2.

```
public class SelectionSorting{
    public static void selectionSort(int[] arr){
        for (int i = 0; i < arr.length - 1; i++) {
            int index = i;
            for (int j = i + 1; j < arr.length; j++){
                if (arr[j] < arr[index]){
                    index = j;
                }
            }
            int smlNumber = arr[index];
            arr[index] = arr[i];
            arr[i] = smlNumber;
        }
    }
    public static void main(String a[]) {
        int[] arr1 = {11,34,2,3,33,12,59,20};
        System.out.println("Before");
        for(int i:arr1){
            System.out.print(i+" ");
        }
        System.out.println();

        selectionSort(arr1);

        System.out.println("After");
        for(int i:arr1){
            System.out.print(i+" ");
        }
    }
}
```

3.

```
public class InsertionSorting {
    public static void insertionSort(int array[]) {
        int n = array.length;
        for (int j = 1; j < n; j++) {
            int key = array[j];
            int i = j-1;
            while ( (i > -1) && ( array [i] > key ) ) {
                array [i+1] = array [i];
                i--;
            }
            array[i+1] = key;
        }
    }

    public static void main(String a[]){
        int[] arr1 = {9,5,3,7,1,9,4};
        System.out.println("Before ");
        for(int i:arr1){
            System.out.print(i+" ");
```

```
        }
        System.out.println();

        insertionSort(arr1);

        System.out.println("After ");
        for(int i:arr1){
            System.out.print(i+" ");
        }
    }
}
```

1.

```
class MergeSort {
      void merge(int arr[], int l, int m, int r) {
            int n1 = m - l + 1;
            int n2 = r - m;
            int L[] = new int[n1];
            int R[] = new int[n2];
            for (int i = 0; i < n1; ++i)
                  L[i] = arr[l + i];
            for (int j = 0; j < n2; ++j)
                  R[j] = arr[m + 1 + j];
            int i = 0, j = 0;
            int k = l;
            while (i < n1 && j < n2) {
                  if (L[i] <= R[j]) {
                        arr[k] = L[i];
                        i++;
                  }
                  else {
                        arr[k] = R[j];
                        j++;
                  }
                  k++;
            }
            while (i < n1) {
                  arr[k] = L[i];
                  i++;
                  k++;
            }
            while (j < n2) {
                  arr[k] = R[j];
                  j++;
                  k++;
            }
      }
      void sort(int arr[], int l, int r)
      {
            if (l < r) {
                  int m = l + (r - l) / 2;
                  sort(arr, l, m);
                  sort(arr, m + 1, r);
                  merge(arr, l, m, r);
            }
```

```
      }
      static void printArray(int arr[])
      {
            int n = arr.length;
            for (int i = 0; i < n; ++i)
                  System.out.print(arr[i] + " ");
            System.out.println();
      }
      public static void main(String args[])
      {
            int arr[] = { 12, 11, 13, 5, 6, 7 };

            System.out.println("Given ");
            printArray(arr);

            MergeSort ob = new MergeSort();
            ob.sort(arr, 0, arr.length - 1);

            System.out.println("Sorted array");
            printArray(arr);
      }
}
```

4.

```
public class BubbleSort {
    static void bubbleSort(int[] arr) {
        int n = arr.length;
        int temp = 0;
        for(int i=0; i < n; i++){
        for(int j=1; j < (n-i); j++){
        if(arr[j-1] > arr[j]){

        temp = arr[j-1];
        arr[j-1] = arr[j];
        arr[j] = temp;
            }

        }
    }

}
    public static void main(String[] args) {
    int arr[] ={77,788,282,999,123,27,89};

    System.out.println("Array Before");
    for(int i=0; i < arr.length; i++){
    System.out.print(arr[i] + " ");
    }
    System.out.println();
    bubbleSort(arr);
    System.out.println("Array After ");
    for(int i=0; i < arr.length; i++){
    System.out.print(arr[i] + " ");
        }
```

```
    }
}
```

5.

```
class Stack {
    private int arr[];
    private int top;
    private int capacity;
    Stack(int size) {
    arr = new int[size];
    capacity = size;
    top = -1;
    }
    public void push(int x)
    {
        if (isFull())
        {
            System.out.println("Terminated program");
            System.exit(-1);
        }

        System.out.println("Insert " + x);
        arr[++top] = x;
    }
    public int pop()
    {
        if (isEmpty())
        {
            System.out.println("Terminated");
            System.exit(-1);
        }

        System.out.println("Removing " + peek());
        return arr[top--];
    }
    public int peek()
    {
        if (!isEmpty()) {
            return arr[top];
        }
        else {
            System.exit(-1);
        }

        return -1;
    }
    public int size() {
        return top + 1;
    }
    public boolean isEmpty() {
        return top == -1;
    }
    public boolean isFull() {
        return top == capacity - 1;
```

```
    }
}

class Main
{
    public static void main (String[] args)
    {
        Stack stack = new Stack(3);

        stack.push(1);
        stack.push(2);
        stack.pop();
        stack.pop();

        stack.push(3);

        System.out.println("top element " + stack.peek());
        System.out.println("stack size " + stack.size());

        stack.pop();


        if (stack.isEmpty()) {
            System.out.println("stack empty");
        }
        else {
            System.out.println("not empty");
        }
    }
}
```

