D**
**

EXPERIMENT NO : 7

TITLE : Write a java program to implement Page Replacement Algorithm FIFO, LRU and OPT.

NAME : DESALE DARSHAN RAMDAS

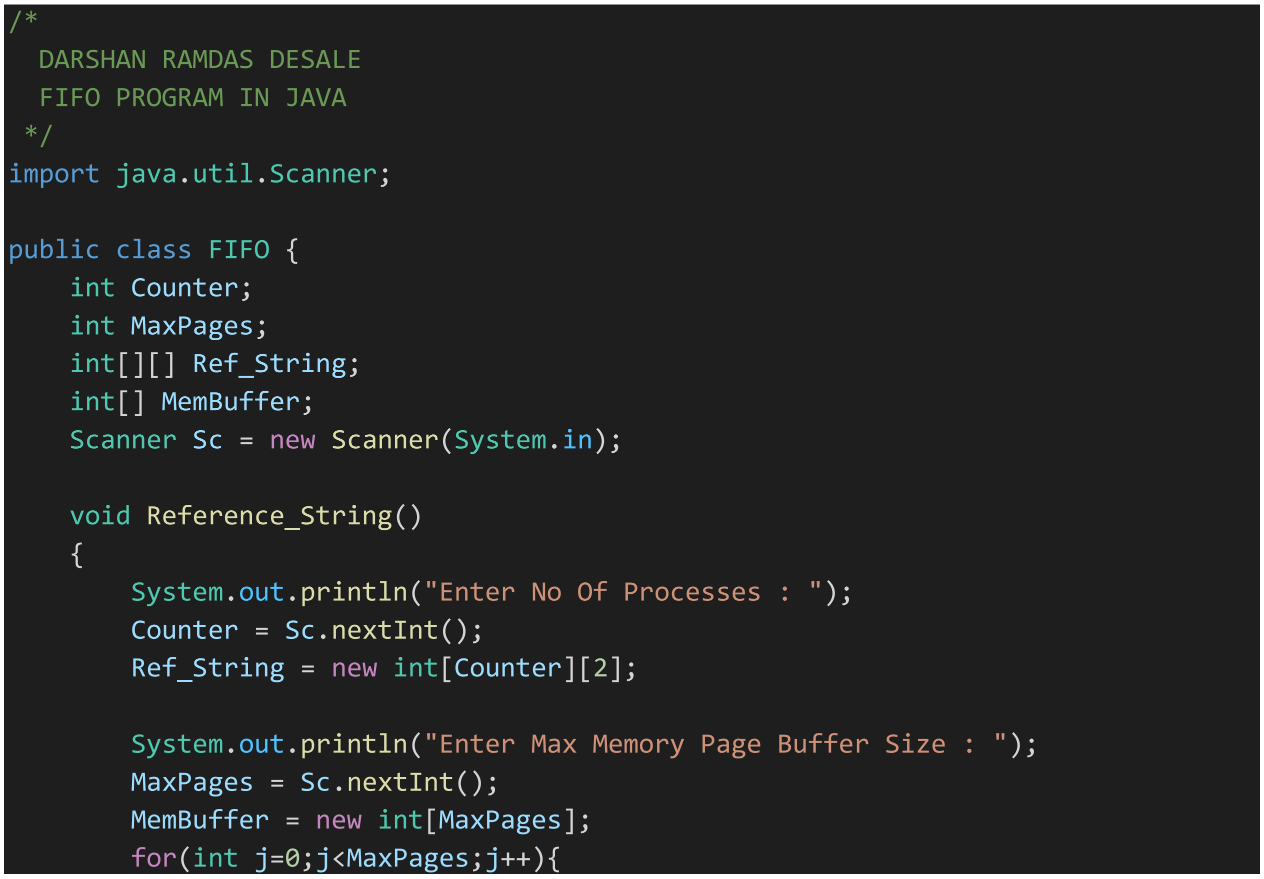
CLASS : TE

ROLL NO : 23 A

DATE : 24-09-2022 BATCH : A

****

PAGE REPLACEMENT PROGRAMS : FIFO, LRU, OPT



```java
/*
  DARSHAN RAMDAS DESALE
  FIFO PROGRAM IN JAVA
 */
import java.util.Scanner;

public class FIFO {
    int Counter;
    int MaxPages;
    int[][] Ref_String;
    int[] MemBuffer;
    Scanner Sc = new Scanner(System.in);

    void Reference_String()
    {
        System.out.println("Enter No Of Processes : ");
        Counter = Sc.nextInt();
        Ref_String = new int[Counter][2];

        System.out.println("Enter Max Memory Page Buffer Size : ");
        MaxPages = Sc.nextInt();
        MemBuffer = new int[MaxPages];
        for(int j=0;j<MaxPages;j++){
            MemBuffer[j] = -1;
        }

        System.out.println("Enter Processes in Reference String : ");
        for(int i = 0 ; i < Counter ; i++){
            Ref_String[i][0] = Sc.nextInt();
            Ref_String[i][1] = -1;
        }
    }

    void rfifo(){
        for(int i = 0; i < Counter; i++){
            if(! isExists(Ref_String[i][0])){
                getMemLoc(Ref_String[i][0]);
                Ref_String[i][1] = 1;
            }else{
                Ref_String[i][1] = 0;
            }

            System.out.println(" Ref_String : "+Ref_String[i][0]);
            for(int j=0;j<MaxPages;j++){
                System.out.println(" M["+j+"]:"+MemBuffer[j]);
            }
        }

    }

    void displayHitFault(){

        System.out.println("Hit Counter : "+getHitCounter());
        float HitRatio = Float.parseFloat(Integer.toString(getHitCounter())) /
Float.parseFloat(Integer.toString(Counter));
        System.out.println("Hit Ratio : "+HitRatio);

        System.out.println("Fault Counter : "+getFaultCounter());
        float FaultRatio = Float.parseFloat(Integer.toString(getFaultCounter()))
