    // FIFO Page Replacement
    static void fifo(int[] pages, int n, int capacity) {
        Queue<Integer> frames = new LinkedList<>();
        int pageFaults = 0;
        System.out.println("\n--- FIFO Page Replacement ---");

        for (int i = 0; i < n; i++) {
            int page = pages[i];
            if (!frames.contains(page)) {
                if (frames.size() == capacity) {
                    frames.poll(); // remove oldest
                }
                frames.add(page);
                pageFaults++;
                System.out.print("Page " + page + " caused FAULT -> Frames: ");
            } else {
                System.out.print("Page " + page + " -> Frames: ");
            }
            printFrames(new ArrayList<>(frames));
        }

        System.out.println("Total Page Faults = " + pageFaults);
        System.out.printf("Page Fault Rate = %.2f%%\n", (pageFaults * 100.0 /
n));
    }

    // LRU Page Replacement
    static void lru(int[] pages, int n, int capacity) {
        Set<Integer> frames = new HashSet<>();
        Map<Integer, Integer> indexes = new HashMap<>();
        int pageFaults = 0;
        System.out.println("\n--- LRU Page Replacement ---");

        for (int i = 0; i < n; i++) {
            int page = pages[i];
            if (!frames.contains(page)) {
                if (frames.size() < capacity) {
                    frames.add(page);
                } else {
                    int lru = Integer.MAX_VALUE, val = -1;
                    for (int f : frames) {
                        if (indexes.get(f) < lru) {
                            lru = indexes.get(f);
                            val = f;
                        }
                    }
                    frames.remove(val);
                    frames.add(page);
                }
                pageFaults++;
```

```
                System.out.print("Page " + page + " caused FAULT -> Frames: ");
            } else {
                System.out.print("Page " + page + " -> Frames: ");
            }
            indexes.put(page, i);
            printFrames(new ArrayList<>(frames));
        }

        System.out.println("Total Page Faults = " + pageFaults);
        System.out.printf("Page Fault Rate = %.2f%%\n", (pageFaults * 100.0 /
n));
    }

    // Optimal Page Replacement
    static void optimal(int[] pages, int n, int capacity) {
        List<Integer> frames = new ArrayList<>();
        int pageFaults = 0;
        System.out.println("\n--- Optimal Page Replacement ---");

        for (int i = 0; i < n; i++) {
            int page = pages[i];
            if (!frames.contains(page)) {
                if (frames.size() < capacity) {
                    frames.add(page);
                } else {
                    int farthest = i + 1, idx = -1;
                    for (int j = 0; j < frames.size(); j++) {
                        int nextUse = Integer.MAX_VALUE;
                        for (int k = i + 1; k < n; k++) {
                            if (frames.get(j) == pages[k]) {
                                nextUse = k;
                                break;
                            }
                        }
                        if (nextUse > farthest) {
                            farthest = nextUse;
                            idx = j;
                        }
                    }
                    if (idx == -1) idx = 0;
                    frames.set(idx, page);
                }
                pageFaults++;
                System.out.print("Page " + page + " caused FAULT -> Frames: ");
            } else {
                System.out.print("Page " + page + " -> Frames: ");
            }
            printFrames(frames);
        }

        System.out.println("Total Page Faults = " + pageFaults);
        System.out.printf("Page Fault Rate = %.2f%%\n", (pageFaults * 100.0 /
n));
    }

    // Main Menu
    public static void main(String[] args) {
        System.out.print("Enter number of pages in reference string: ");
        int n = sc.nextInt();
        int[] pages = new int[n];
        System.out.println("Enter reference string:");
        for (int i = 0; i < n; i++) pages[i] = sc.nextInt();

        System.out.print("Enter number of frames: ");
```

```
        int capacity = sc.nextInt();

        while (true) {
            System.out.println("\nChoose Page Replacement Algorithm:");
            System.out.println("1. FIFO");
            System.out.println("2. LRU");
            System.out.println("3. Optimal");
            System.out.println("4. Run All");
            System.out.println("5. Exit");
            int choice = sc.nextInt();
            switch (choice) {
                case 1: fifo(pages, n, capacity); break;
                case 2: lru(pages, n, capacity); break;
                case 3: optimal(pages, n, capacity); break;
                case 4:
                    fifo(pages, n, capacity);
                    lru(pages, n, capacity);
                    optimal(pages, n, capacity);
                    break;
                case 5: System.exit(0);
                default: System.out.println("Invalid choice!");
            }
        }
    }
}
```

OUTPUT:-

```
swaraj@swaraj-VirtualBox:~/LP-1$ javac Ass4.java
swaraj@swaraj-VirtualBox:~/LP-1$ java Ass4
Enter number of pages in reference string: 12
Enter reference string:
4 2 5 2 1 6 4 7 2 5 6 3
Enter number of frames: 3

Choose Page Replacement Algorithm:
1. FIFO
2. LRU
3. Optimal
4. Run All
5. Exit
1