6.

```
class Queue {

    private static int frnt, rear, capacity;
    private static int queue[];
    Queue(int size) {
        front = rear = 0;
        capacity = size;
        queue = new int[capacity];
    }
    static void queueEnqueue(int item)  {
        if (capacity == rear) {
            System.out.printf("full");
            return;
        }
        else {
            queue[rear] = item;
            rear++;
        }
        return;
    }
    static void queueDequeue()  {
        if (front == rear) {
```

```
            System.out.printf("empty");
            return;
        }
        else {
            for (int i = 0; i < rear - 1; i++) {
                queue[i] = queue[i + 1];
            }
            if (rear < capacity)
                queue[rear] = 0;
            rear--;
        }
        return;
    }
    static void queueDisplay()
    {
        int i;
        if (front == rear) {
            System.out.printf("Empty");
            return;
        }
        for (i = front; i < rear; i++) {
            System.out.printf(" %d , ", queue[i]);
        }
        return;
    }
    static void queueFront()
    {
        if (front == rear) {
            System.out.printf("Empty");
            return;
        }
        System.out.printf("Front Element : %d", queue[front]);
        return;
    }
}

public class Queue{
    public static void main(String[] args) {
        Queue q = new Queue(4);

        System.out.println("Initial:");
        q.queueDisplay();
        q.queueEnqueue(12);
        q.queueEnqueue(20);
        q.queueEnqueue(40);
        q.queueEnqueue(50);
        System.out.println("after Enqueue:");
        q.queueDisplay();
        q.queueFront();
        q.queueEnqueue(90);
        q.queueDisplay();
        q.queueDequeue();
        q.queueDequeue();
        System.out.printf("after two operations:");
```

```
        q.queueDisplay();
        q.queueFront();
    }
}
```

9.

```
class QuickSort {
      static void swap(int[] arr, int i, int j)
      {
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
      }
      static int partition(int[] arr, int low, int high)
      {
            int pivot = arr[high];
            int i = (low - 1);
            for (int j = low; j <= high - 1; j++) {
                  if (arr[j] < pivot) {
                        i++;
                        swap(arr, i, j);
                  }
            }
            swap(arr, i + 1, high);
            return (i + 1);
      }
      static void quickSort(int[] arr, int low, int high)
      {
            if (low < high) {
                  int pi = partition(arr, low, high);
                  quickSort(arr, low, pi - 1);
                  quickSort(arr, pi + 1, high);
            }
      }
      static void printArray(int[] arr, int size)
      {
            for (int i = 0; i < size; i++)
                  System.out.print(arr[i] + " ");

            System.out.println();
      }
      public static void main(String[] args)
      {
            int[] arr = { 10, 7, 8, 9, 1, 5 };
            int n = arr.length;

            quickSort(arr, 0, n - 1);
            System.out.println("Sorted array: ");
            printArray(arr, n);
      }
}
```

10.

```
public class HeapSort {
	public void sort(int arr[])
	{
		int n = arr.length;
		for (int i = n / 2 - 1; i >= 0; i--)
			heapify(arr, n, i);
		for (int i=n-1; i>=0; i--) {
			int temp = arr[0];
			arr[0] = arr[i];
			arr[i] = temp;
			heapify(arr, i, 0);
		}
	}
	void heapify(int arr[], int n, int i) {
		int largest = i;
		int l = 2*i + 1;
		int r = 2*i + 2;
		if (l < n && arr[l] > arr[largest])
			largest = l;
		if (r < n && arr[r] > arr[largest])
			largest = r;
		if (largest != i)
		{
			int swap = arr[i];
			arr[i] = arr[largest];
			arr[largest] = swap;
			heapify(arr, n, largest);
		}
	}
	static void printArray(int arr[])
	{
		int n = arr.length;
		for (int i=0; i<n; ++i)
			System.out.print(arr[i]+" ");
		System.out.println();
	}
	public static void main(String args[])
	{
		int arr[] = {22, 29, 10, 45, 9, 2};
		int n = arr.length;

		HeapSort ob = new HeapSort();
		ob.sort(arr);

		System.out.println("array is : ");
		printArray(arr);
	}
}
```