1. Bubble Sorting:

```
public class BubbleSorting {
	public void bubbleSorting(int[] arr) {
		int temp;
		for (int i = 0; i < arr.length; i++) {
			for (int j = 1; j < arr.length; j++) {
				if(arr[j - 1] > arr[j]) {
					temp = arr[j - 1];
					arr[j - 1] = arr[j];
					arr[j] = temp;
				}
			}
		}

		System.out.println("Bubble Sorting: ");
		for (int i = 0; i < arr.length; i++) {
			System.out.println("\t" + arr[i]);
		}
	}
}
```

2. Selection Sorting:

```
public class SelectionSorting {
	static void selectionSorting(int[] arr) {
		int min, temp;
		for (int i = 0; i < arr.length - 1; i++) {
			min = i;
			for (int j = i + 1; j < arr.length; j++) {
				if (arr[min] > arr[j]) {
					min = j;
				}
			}
			temp = arr[min];
			arr[min] = arr[i];
			arr[i] = temp;
		}
	}

	public static void main(String[] args) {
		int[] arr = {12, 3, 5, 1, 18, 23, 4, 9};
		System.out.println("Unsorted Array:\n" + Arrays.toString(arr));
```

```
        selectionSorting(arr);
        System.out.println("Sorted Array:\n" + Arrays.toString(arr));
    }
}
```

3. Insertion Sorting:

```
public class InsertedSorting {
    public static void insertedSorting(int[] arr) {

        int size = arr.length;
        for (int i = 1; i < size; i++) {
            int key = arr[i];
            int j = i - 1;

            while (j >= 0 && key < arr[j]) {
                arr[j + 1] = arr[j];
                --j;
            }
            arr[j + 1] = key;

        }
    }

    public static void main(String[] args) {

        int[] arr = {15, 19, 2, 6, 1, 23, 8, 37};
        System.out.println("Unsorted array: ");
        System.out.println(Arrays.toString(arr));
        insertedSorting(arr);
        System.out.println("Sorted array is: \n" + Arrays.toString(arr));

    }
}
```

4. Shell Sorting:

```
public class ShellSort {

    public static void shellSort(int[] arr, int n) {
        for (int interval = n / 2; interval > 0; interval /= 2) {
            for (int i = interval; i < n; i += 1) {
                int temp = arr[i];
                int j;

                for (j = i; j >= interval && arr[j - interval] > temp; j -= interval) {
                    arr[j] = arr[j - interval];
                }
                arr[j] = temp;
            }
        }
    }

    public static void main(String[] args) {
        int[] arr = {12, 1, 5, 15, 2, 8, 6, 18};
        System.out.println("Unsorted Array:\n" + Arrays.toString(arr));
        shellSort(arr, arr.length);
        System.out.println("Sorted Array:\n" + Arrays.toString(arr));
    }
}
```

5. Merge Sorting:

```
public class MergeSort {
    static 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[] M = new int[n2];

        for (int i = 0; i < n1; i++) {
            L[i] = arr[l + i];
        }

        for (int i = 0; i < n2; i++) {
            M[i] = arr[m + i + 1];
        }

        int i = 0, j = 0, k = l;

        while(i < n1 && j < n2) {
            if (L[i] < M[j]) {
                arr[k] = L[i];
                ++i;
            } else {
                arr[k] = M[j];
```

```
                    ++j;
                }
                ++k;
            }

            while (i < n1) {
                arr[k] = L[i];
                ++i;
                ++k;
            }

            while (j < n2) {
                arr[k] = M[j];
                ++j;
                ++k;
            }

        }

        static void mergeSort(int[] arr, int l, int r) {
            if (l < r) {
                int m = (l + r) / 2;
                mergeSort(arr, l, m);
                mergeSort(arr, m + 1, r);
                merge(arr, l, m, r);
            }
        }

        public static void main(String[] args) {
            int[] arr = {12, 3, 5, 1, 18, 23, 4, 9};
            System.out.println("Unsorted Array:\n" + Arrays.toString(arr));
            mergeSort(arr, 0, arr.length - 1);
            System.out.println("Sorted Array:\n" + Arrays.toString(arr));
        }
    }
```

6. Quick Sorting:

```
public class QuickSorting {
    public 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);
        }
    }

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

        int pivot = arr[high];
        int i = low - 1;

        for (int j = low; j < high; ++j) {

            if (arr[j] <= pivot) {

                i++;
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;

            }
        }

        int temp = arr[i + 1];
        arr[i + 1] = arr[high];
        arr[high] = temp;
        return (i + 1);

    }

    public static void main(String[] args) {
        int[] arr = {15, 19, 2, 6, 1, 23, 8, 37};
        System.out.println("Unsorted array: ");
        System.out.println(Arrays.toString(arr));
        quickSort(arr, 0, arr.length - 1);
        System.out.println("Sorted array is: \n" + Arrays.toString(arr));

    }
}
```

7. Counting Sorting:

```
public class CountingSorting {
    static void countingSort(int[] arr, int n) {
        int[] output = new int[n + 1];
        int max = arr[0];
        for (int i = 1; i < n; i++) {
            if (arr[i] > max) {
                max = arr[i];
            }
        }

        int[] count = new int[max + 1];

        for (int i = 0; i < n; i++) {
            ++count[arr[i]];
        }

        for (int i = 1; i <= max; i++) {
            count[i] += count[i - 1];
        }

        for (int i = n - 1; i >= 0; --i) {
            output[count[arr[i]] - 1] = arr[i];
            --count[arr[i]];
        }

        for (int i = 0; i < n; i++) {
            arr[i] = output[i];
        }
    }

    public static void main(String[] args) {
        int[] arr = {15, 2, 6, 1, 23, 8, 37};
        System.out.println("Unsorted array: ");
        System.out.println(Arrays.toString(arr));
        countingSort(arr, arr.length);
        System.out.println("Sorted array is: \n" + Arrays.toString(arr));
    }
}
```

8. Radix Sorting:

```
public class RadixSort {
        static void radixSort(int[] arr, int n) {
            int max = getMax(arr, n);

            for (int pass = 1; max / pass > 0; pass *= 10) {
                countSort(arr, n, pass);
            }
        }

        static int getMax(int[] arr, int n) {
```

```
        int max = arr[0];
        for (int i = 1; i < n; i++) {
            if (arr[i] > max) {
                max = arr[i];
            }
        }
        return max;
    }

    static void countSort(int[] arr, int n, int pass) {
        int[] output = new int[n + 1];
        int max = arr[0];
        for (int i = 1; i < n; i++) {
            if (arr[i] > max) {
                max = arr[i];
            }
        }

        int[] count = new int[max + 1];

        for (int i = 0; i < n; i++) {
            ++count[(arr[i] / pass) % 10];
        }

        for (int i = 1; i < 10; i++) {
            count[i] += count[i - 1];
        }

        for (int i = n - 1; i >= 0; --i) {
            output[count[(arr[i] / pass) % 10] - 1] = arr[i];
            --count[(arr[i] / pass) % 10];
        }

        for (int i = 0; i < n; i++) {
            arr[i] = output[i];

        }
    }

    public static void main(String[] args) {
        int[] arr = {15, 2, 6, 1, 23, 8, 37};
        System.out.println("Unsorted array: ");
        System.out.println(Arrays.toString(arr));
        radixSort(arr, arr.length);
        System.out.println("Sorted array is: \n" + Arrays.toString(arr));
    }
}
```