--- FIFO Page Replacement ---
Page 4 caused FAULT -> Frames: 4
Page 2 caused FAULT -> Frames: 4 2
Page 5 caused FAULT -> Frames: 4 2 5
Page 2 -> Frames: 4 2 5
Page 1 caused FAULT -> Frames: 2 5 1
Page 6 caused FAULT -> Frames: 5 1 6
Page 4 caused FAULT -> Frames: 1 6 4
Page 7 caused FAULT -> Frames: 6 4 7
Page 2 caused FAULT -> Frames: 4 7 2
Page 5 caused FAULT -> Frames: 7 2 5
Page 6 caused FAULT -> Frames: 2 5 6
Page 3 caused FAULT -> Frames: 5 6 3
Total Page Faults = 11
Page Fault Rate = 91.67%

Choose Page Replacement Algorithm:
1. FIFO
2. LRU
3. Optimal
```

```
4. Run All
5. Exit
2

--- LRU Page Replacement ---
Page 4 caused FAULT -> Frames: 4
Page 2 caused FAULT -> Frames: 2 4
Page 5 caused FAULT -> Frames: 2 4 5
Page 2 -> Frames: 2 4 5
Page 1 caused FAULT -> Frames: 1 2 5
Page 6 caused FAULT -> Frames: 1 2 6
Page 4 caused FAULT -> Frames: 1 4 6
Page 7 caused FAULT -> Frames: 4 6 7
Page 2 caused FAULT -> Frames: 2 4 7
Page 5 caused FAULT -> Frames: 2 5 7
Page 6 caused FAULT -> Frames: 2 5 6
Page 3 caused FAULT -> Frames: 3 5 6
Total Page Faults = 11
Page Fault Rate = 91.67%

Choose Page Replacement Algorithm:
1. FIFO
2. LRU
3. Optimal
4. Run All
5. Exit
3

--- Optimal Page Replacement ---
Page 4 caused FAULT -> Frames: 4
Page 2 caused FAULT -> Frames: 4 2
Page 5 caused FAULT -> Frames: 4 2 5
Page 2 -> Frames: 4 2 5
Page 1 caused FAULT -> Frames: 4 2 1
Page 6 caused FAULT -> Frames: 4 2 6
Page 4 -> Frames: 4 2 6
Page 7 caused FAULT -> Frames: 7 2 6
Page 2 -> Frames: 7 2 6
Page 5 caused FAULT -> Frames: 5 2 6
Page 6 -> Frames: 5 2 6
Page 3 caused FAULT -> Frames: 3 2 6
Total Page Faults = 8
Page Fault Rate = 66.67%

Choose Page Replacement Algorithm:
1. FIFO
2. LRU
3. Optimal
4. Run All
5. Exit
4

--- FIFO Page Replacement ---
Page 4 caused FAULT -> Frames: 4
Page 2 caused FAULT -> Frames: 4 2
Page 5 caused FAULT -> Frames: 4 2 5
Page 2 -> Frames: 4 2 5
Page 1 caused FAULT -> Frames: 2 5 1
Page 6 caused FAULT -> Frames: 5 1 6
Page 4 caused FAULT -> Frames: 1 6 4
Page 7 caused FAULT -> Frames: 6 4 7
Page 2 caused FAULT -> Frames: 4 7 2
Page 5 caused FAULT -> Frames: 7 2 5
Page 6 caused FAULT -> Frames: 2 5 6
```

```
Page 3 caused FAULT -> Frames: 5 6 3
Total Page Faults = 11
Page Fault Rate = 91.67%

--- LRU Page Replacement ---
Page 4 caused FAULT -> Frames: 4
Page 2 caused FAULT -> Frames: 2 4
Page 5 caused FAULT -> Frames: 2 4 5
Page 2 -> Frames: 2 4 5
Page 1 caused FAULT -> Frames: 1 2 5
Page 6 caused FAULT -> Frames: 1 2 6
Page 4 caused FAULT -> Frames: 1 4 6
Page 7 caused FAULT -> Frames: 4 6 7
Page 2 caused FAULT -> Frames: 2 4 7
Page 5 caused FAULT -> Frames: 2 5 7
Page 6 caused FAULT -> Frames: 2 5 6
Page 3 caused FAULT -> Frames: 3 5 6
Total Page Faults = 11
Page Fault Rate = 91.67%

--- Optimal Page Replacement ---
Page 4 caused FAULT -> Frames: 4
Page 2 caused FAULT -> Frames: 4 2
Page 5 caused FAULT -> Frames: 4 2 5
Page 2 -> Frames: 4 2 5
Page 1 caused FAULT -> Frames: 4 2 1
Page 6 caused FAULT -> Frames: 4 2 6
Page 4 -> Frames: 4 2 6
Page 7 caused FAULT -> Frames: 7 2 6
Page 2 -> Frames: 7 2 6
Page 5 caused FAULT -> Frames: 5 2 6
Page 6 -> Frames: 5 2 6
Page 3 caused FAULT -> Frames: 3 2 6
Total Page Faults = 8
Page Fault Rate = 66.67%

Choose Page Replacement Algorithm:
1. FIFO
2. LRU
3. Optimal
4. Run All
5. Exit
5
```