Exp:7

FCFS.java

```
import java.util.HashSet;
import java.util.LinkedList;
import java.util.Queue;
import java.util.Scanner;

public class FCFS {

	private Scanner sc;

	public void execute()
	{
		sc = new Scanner(System.in);
		System.out.println("Enter Number of Pages:");
		int numPages=sc.nextInt();
		int pages[]=new int[numPages];
		System.out.println("Enter Reference String: ");
		for(int i=0;i<numPages;i++)
		{
			pages[i]=sc.nextInt();
		}

		System.out.println("Enter Number of Frames");
		int capacity=sc.nextInt();

		//To represent set of current pages
		HashSet<Integer> frames=new HashSet<>(capacity);

		//To store pages o=in FIFO manner
		Queue<Integer> index=new LinkedList<>();
		int pageFaults=0;
```

```
int hits=0;
for(int i=0;i<numPages;i++)
{
	if(frames.size()<capacity) //check if set can hold n=more pages
	{
		//IF page not present insert into set and increment pagefault
		if(!frames.contains(pages[i]))
		{
			frames.add(pages[i]);
			index.add(pages[i]); //push current page into queue
			pageFaults++;
			// System.out.println(pageFaults);
			// frames.forEach(System.out::print);
			for(int j:index)
				System.out.print(j+"\t");
			System.out.println();
		}
		else
		{
			hits++;
		}
	}
	else //set is full,need replacement
	{
		if(!frames.contains(pages[i])) //frame is not present present
		{
			int val=index.peek();
			index.poll();
			frames.remove(val);

			frames.add(pages[i]);
			index.add(pages[i]);
```

```
                                        pageFaults++;
                                        for(int j:index)
                                                System.out.print(j+"\t");
                                        System.out.println();
                                }
                                else //frame is present in set
                                {
                                        hits++;
                                }
                        }
                }

                System.out.println("Number of Page Faults : "+pageFaults);
                System.out.println("Hits:\t"+hits);
                System.out.println("hit ratio: "+((double)hits/(double)numPages));

        }
}
```

Output:

```
Enter Number of Pages:
10
Enter Reference String:
2
2
5
7
2
3
8
4
5
1
Enter Number of Frames
22
2
2
25
257
257
2357
23578
234578
234578
```

1234578
Number of Page Faults : 7
Hits: 3
hit ratio: 0.3

LRU.java

```
import java.util.HashMap;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Scanner;

public class LRU {
	private Scanner sc;

	public void execute()
	{
		sc = new Scanner(System.in);
		System.out.println("Enter Number of Pages:");
		int numPages=sc.nextInt();
		int pages[]=new int[numPages];
		System.out.println("Enter Reference String: ");
		for(int i=0;i<numPages;i++)
		{
			pages[i]=sc.nextInt();
		}

		System.out.println("Enter Number of Frames");
		int capacity=sc.nextInt();

		//To represent set of current pages
		HashSet<Integer> frames=new HashSet<>(capacity);

		//To store LRU Indexes of pages //<key=Page,value=index>
		HashMap<Integer,Integer>  index=new HashMap<>();
		int pageFaults=0;
		int hits=0;
		for(int i=0;i<numPages;i++)
		{
			if(frames.size()<capacity) //check if set can hold n=more pages
			{
				//IF page not present insert into set and increment pagefault
				if(!frames.contains(pages[i]))
				{
					frames.add(pages[i]);
					index.put(pages[i],i); //push current page into queue
					pageFaults++;

					/*for(int j:index)
						System.out.print(j+"\t");
					System.out.println();*/
				}
				else
				{
					hits++;
```

```
                        index.put(pages[i],i);
                    }
                }
                else //set is full,need replacement
                {
                    if(!frames.contains(pages[i])) //frame is not present present
                    {
                        int lru=Integer.MAX_VALUE;
                        int val=Integer.MIN_VALUE;

                        Iterator<Integer> itr=frames.iterator();
                        while(itr.hasNext())
                        {
                            int temp=itr.next();
                            if(index.get(temp)<lru)
                            {
                                lru=index.get(temp);
                                val=temp;
                            }
                        }

                        frames.remove(val);
                        frames.add(pages[i]);
                        pageFaults++;
                        index.put(pages[i], i);
                    }
                    else //frame is present in set
                    {
                        hits++;
                        index.put(pages[i],i);
                    }
                }
                frames.forEach(System.out::print);
                System.out.println();
            }

            System.out.println("Number of Page Faults : "+pageFaults);
            System.out.println("Hits:\t"+hits);
            System.out.println("hit ratio: "+((double)hits/(double)numPages));

    }
```