/ Float.parseFloat(Integer.toString(Counter));
        System.out.println("Fault Ratio : "+FaultRatio);
    }

    boolean isExists(int num){
        for(int j=0;j<MaxPages;j++){
            if(MemBuffer[j] == num )
                    return true;
        }
        return false;
    }

    void getMemLoc(int num){
        int j=0;
        for(;j<MaxPages-1;j++){
            MemBuffer[j] = MemBuffer[j+1];
        }
        MemBuffer[j] = num;
    }

    int getHitCounter(){
        int hitCounter=0;
        for(int i = 0; i < Counter; i++){
            if(Ref_String[i][1] == 0){
                hitCounter++;
            }
        }
        return hitCounter;
    }

    int getFaultCounter(){
        int faultCounter=0;
        for(int i = 0; i < Counter; i++){
            if(Ref_String[i][1] == 1){
                faultCounter++;
            }
        }
        return faultCounter;
    }

    public static void main(String[] args) {

        System.out.println("FIFO Page Replacement Algorithm:");
        FIFO fifo = new FIFO();
        fifo.Reference_String();
        fifo.rfifo();
        fifo.displayHitFault();
    }

}
```

OUTPUT :

FIFO :

```
Enter No Of Processes :
14
Enter Max Memory Page Buffer Size :
3
Enter Processes in Reference String :
7 0 1 2 0 3 0 4 2 3 0 3 2 1
 Ref_String : 7
 M[0]:-1
 M[1]:-1
 M[2]:7
 Ref_String : 0
 M[0]:-1
 M[1]:7
 M[2]:0
 Ref_String : 1
 M[0]:7
 M[1]:0
 M[2]:1
 Ref_String : 2
 M[0]:0
 M[1]:1
 M[2]:2
 Ref_String : 0
 M[0]:0
 M[1]:1
 M[2]:2
 Ref_String : 3
 M[0]:1
 M[1]:2
 M[2]:3
 Ref_String : 0
 M[0]:2
 M[1]:3
 M[2]:0
 Ref_String : 4
 M[0]:3
 M[1]:0
 M[2]:4
 Ref_String : 2
 M[0]:0
 M[1]:4
 M[2]:2
 Ref_String : 3
 M[0]:4
 M[1]:2
 M[2]:3
 Ref_String : 0
 M[0]:2
 M[1]:3
 M[2]:0
 Ref_String : 3
 M[0]:2
 M[1]:3
 M[2]:0
 Ref_String : 2
 M[0]:2
 M[1]:3
 M[2]:0
 Ref_String : 1
 M[0]:3
 M[1]:0
 M[2]:1
Hit Counter : 3
Hit Ratio : 0.21428572
Fault Counter : 11
Fault Ratio : 0.78571427
```

