FIFO and LRU

```
//Page replacemnet

import java.util.Scanner;

class FIFO

{
   private int front = -1;
   private int rear = -1;
   private int arr[];

   FIFO(int n)
   {
      arr = new int[n];
      for(int i = 0; i < n; i++)
         arr[i] = -1;
   }

   boolean isEmpty()
   {
      return front == -1;
   }

   boolean isFull()
   {
      return front == rear + 1;
   }

   void enque(int ele)
   {
    if(!isFull())
    {
        if(front == -1)
            front = 0;
        rear = (rear + 1) % arr.length;
        arr[rear] = ele;
    }
}

int deque()
{
    if(!isEmpty())
    {
        int temp = arr[front];
        front = (front + 1) % arr.length;

        if(front == 0 && rear == arr.length - 1 || rear == front - 1)
            front = rear = -1;
        return temp;
    }
    return -1;
}

boolean search(int ele)
{
    for(int i : arr)
        if(i == ele)
            return true;
    return false;
}

void display()
{
        for(int i = 0; i < arr.length; i++)
        {
            System.out.printf("%3d",arr[i]);
        }
        System.out.println();
    }
}

public class PageTrans {
    static void display(int lru[]) {
        for (int i : lru)
            System.out.printf("%3d", i);
        System.out.println();
    }

    static boolean search(int lru[], int e) {
        for (int i : lru)
            if (i == e)
                return true;
        return false;
    }

    static int findLRU(int lru[], int pages[], int ind) {
        int maxd = 0;
        int maxi = 0;
        for (int i = 0; i < lru.length; i++) {
            for (int j = ind - 1; j >= 0; j--) {
                if (lru[i] == pages[j]) {
                    if (maxd < ind - j) {
                        maxd = ind - j;
                        maxi = i;
                    }
                    break;
            }
        }
    }
    return maxi;
}

static boolean forward(int pages[], int ind, int e) {
    for (int i = ind; i < pages.length; i++)
        if (pages[i] == e)
            return true;
    return false;
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter Size : ");
    int size = sc.nextInt();

    System.out.print("Enter Number of pages : ");
    int n = sc.nextInt();

    int pages[] = new int[n];
    System.out.print("Enter " + (n) + " Pages : ");
    for (int i = 0; i < n; i++)
        pages[i] = sc.nextInt();
    int hit = 0;
    int ch;

    do {
        System.out.println("\n--------MENU---------");
        System.out.println("1.FIFO");
        System.out.println("2.LRU");
        System.out.println("3.Exit");
System.out.print("Enter your choice : ");
ch = sc.nextInt();

switch (ch) {
    case 1:
        FIFO que = new FIFO(size);
        System.out.println("FIFO : ");
        for (int i = 0; i < n; i++) {
            if (que.search(-1)) {
                que.enque(pages[i]);
                que.display();
            } else {
                if (!que.search(pages[i])) {
                    que.deque();
                    que.enque(pages[i]);
                    que.display();
                } else {
                    que.display();
                    hit++;
                }
            }

        }
        System.out.println("Total Hits : " + hit);
        System.out.println("Total Faults : " + (n - hit));
        System.out.println();

        break;
    case 2:
        int lru[] = new int[size];
        for (int i = 0; i < size; i++)
            lru[i] = -1;
        int i = 0;
```

```
                hit = 0;
                // For first elements
                System.out.println("\nLRU : ");
                for (int j = 0; j < size; j++) {
                    if (lru[j] == -1) {
                        if (i < n) {
                            lru[j] = pages[i++];
                            display(lru);
                        } else
                            break;
                    }
                }

                // Not for first elements
                for (; i < n; i++) {
                    if (!search(lru, pages[i]))
                        lru[findLRU(lru, pages, i)] = pages[i];
                    else
                        hit++;
                    display(lru);
                }
                System.out.println();
                System.out.println("Total Hits : " + hit);
                System.out.println("Total Faults : " + (n - hit));
                System.out.println();

                break;
            }
        } while (ch != 3);
    }
}
```

Output:

```
Enter Size : 3

Enter Number of pages : 20

Enter 20 Pages : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1


--------MENU---------

1.FIFO

2.LRU

3.Exit

Enter your choice : 1

FIFO :

  7 -1 -1

  7  0 -1

  7  0  1

  2  0  1

  2  0  1

  2  3  1

  2  3  0

  4  3  0

  4  2  0

  4  2  3

  0  2  3

  0  2  3

  0  2  3

  0  1  3
  0  1  2
  0  1  2
  0  1  2
  7  1  2
  7  0  2
  7  0  1
Total Hits : 5
Total Faults : 15

--------MENU---------
1.FIFO
2.LRU
3.Optimal
4.Exit
Enter your choice : 2

LRU :
  7 -1 -1
  7  0 -1
  7  0  1
  2  0  1
  2  0  1
  2  0  3
  2  0  3
```

```
 4  0  3

 4  0  2

 4  3  2

 0  3  2

 0  3  2

 0  3  2

 1  3  2

 1  3  2

 1  0  2

 1  0  2

 1  0  7

 1  0  7

 1  0  7


Total Hits : 8

Total Faults : 12
```

