1)

```
import java.util.Arrays;
import java.util.Scanner;

interface Sortable	{
	void sort(int[] arr, int n);
}

class BubbleSort implements Sortable	{

	public void sort(int[] arr, int n) {

		for(int i=0; i<n-1; i++)  {
			for(int j=0; j<n-i-1; j++)	{
				if(arr[j] > arr[j+1]) {
					int temp = arr[j];
					arr[j] = arr[j+1];
					arr[j+1] = temp;
				}
			}
		}

		System.out.println(Arrays.toString(arr));

	}

}

class QuickSort implements Sortable{

	int partition(int[] arr, int low, int high)  {

		int pivot = arr[low];
		int i=low, j = high;

		while(i<j)	{
			while(i <= high - 1 && arr[i] <= pivot)  i++;
			while(j >= low + 1 && arr[j] > pivot)    j--;

			if(i<j)	{
				int temp = arr[i];
				arr[i] = arr[j];
				arr[j] = temp;
			}
```

```
            }

            int temp = arr[low];
            arr[low] = arr[j];
            arr[j] = temp;

            return j;

        }

        void quickSort(int[] arr, int low, int high)        {
            if(low >= high) return;
            int pivot = partition(arr, low, high);
            quickSort(arr, low, pivot-1);
            quickSort(arr, pivot+1, high);
        }

        public void sort(int[] arr, int n )          {
            quickSort(arr, 0, n-1);
            System.out.println(Arrays.toString(arr));
        }

}

class MergeSort implements Sortable          {

        void merge(int[] arr, int low, int mid, int high) {
            int[] temp = new int[high - low + 1];
            int i = low, j = mid+1, k = 0;
            while(i <= mid && j <= high)   {
                if(arr[i] < arr[j]) temp[k++] = arr[i++];
                else     temp[k++] = arr[j++];
            }

            while(i <= mid)temp[k++] = arr[i++];
            while(j <= high)temp[k++] = arr[j++];

            for(i=0; i<temp.length; i++)    {
                arr[low+i] = temp[i];
            }
        }

        void mergeSort(int[] arr, int low, int high)       {
            if(low >= high) {
```

```
                return;
            }

            int mid = (low + high) / 2;
            mergeSort(arr, low, mid);
            mergeSort(arr, mid+1, high);
            merge(arr, low, mid, high);
        }

        public void sort(int[] arr, int n) {
            mergeSort(arr, 0, n-1);
            System.out.println(Arrays.toString(arr));
        }

}

public class Assignment        {
        public static void main(String[] args)  {
            Scanner sc = new Scanner(System.in);
            System.out.println("Enter the length of the array: ");
            int n = sc.nextInt();

            int[] arr = new int[n];
            for(int i=0; i<n; i++)     {
                System.out.println("Enter the value for " + i + " index :");
                arr[i] = sc.nextInt();
            }

            System.out.println("Enter the sort that you want to do: ");
            System.out.println("1. Bubble Sort");
            System.out.println("2. Quick Sort");
            System.out.println("3. Merge Sort");

            int choice = sc.nextInt();
            Sortable sort;
            switch(choice) {

            case 1:
                sort = new BubbleSort();
                sort.sort(arr, n);
                break;

            case 2:
                sort = new QuickSort();
```

```
                        sort.sort(arr, n);
                        break;

                case 3:
                        sort = new MergeSort();
                        sort.sort(arr, n);
                        break;

                }
        }
}

2)
import java.util.Scanner;

interface Playable      {
        default String play()   {
                return "The player is playing";
        }

        default String pause() {
                return "The player is paused";
        }
        default String stop()   {
                return "The player is stopped";
        }
}

class MP3Player implements Playable         {

        public String play()    {
                return "The mp3 player is playing";
        }

        public String pause()  {
                return "The mp3 player is paused";
        }

        public String stop()    {
                return "The mp3 player is stopped";
        }

}
```

```
class CDPlayer implements Playable{

	public String play()	{
		return "The CD player is playing";
	}

	public String pause()	{
		return "The CD player is paused";
	}

	public String stop()	{
		return "The CD player is stopped";
	}

}

class StreamingPlayer implements Playable {

	public String play()	{
		return "The streaming player is playing";
	}

	public String pause()	{
		return "The streaming player is paused";
	}

	public String stop()	{
		return "The streaming player is stopped";
	}

}

public class Assignment		{
	public static void main(String[] args)	{

		Scanner sc = new Scanner(System.in);
		do	{
			System.out.println("What player do you want to access ?");
			System.out.println("1. MP3 player");
			System.out.println("2. CD player");
			System.out.println("3. Streaming player");
			System.out.println("4. Quit");

			Playable player;
```

```
int ch = sc.nextInt();
switch(ch)	{
		case 1:
			player = new MP3Player();
			break;

		case 2:
			player = new CDPlayer();
			break;

		case 3:
			player = new StreamingPlayer();
			break;

		case 4:
			return;

		default:
			System.out.println("Please enter valid input");
			continue;
}

boolean inThePlayer = true;

do {
		System.out.println("What do you want to do?");
		System.out.println("1. Play");
		System.out.println("2. Pause");
		System.out.println("3. Stop");

		int choice = sc.nextInt();
		switch(choice) {
			case 1:
				System.out.println(player.play());
				break;

			case 2:
				System.out.println(player.pause());
				break;
			case 3:
				System.out.println(player.stop());
				inThePlayer = false;
				break;
```

```
                                default:
                                        System.out.println("Please enter valid input");
                                        continue;
                                }
                        }while(inThePlayer);

                }while(true);


        }
}
```

3)