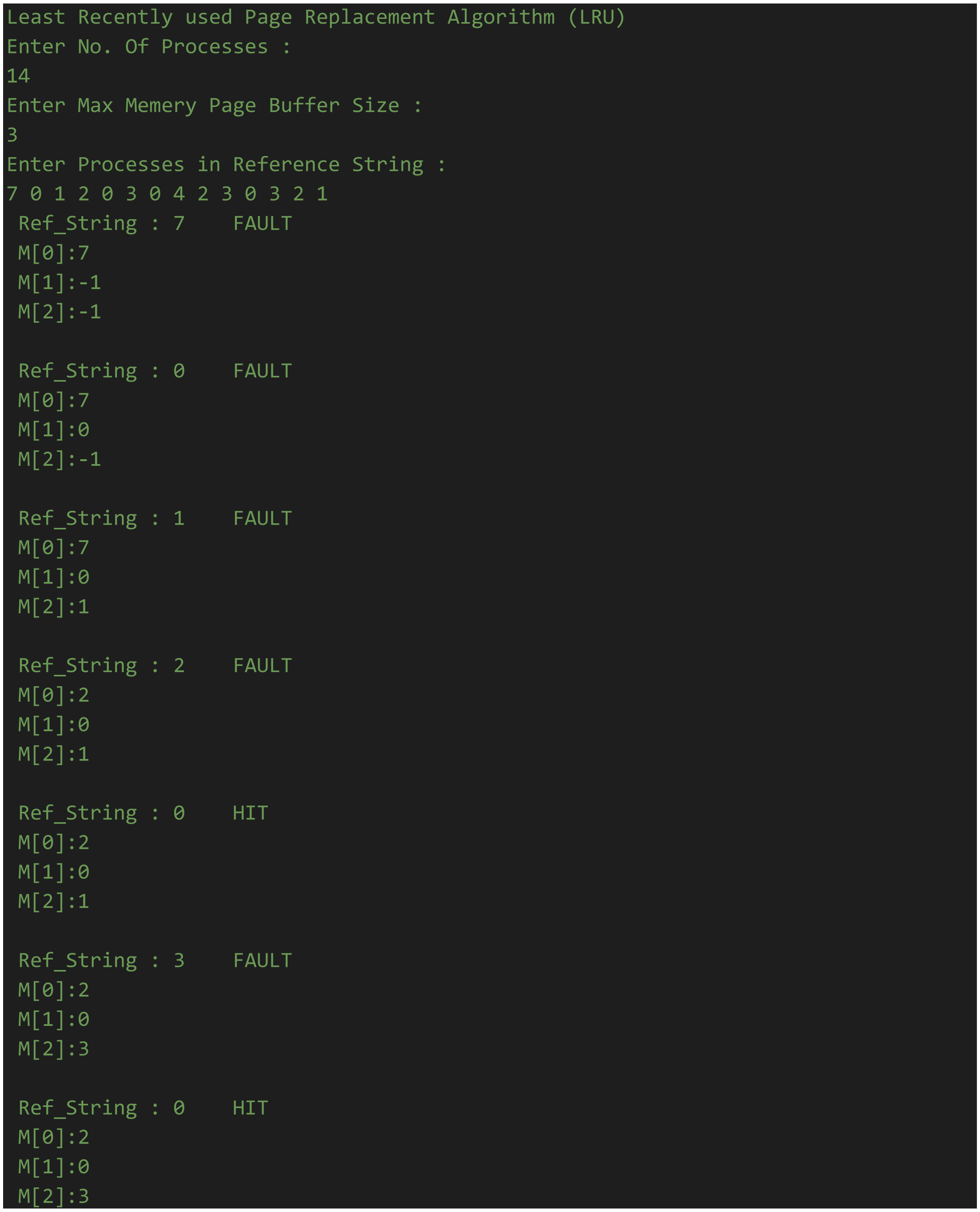
PAGE REPLACEMENT PROGRM :LRU IN JAVA :

```java
import java.util.*;
public class LeastRecentlyUsed {

    int Counter;
    int MaxPages;
    int[][] Ref_String;
    int[][] MemBuffer;
    Scanner Sc = new Scanner(System.in);

    void Reference_String(){
        System.out.println("Enter No. Of Processes : ");
        Counter = Sc.nextInt();
        Ref_String = new int[Counter][2];

        System.out.println("Enter Max Memery Page Buffer Size : ");
        MaxPages = Sc.nextInt();
        MemBuffer = new int[MaxPages][2];
        for(int i=0;i<MaxPages;i++){
            MemBuffer[i][0] = -1;
            MemBuffer[i][1] = 0;
        }
        System.out.println("Enter Processes in Reference String : ");
        for(int i = 0 ; i < Counter ; i++){
            Ref_String[i][0] = Sc.nextInt();
            Ref_String[i][1] = -1;
        }
    }

    void rLRU(){
        for(int i = 0; i < Counter; i++){
            if(isEmptyLocInMemBuffer()){
                Ref_String[i][1] = insertInitially(Ref_String[i][0]);
            }else{
                if(isExists(Ref_String[i][0])){
                    Ref_String[i][1] = 0; //It i s Hit
                }else{
                    replacePage(Ref_String[i][0],i);
                    Ref_String[i][1] = 1;   // It is a Fault
                }
            }
            System.out.print(" Ref_String : "+Ref_String[i][0]+"    " );
            if(Ref_String[i][1] == 0)               System.out.println("HIT");
            else                System.out.println("FAULT");
            for(int j=0;j<MaxPages;j++){
                System.out.println(" M["+j+"]:"+MemBuffer[j][0]+"  ");
            }
            System.out.println(" ");
        }
    }

    void replacePage(int PageNo,int PageLocInRef_String){
        int loc = getReplaceLoc(PageLocInRef_String);
        MemBuffer[loc][0] = PageNo;
    }

    int getReplaceLoc(int PageLocInRef_String){
        int memLoc=0;
        int Counter = 0;
        for(int j=PageLocInRef_String-1;j >= 0;j--){
            for(int i=0;i < MaxPages;i++){
                if(MemBuffer[i][0] == Ref_String[j][0] && MemBuffer[i][1] != 1){
                    MemBuffer[i][1] = 1;
                    Counter++;
                    break;
                }
                if(Counter == MaxPages-1)
