```
/*******************************************************************************
Write a program to implement following disk scheduling algorithms:
First Come First Serve (FCFS), Shortest Seek Time First (SSTF)

********************************************************************************/
import java.util.*;
class node
{
   //dist is the distance between the track and the head position
   int dist=0;
   boolean accessed=false;
   //accessed becomes true if track has been accessed

}
class SSTF
{
    static void calculateDifference(ArrayList<Integer> queue,node difference[],int head)
   {
      for(int i=0;i<difference.length;i++)
      {
         difference[i].dist=Math.abs(queue.get(i)-head);
      }
   }
   //find unaccessed track at min distance from the head
   static int findMin(node diff[])
   {
      int index=-1,minimum=1000;
      for(int j=0;j<diff.length;j++)
      {
         if(!diff[j].accessed && minimum>diff[j].dist)
         {
            minimum=diff[j].dist;
            index=j;
         }

      }
      return(index);
   }
    static void SSTFA(ArrayList<Integer> request,int head)
   {
       int seek_count = 0;

        //create array of objects of class node
       node diff[] = new node[request.size()];

       // initialize array
      for(int i = 0; i < diff.length; i++)
        diff[i] = new node();

       // stores sequence in which disk access is done
      int[] seek_sequence = new int[request.size() + 1];

      for(int i=0;i<request.size();i++)
      {
            seek_sequence[i]=head;
            calculateDifference(request,diff,head);

            int ind=findMin(diff);

            diff[ind].accessed = true;
            // increase the total count
            seek_count += diff[ind].dist;

            // accessed track is now new head
            head = request.get(ind);
        }

      seek_sequence[seek_sequence.length - 1] = head;

        System.out.println("Total distance travelled by the head = "+seek_count);

        System.out.println("Seek Sequence is");

        for (int i = 0; i < seek_sequence.length; i++)
            System.out.print(seek_sequence[i]+" ");
    }    //close void SSTF


}//close class SSTF
public class Main
{
    static void FCFS(ArrayList<Integer> alist,int head)
    {
        int distance,cur_track=0,seekCount=0;
        System.out.println("\nDistance calculation:");

        for(int i=0;i<alist.size();i++)
        {
            cur_track=alist.get(i);
            System.out.print(cur_track+"-"+head+"=");

            distance=Math.abs(cur_track-head);

            System.out.print(distance+"\n");

            seekCount+=distance;

            //accessed track is now head
            head=cur_track;
        }

        System.out.print("\nTotal distance travelled by head ="+seekCount);
    }

 public static void main(String[] args)
 {
     int choice,head,maxNo,sizeC,inputS;
     char no;
     Scanner sc=new Scanner(System.in);
     System.out.println("\nEnter maximum number of cylinders:");
     maxNo=sc.nextInt();
     System.out.println("\nEnter total number of cylinders to be accessed:");
     sizeC=sc.nextInt();
     System.out.println("Enter starting location of head:");
     head=sc.nextInt();

     ArrayList<Integer> arr=new ArrayList<Integer>(sizeC);
     System.out.println("\nEnter the sequence of cylinders:");

     for(int i=0;i<sizeC;i++)
     {
        inputS=sc.nextInt();
        arr.add(inputS);
     }
     System.out.println("\nSequence is:");

        for (int i = 0; i < arr.size(); i++)
            System.out.print(arr.get(i)+"  ");

        System.out.println("\n Starting location of head:"+head+"\n");
     System.out.println("\n Maximum number of cylinders:"+maxNo+"\n");

     do
     {
   System.out.println("***DISK SCHEDULING***");
        System.out.println("\n1. First Come, First Serve (FCFS)");
        System.out.println("\n2. Shortest-Seek-Time-First (SSTF)");
        System.out.println("\n3. Exit menu");
        System.out.println("\nEnter your choice:");
        choice=sc.nextInt();
        switch(choice)
        {
        case 1:FCFS(arr,head);
            break;

        case 2:SSTF.SSTFA(arr,head);
            break;

        case 3:System.out.println("\nExit menu");
            break;

        default:System.out.println("\nInvalid Choice!!!");
             break;
        }
        System.out.println("\nDo you wish to continue?(y/n):");
        no=sc.next().charAt(0);


     }while(no=='y'||no=='Y');

 }
}
```

```
/*OUTPUT

Enter maximum number of cylinders:
200

Enter total number of cylinders to be accessed:
8
Enter starting location of head:
50

Enter the sequence of cylinders:
176
79
34
60
92
11
41
114

Sequence is:
176  79  34  60  92  11  41  114
 Starting location of head:50


 Maximum number of cylinders:200

***DISK SCHEDULING***

1. First Come, First Serve (FCFS)

2. Shortest-Seek-Time-First (SSTF)

3. Exit menu

Enter your choice:
2
Total distance travelled by the head = 204
Seek Sequence is
50 41 34 11 60 79 92 114 176
Do you wish to continue?(y/n):
y
***DISK SCHEDULING***

1. First Come, First Serve (FCFS)

2. Shortest-Seek-Time-First (SSTF)

3. Exit menu

Enter your choice:
1

Distance calculation:
```

```
176-50=126
79-176=97
34-79=45
60-34=26
92-60=32
11-92=81
41-11=30
114-41=73

Total distance travelled by head =510
Do you wish to continue?(y/n):
y
***DISK SCHEDULING***

1. First Come, First Serve (FCFS)

2. Shortest-Seek-Time-First (SSTF)

3. Exit menu

Enter your choice:
4

Invalid Choice!!!

Do you wish to continue?(y/n):
n
*/
```