

Program Code –

```
import java.util.concurrent.Semaphore;

class SharedData {
    int data = 0; // Shared resource
    int readerCount = 0; // Number of active readers

    // Semaphores
    Semaphore mutex = new Semaphore(1); //
    Protect readerCount
    Semaphore wrt = new Semaphore(1); //
    Protect shared data (writers)
}

// Reader class
class Reader extends Thread {
    SharedData shared;
    int readerId;

    Reader(SharedData shared, int id) {
        this.shared = shared;
        this.readerId = id;
    }

    public void run() {
        try {
            while (true) {
                // Entry section
                shared.mutex.acquire();
                shared.readerCount++;
                if (shared.readerCount == 1)
                    shared.wrt.acquire(); // First reader
locks writer
                shared.mutex.release();
```

```
                // Critical section
                System.out.println("Reader " + readerId
+ " is reading: " + shared.data);
                Thread.sleep(500); // Simulate reading
time
                // Exit section
                shared.mutex.acquire();
                shared.readerCount--;
                if (shared.readerCount == 0)
                    shared.wrt.release(); // Last reader
releases writer
                shared.mutex.release();

                Thread.sleep(500); // Simulate other
work
            }
        } catch (InterruptedException e) {
            System.out.println("Reader interrupted");
        }
    }
}

// Writer class
class Writer extends Thread {
    SharedData shared;
    int writerId;

    Writer(SharedData shared, int id) {
        this.shared = shared;
        this.writerId = id;
    }

    public void run() {
```

```

try {
    while (true) {
        shared.wrt.acquire(); // Only writer
access
        shared.data += 1;    // Modify shared
data
        System.out.println("Writer " + writerId
+ " is writing: " + shared.data);
        Thread.sleep(1000); // Simulate writing
time
        shared.wrt.release();

        Thread.sleep(500); // Simulate other
work
    }
} catch (InterruptedException e) {
    System.out.println("Writer interrupted");
}
}
}

```

```

// Main class
public class ReaderWriterProblem {

    public static void main(String[] args) {

        SharedData shared = new SharedData();

        // Create readers and writers
        Reader r1 = new Reader(shared, 1);
        Reader r2 = new Reader(shared, 2);
        Writer w1 = new Writer(shared, 1);
        Writer w2 = new Writer(shared, 2);

        // Start threads
        r1.start();
        r2.start();
        w1.start();
        w2.start();
    }
}

```

Output –

```

sanket-kotkar@sanket-kotkar-VirtualBox: ~/Documents/Practical 4
sanket-kotkar@sanket-kotkar-VirtualBox:~/Documents/Practical 4$ javac ReaderWriterProblem.java
sanket-kotkar@sanket-kotkar-VirtualBox:~/Documents/Practical 4$ java ReaderWriterProblem
Reader 1 is reading: 0
Reader 2 is reading: 0
Writer 1 is writing: 1
Writer 2 is writing: 2
Reader 1 is reading: 2
Reader 2 is reading: 2
Writer 1 is writing: 3
Writer 2 is writing: 4
Reader 1 is reading: 4
Reader 2 is reading: 4
Writer 2 is writing: 5
Writer 1 is writing: 6
Reader 1 is reading: 6
Reader 2 is reading: 6
Writer 2 is writing: 7
Writer 1 is writing: 8
Reader 2 is reading: 8
Reader 1 is reading: 8
Writer 2 is writing: 9
Writer 1 is writing: 10
Reader 2 is reading: 10
Reader 1 is reading: 10
Writer 2 is writing: 11
Writer 1 is writing: 12
Reader 2 is reading: 12
Reader 1 is reading: 12
Writer 2 is writing: 13
Writer 1 is writing: 14
Reader 2 is reading: 14
Reader 1 is reading: 14
Writer 2 is writing: 15

```

1. FIFO Page Replacement -

Program Code –

```
import java.util.*;
```

```
public class FIFO {  
    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();  
        }  
  
        Queue<Integer> queue = new  
LinkedList<>();  
        Set<Integer> set = new HashSet<>();  
        int faults = 0;
```

```
        for (int p : pages) {  
            if (!set.contains(p)) {  
                if (set.size() < frames) {  
                    set.add(p);  
                    queue.add(p);  
                } else {  
                    int removed = queue.poll();  
                    set.remove(removed);  
                    set.add(p);  
                    queue.add(p);  
                }  
                faults++;  
            }  
            System.out.println("Page: " + p + " ->  
Frames: " + set);  
        }  
  
        System.out.println("Total Page Faults  
(FIFO): " + faults);  
        sc.close();  
    }  
}
```