                    break;
            }
        }
        for(int i=0;i < MaxPages;i++){
            if(MemBuffer[i][1] != 1){
                memLoc = i;
                break;
            }else{
                MemBuffer[i][1]=0;
            }
        }

        return memLoc;
    }
    boolean isExists(int num){
        for(int j=0;j<MaxPages;j++){
            if(MemBuffer[j][0] == num )
                    return true;
        }
        return false;
    }

    boolean isEmptyLocInMemBuffer(){
        for(int j=0;j<MaxPages;j++){
            if(MemBuffer[j][0] == -1 )
                    return true;
        }
        return false;
    }

    int insertInitially(int num){
        for(int j=0;j<MaxPages;j++){
            if(MemBuffer[j][0] == -1){
                if(isExists(num)){
                    return  0;  // It is a HIT
                }else{
                    MemBuffer[j][0] = num;
                    return  1;  // It is a Fault
                }
            }
        }
        return 0;
```

```java
    }

    void displayHitFault(){

        System.out.println("Hit Counter : "+getHitCounter());
        float HitRatio = Float.parseFloat(Integer.toString(getHitCounter())) /
Float.parseFloat(Integer.toString(Counter));
        System.out.println("Hit Ratio : "+HitRatio);

        System.out.println("Fault Counter : "+getFaultCounter());
        float FaultRatio = Float.parseFloat(Integer.toString(getFaultCounter()))
/ Float.parseFloat(Integer.toString(Counter));
        System.out.println("Fault Ratio : "+FaultRatio);
    }

    int getHitCounter(){
        int hitCounter=0;
        for(int i = 0; i < Counter; i++){
            if(Ref_String[i][1] == 0){
                hitCounter++;
            }
        }
        return hitCounter;
    }

    int getFaultCounter(){
        int faultCounter=0;
        for(int i = 0; i < Counter; i++){
            if(Ref_String[i][1] == 1){
                faultCounter++;
            }
        }
        return faultCounter;
    }

    public static void main(String[] args) {
        System.out.println("Least Recently used Page Replacement Algorithm
(LRU)");
        LeastRecentlyUsed lru = new LeastRecentlyUsed();
        lru.Reference_String();
        lru.rLRU();
        lru.displayHitFault();
    }
}
```