Output:

```
Enter Number of Pages:
10
Enter Reference String:
2
3
5
7
1
5
6
2
```

```
3
2
Enter Number of Frames
44
2
23
235
2357
12357
12357
123567
123567
123567
123567
Number of Page Faults : 6
Hits:  4
hit ratio: 0.4
```

Optimal.java

```
import java.util.HashMap;

import java.util.HashSet;

import java.util.Iterator;

import java.util.Scanner;


public class Optimal {

	private Scanner sc;


	public void execute()

	{

		sc = new Scanner(System.in);

		 System.out.println("Enter Number of Pages:");

		 int numPages=sc.nextInt();

		 int pages[]=new int[numPages];

		 System.out.println("Enter Reference String: ");

		 for(int i=0;i<numPages;i++)

		 {

			pages[i]=sc.nextInt();

		 }
```

```
System.out.println("Enter Number of Frames");
int capacity=sc.nextInt();

HashSet<Integer> frames=new HashSet<>();

HashMap<Integer, Integer> index=new HashMap<>();
int pagefaults=0;
int hits=0;
for(int i=0;i<numPages;i++)
{
	if(frames.size()<capacity)
	{
		if(!frames.contains(pages[i]))
		{
			pagefaults++;
			frames.add(pages[i]);

			//finding next access of page
			int farthest=nextIndex(pages, i);

			index.put(pages[i], farthest);

		}
		else
		{
			hits++;
			index.put(pages[i], nextIndex(pages,i));
		}
	}
	else
	{
		if(!frames.contains(pages[i]))
```

```
        {
            int farthest=-1;
            int val=0;

            Iterator<Integer> itr=frames.iterator();
            while(itr.hasNext())
            {
                int temp=itr.next();

                if(index.get(temp)>farthest)
                {
                    farthest=index.get(temp);
                    val=temp;
                }
            }

            frames.remove(val);
            index.remove(farthest);
            frames.add(pages[i]);
            pagefaults++;
            int nextUse=nextIndex(pages, i);
            index.put(pages[i],nextUse);
        }
        else
        {
            hits++;
            index.put(pages[i], nextIndex(pages, i));
        }

    }
    frames.forEach(System.out::print);
    System.out.println();
```

```
                }
                System.out.println("Number of Page Faults : "+pagefaults);
                System.out.println("Hits:\t"+hits);
                System.out.println("hit ratio: "+((double)hits/(double)numPages));
        }
        public static int nextIndex(int pages[],int curIndex)
        {
                int farthest=curIndex;
                int currentPage=pages[curIndex];
                int j=farthest;
                for(j=farthest+1;j<pages.length;j++)
                {
                        if(pages[j]==currentPage)
                        {
                                farthest=j;
                                return farthest; //5 6 7 8 9
                        }
                }
                return Integer.MAX_VALUE;
        }
}
```

Output

```
Enter Number of Pages:
10
Enter Reference String:
1
3
5
2
6
2
8
2
3
4
Enter Number of Frames
```

```
33
1
13
135
1235
12356
12356
123568
123568
123568
1234568
Number of Page Faults : 7
Hits:  3
hit ratio: 0.3
```

PageReplacement.java

```
public class PageReplacement {

	public static void main(String[] args) {

		FCFS fcfs=new FCFS();
		//fcfs.execute();

		LRU lru=new  LRU();
		//lru.execute();

		Optimal optimal=new Optimal();
		optimal.execute();
	}

}
```