Output –

```
Oct 2 09:25
sanket-kotkar@sanket-kotkar-VirtualBox: ~/Documents/Practical 7
sanket-kotkar@sanket-kotkar-VirtualBox:~/Documents/Practical 7$ javac FIFO.java
sanket-kotkar@sanket-kotkar-VirtualBox:~/Documents/Practical 7$ java FIFO
Enter number of frames: 3
Enter number of pages: 12
Enter page reference string:
7 0 1 2 0 3 0 4 2 3 0 3
Page: 7 -> Frames: [7]
Page: 0 -> Frames: [0, 7]
Page: 1 -> Frames: [0, 1, 7]
Page: 2 -> Frames: [0, 1, 2]
Page: 0 -> Frames: [0, 1, 2]
Page: 3 -> Frames: [1, 2, 3]
Page: 0 -> Frames: [0, 2, 3]
Page: 4 -> Frames: [0, 3, 4]
Page: 2 -> Frames: [0, 2, 4]
Page: 3 -> Frames: [2, 3, 4]
Page: 0 -> Frames: [0, 2, 3]
Page: 3 -> Frames: [0, 2, 3]
Total Page Faults (FIFO): 10
sanket-kotkar@sanket-kotkar-VirtualBox:~/Documents/Practical 7$
```

2. LRU Page Replacement –

Program Code –

```
import java.util.*;

public class LRU {

    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();
        }

        List<Integer> list = new ArrayList<>();
        Set<Integer> set = new HashSet<>();
        int faults = 0;

        for (int p : pages) {
            if (!set.contains(p)) {
                if (set.size() < frames) {
                    set.add(p);
                    list.add(p);
                } else {
                    int lru = list.remove(0);
                    set.remove(lru);
                    set.add(p);
                }
            }
            faults++;
        }

        System.out.println("Total Page Faults (LRU): " + faults);
    }
}
```

```

        list.add(p);
    }

    faults++;
    System.out.println("Total Page Faults (LRU): " + faults);
} else {
    sc.close();
    list.remove((Integer) p);
    list.add(p);
}

System.out.println("Page: " + p + " -> Frames: " + list);

```

Output –

```

sanket-kotkar@sanket-kotkar-VirtualBox: ~/Documents/Practical 7
sanket-kotkar@sanket-kotkar-VirtualBox:~/Documents/Practical 7$ javac LRU.java
sanket-kotkar@sanket-kotkar-VirtualBox:~/Documents/Practical 7$ java LRU
Enter number of frames: 3
Enter number of pages: 12
Enter page reference string:
7 0 1 2 0 3 0 4 2 3 0 3
Page: 7 -> Frames: [7]
Page: 0 -> Frames: [7, 0]
Page: 1 -> Frames: [7, 0, 1]
Page: 2 -> Frames: [0, 1, 2]
Page: 0 -> Frames: [1, 2, 0]
Page: 3 -> Frames: [2, 0, 3]
Page: 0 -> Frames: [2, 3, 0]
Page: 4 -> Frames: [3, 0, 4]
Page: 2 -> Frames: [0, 4, 2]
Page: 3 -> Frames: [4, 2, 3]
Page: 0 -> Frames: [2, 3, 0]
Page: 3 -> Frames: [2, 0, 3]
Total Page Faults (LRU): 9
sanket-kotkar@sanket-kotkar-VirtualBox:~/Documents/Practical 7$

```

3. Optimal Page Replacement –

Program Code –

```

import java.util.*;

public class Optimal {
    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();
}

List<Integer> memory = new ArrayList<>();
int faults = 0;

for (int i = 0; i < pages.length; i++) {
    int p = pages[i];

    if (!memory.contains(p)) {
        if (memory.size() < frames) {
            memory.add(p);
        } else {
            int farthest = -1, indexToReplace = -
1;
            for (int j = 0; j < memory.size(); j++)
            {
                int page = memory.get(j);
                int nextUse =
Integer.MAX_VALUE;

                for (int k = i + 1; k < pages.length;
k++) {
                    if (pages[k] == page) {
                        nextUse = k;
                        break;
                    }
                }
                if (nextUse > farthest) {
                    farthest = nextUse;
                    indexToReplace = j;
                }
            }
            memory.set(indexToReplace, p);
            faults++;
        }
        System.out.println("Page: " + p + " ->
Frames: " + memory);
    }

    System.out.println("Total Page Faults
(Optional): " + faults);
    sc.close();
}

```

Output -

```
sanket-kotkar@sanket-kotkar-VirtualBox:~/Documents/Practical7$ javac Optimal.java
sanket-kotkar@sanket-kotkar-VirtualBox:~/Documents/Practical7$ java Optimal
Enter number of frames: 3
Enter number of pages: 12
Enter page reference string:
7 0 1 2 0 3 0 4 2 3 0 3
Page: 7 -> Frames: [7]
Page: 0 -> Frames: [7, 0]
Page: 1 -> Frames: [7, 0, 1]
Page: 2 -> Frames: [2, 0, 1]
Page: 0 -> Frames: [2, 0, 1]
Page: 3 -> Frames: [2, 0, 3]
Page: 0 -> Frames: [2, 0, 3]
Page: 4 -> Frames: [2, 4, 3]
Page: 2 -> Frames: [2, 4, 3]
Page: 3 -> Frames: [2, 4, 3]
Page: 0 -> Frames: [0, 4, 3]
Page: 3 -> Frames: [0, 4, 3]
Total Page Faults (Optimal): 7
sanket-kotkar@sanket-kotkar-VirtualBox:~/Documents/Practical7$
```