OUTPUT :

LRU:



```
Least Recently used Page Replacement Algorithm (LRU)
Enter No. Of Processes :
14
Enter Max Memery Page Buffer Size :
3
Enter Processes in Reference String :
7 0 1 2 0 3 0 4 2 3 0 3 2 1
 Ref_String : 7    FAULT
 M[0]:7
 M[1]:-1
 M[2]:-1

 Ref_String : 0    FAULT
 M[0]:7
 M[1]:0
 M[2]:-1

 Ref_String : 1    FAULT
 M[0]:7
 M[1]:0
 M[2]:1

 Ref_String : 2    FAULT
 M[0]:2
 M[1]:0
 M[2]:1

 Ref_String : 0    HIT
 M[0]:2
 M[1]:0
 M[2]:1

 Ref_String : 3    FAULT
 M[0]:2
 M[1]:0
 M[2]:3

 Ref_String : 0    HIT
 M[0]:2
 M[1]:0
 M[2]:3
 Ref_String : 4    FAULT
 M[0]:4
 M[1]:0
 M[2]:3

 Ref_String : 2    FAULT
 M[0]:4
 M[1]:0
 M[2]:2

 Ref_String : 3    FAULT
 M[0]:4
 M[1]:3
 M[2]:2

 Ref_String : 0    FAULT
 M[0]:0
 M[1]:3
 M[2]:2

 Ref_String : 3    HIT
 M[0]:0
 M[1]:3
 M[2]:2

 Ref_String : 2    HIT
 M[0]:0
 M[1]:3
 M[2]:2

 Ref_String : 1    FAULT
 M[0]:1
 M[1]:3
 M[2]:2

Hit Counter : 4
Hit Ratio : 0.2857143
Fault Counter : 10
Fault Ratio : 0.71428573
```