FIFO and Optimal

```
import java.util.Scanner;

class FIFO {
    private int front = -1;
    private int rear = -1;
    private int arr[];

    FIFO(int n) {
        arr = new int[n];
        for (int i = 0; i < n; i++)
            arr[i] = -1;
    }

    boolean isEmpty() {
        return front == -1;
    }

    boolean isFull() {
        return front == rear + 1;
    }

    void enque(int ele) {
        if (!isFull()) {
            if (front == -1)
                front = 0;
            rear = (rear + 1) % arr.length;
            arr[rear] = ele;
        }
    }

    int deque() {
        if (!isEmpty()) {
            int temp = arr[front];
            front = (front + 1) % arr.length;

            if (front == 0 && rear == arr.length - 1 || rear == front - 1)
                front = rear = -1;
            return temp;
        }
        return -1;
    }

    boolean search(int ele) {
        for (int i : arr)
            if (i == ele)
                return true;
        return false;
    }

    void display() {
        for (int i = 0; i < arr.length; i++) {
            System.out.printf("%3d", arr[i]);
        }
        System.out.println();
    }
}

public class PageTrans1 {
    static void display(int opti[]) {
        for (int i : opti)
            System.out.printf("%3d", i);
        System.out.println();
    }
```

```
static boolean search(int opti[], int e) {
    for (int i : opti)
        if (i == e)
            return true;
    return false;
}

static boolean forward(int pages[], int ind, int e) {
    for (int i = ind; i < pages.length; i++)
        if (pages[i] == e)
            return true;
    return false;
}

static int findOP(int opti[], int pages[], int ind) {
    int maxd = -1;
    int maxi = -1;
    for (int i = 0; i < opti.length; i++) {
        if (!forward(pages, ind + 1, opti[i]))
            return i;
        for (int j = ind + 1; j < pages.length; j++) {
            if (opti[i] == pages[j]) {
                if (maxd < j - ind) {
                    maxd = j - ind;
                    maxi = i;
                }
                break;
            }
        }
    }
    return maxi;
}
```

```
public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter Size : ");
    int size = sc.nextInt();

    System.out.print("Enter Number of pages : ");
    int n = sc.nextInt();

    int pages[] = new int[n];
    System.out.print("Enter " + n + " Pages : ");
    for (int i = 0; i < n; i++)
        pages[i] = sc.nextInt();
    int hit = 0;
    int ch;

    do {
        System.out.println("\n--------MENU---------");
        System.out.println("1.FIFO");
        System.out.println("2.Optimal");
        System.out.println("3.Exit");
        System.out.print("Enter your choice : ");
        ch = sc.nextInt();

        switch (ch) {
            case 1:
                FIFO que = new FIFO(size);
                System.out.println("FIFO : ");
                for (int i = 0; i < n; i++) {
                    if (que.search(-1)) {
                        que.enque(pages[i]);
                        que.display();
                    } else {
                        if (!que.search(pages[i])) {
```

```
                que.deque();
                que.enque(pages[i]);
                que.display();
            } else {
                que.display();
                hit++;
            }
        }
    }
    System.out.println("Total Hits : " + hit);
    System.out.println("Total Faults : " + (n - hit));
    System.out.println();
    break;

case 2:
    int opti[] = new int[size];
    for (int i = 0; i < size; i++)
        opti[i] = -1;
    int i = 0;
    hit = 0;
    System.out.println("\nOptimal : ");
    for (int j = 0; j < size; j++) {
        if (opti[j] == -1) {
            if (i < n) {
                opti[j] = pages[i++];
                display(opti);
            } else
                break;
        }
    }

    for (; i < n; i++) {
        if (!search(opti, pages[i]))
```

```
                        opti[findOP(opti, pages, i)] = pages[i];
                    else
                        hit++;
                    display(opti);
                }
                System.out.println("Total Hits : " + hit);
                System.out.println("Total Faults : " + (n - hit));
                System.out.println();
                break;
            }
        } while (ch != 3);

        sc.close();
    }
}
```

Output:

```
Enter Size : 3

Enter Number of pages : 20

Enter 20 Pages : 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1


--------MENU---------

1.FIFO

2.Optimal

3.Exit

Enter your choice : 1

FIFO :

  7 -1 -1

  7  0 -1
  7  0  1
  2  0  1
  2  0  1
  2  3  1
  2  3  0
  4  3  0
  4  2  0
  4  2  3
  0  2  3
  0  2  3
  0  2  3
  0  1  3
  0  1  2
  0  1  2
  0  1  2
  7  1  2
  7  0  2
  7  0  1
Total Hits : 5
Total Faults : 15
Optimal :
  7 -1 -1
  7  0 -1
  7  0  1
```

```
 2  0  1

 2  0  1

 2  0  3

 2  0  3

 2  4  3

 2  4  3

 2  4  3

 2  0  3

 2  0  3

 2  0  3

 2  0  1

 2  0  1

 2  0  1

 2  0  1

 7  0  1

 7  0  1

 7  0  1


Total Hits : 11

Total Faults : 9
```