PAGE REPLACEMENT PROGRAM OF OPTIMAL IN JAVA :

```
import java.util.Scanner;

public class Optimal {
```

```
int Counter;
int Maxpages;
int[][] Ref_String;
int[][] MemBuffer;
Scanner scan = new Scanner(System.in);

void Reference_String(){
    System.out.println("Enter No. Of Processes : ");
    Counter = scan.nextInt();
    Ref_String = new int[Counter][2];

    System.out.println("Enter Max Memery Page Buffer Size : ");
    Maxpages = scan.nextInt();
    MemBuffer = new int[Maxpages][2];
    for(int i=0;i<Maxpages;i++){
        MemBuffer[i][0] = -1;
        MemBuffer[i][1] = 0;
    }
    System.out.println("Enter Processes in Reference String : ");
    for(int i = 0 ; i < Counter ; i++){
        Ref_String[i][0] = scan.nextInt();
        Ref_String[i][1] = -1;
    }
}

void rOPT(){
    for(int i = 0; i < Counter; i++){
        if(isEmptyLocInMemBuffer()){
            Ref_String[i][1] = insertInitially(Ref_String[i][0]);
        }else{
            if(isExists(Ref_String[i][0])){
                Ref_String[i][1] = 0; //It i s Hit
            }else{
                replacePage(Ref_String[i][0],i);
                Ref_String[i][1] = 1;   // It is a Fault
            }
        }
        System.out.print(" Ref_String : "+Ref_String[i][0]+"    " );
        if(Ref_String[i][1] == 0)               System.out.println("HIT");
        else                System.out.println("FAULT");
        for(int j=0;j<Maxpages;j++){
            System.out.println(" M["+j+"]:"+MemBuffer[j][0]+"  ");
        }
        System.out.println(" ");
    }
}

void replacePage(int PageNo,int PageLocInRef_String){
    int loc = getReplaceLoc(PageLocInRef_String);
    MemBuffer[loc][0] = PageNo;
}

int getReplaceLoc(int PageLocInRef_String){
    int memLoc=0;
    int Counter = 0;
    for(int j=PageLocInRef_String+1;j<Counter ;j++){//ONLY Change
        for(int i=0;i < Maxpages;i++){
            if(MemBuffer[i][0] == Ref_String[j][0] && MemBuffer[i][1] != 1){
                MemBuffer[i][1] = 1;
                Counter++;
                break;
            }
            if(Counter == Maxpages-1)
                break;
        }
    }
    for(int i=0;i < Maxpages;i++){
        if(MemBuffer[i][1] != 1){
            memLoc = i;
            break;
        }else{
            MemBuffer[i][1]=0;
        }
    }

    return memLoc;
}

boolean isExists(int num){
    for(int j=0;j<Maxpages;j++){
        if(MemBuffer[j][0] == num )
                return true;
    }
    return false;
}

boolean isEmptyLocInMemBuffer(){
    for(int j=0;j<Maxpages;j++){
        if(MemBuffer[j][0] == -1 )
```

```
                    return true;
            }
        return false;
    }

    int insertInitially(int num){
        for(int j=0;j<Maxpages;j++){
            if(MemBuffer[j][0] == -1){
                if(isExists(num)){
                    return  0;  // It is a HIT
                }else{
                    MemBuffer[j][0] = num;
                    return  1;  // It is a Fault
                }
            }
        }
        return 0;
    }

    void displayHitFault(){

        System.out.println("Hit Counter : "+getHitCounter());
        float HitRatio = Float.parseFloat(Integer.toString(getHitCounter())) /
Float.parseFloat(Integer.toString(Counter));
        System.out.println("Hit Ratio : "+HitRatio);

        System.out.println("Fault Counter : "+getFaultCounter());
        float FaultRatio = Float.parseFloat(Integer.toString(getFaultCounter()))
/ Float.parseFloat(Integer.toString(Counter));
        System.out.println("Fault Ratio : "+FaultRatio);
    }

    int getHitCounter(){
        int hitCounter=0;
        for(int i = 0; i < Counter; i++){
            if(Ref_String[i][1] == 0){
                hitCounter++;
            }
        }
        return hitCounter;
    }

    int getFaultCounter(){
        int faultCounter=0;
        for(int i = 0; i < Counter; i++){
            if(Ref_String[i][1] == 1){
                faultCounter++;
            }
        }
        return faultCounter;
    }

    public static void main(String[] args) {
        System.out.println("Optimal Page Replacement Algorithm");
        Optimal opt = new Optimal();
        opt.Reference_String();
        opt.rOPT();
        opt.displayHitFault();
    }
}
```

<u>OUTPUT</u> :

OPTIMAL

```
Optimal Page Replacement Algorithm
Enter No. Of Processes :
14
Enter Max Memery Page Buffer Size :
3
Enter Processes in Reference String :
7 0 1 2 0 3 0 4 2 3 0 3 2 1
 Ref_String : 7    FAULT
 M[0]:7
 M[1]:-1
 M[2]:-1

 Ref_String : 0    FAULT
 M[0]:7
 M[1]:0
 M[2]:-1

 Ref_String : 1    FAULT
 M[0]:7
 M[1]:0
 M[2]:1

 Ref_String : 2    FAULT
 M[0]:2
 M[1]:0
```

```
M[2]:1

Ref_String : 0    HIT
M[0]:2
M[1]:0
M[2]:1

Ref_String : 3    FAULT
M[0]:3
M[1]:0
M[2]:1

Ref_String : 0    HIT
M[0]:3
M[1]:0
M[2]:1

Ref_String : 4    FAULT
M[0]:4
M[1]:0
M[2]:1

Ref_String : 2    FAULT
M[0]:2
M[1]:0
M[2]:1

Ref_String : 3    FAULT
M[0]:3
M[1]:0
M[2]:1

Ref_String : 0    HIT
M[0]:3
M[1]:0
M[2]:1

Ref_String : 3    HIT
M[0]:3
M[1]:0
M[2]:1

Ref_String : 2    FAULT
M[0]:2
M[1]:0
```

```
 M[2]:1

 Ref_String : 1    HIT
 M[0]:2
 M[1]:0
 M[2]:1

Hit Counter : 5
Hit Ratio : 0.35714287
Fault Counter : 9
Fault Ratio : 0.64285713
```

THANK YOU !!!