● JKad28D「並べ替えデータの準備②」※JKad27D をコピペして作成

以下のメソッドを作成し、整数型配列（要素数 20）に 0 から 99 までの値を設定して表示せよ。

書式	仕様
public static void initArray(int[] array)	JKad27D から流用。
public static void showArray(int[] array)	JKad27D から流用。
public static void showArray(　int[] array, 　int start, 　int pivotNum, 　int end);	※pivotNum は start 以上、end 未満の値 以下の仕様で配列 array の各要素の値を表示する。 start 番目以上、end 番目未満は値をそのまま表示する。 さらに pivotNum に関しては値に「*」を付けて表示する。 それ以外の要素は値を表示せずに代わりに「--」を表示する。

リスト１:「並べ替えデータの準備②」（ファイル「JKad28D.java」）

```
public class JKad28D {
    public static void main(String[] args) {
        int[] array = new int[20];
        initArray(array);
        showArray(array);
        showArray(array, 5, 10, 15);        // 5 番目以上 15 番目未満を表示（10 番目には「*」を付ける）
    }
    public static void initArray(int[] array) {
        JKad27D から流用
    }
    public static void showArray(int[] array) {
        JKad27D から流用
    }
    public static void showArray(int array[], int start, int pivotNum, int end) {
        作成すること
    }
}
```

課題完成時の画面（太字は pivotNum で指定した場所、データの間にタブを入れるときれいに表示される）

```
43  39  47  81  83  19  4   94  54  50  55  32  86  64  66  51  5   55  16  8
--  --  --  --  --  19  4   94  54  50  55* 32  86  64  66  --  --  --  --  --
```

● JKad28C「選択ソート①（いちばん小さい値）」

配列の値を、選択ソートを使って昇順に並べ替える処理を作成する。まずはいちばん小さい値が先頭に来る処理を作成せよ。

選択ソート（たたき台）

書式	仕様
public static void selectionSort(　int[] array, 　int start)	配列 array の start 番目から最後尾までの間で、一番小さい値を start 番目に持ってくる（一番小さい値と start 番目の値を入れ替える）。 なお、一番小さい値を見つけたら（入れ替える前に）start 番目から最後尾までの値を表示する。このとき一番小さい値に「*」を付けること。

リスト１：「選択ソート①」（ファイル「JKad28C.java」）

```
public class JKad28C {
    public static void main(String[] args) {
        int[] array = new int[20];
        JKad28D.initArray(array);
        JKad28D.showArray(array);
        selectionSort(array, 0);          // 選択ソート（0番目から最後尾まで）
        selectionSort(array, 1);          // 選択ソート（1番目から最後尾まで）
        selectionSort(array, 2);          // 選択ソート（2番目から最後尾まで）
        JKad28D.showArray(array);
    }
    public static void selectionSort(int[] array, int start) {
        int min = start;                                    // 一番小さい値のインデックス
        一番小さい値を探す
        JKad28D.showArray(array, start, min, array.length);    // start番目から最後尾まで表示
        start番目とmin番目の入れ替え
    }
}
```

課題完成時の画面（太字はその時点での最小値）

66	39	91	90	7	53	91	45	85	30	83	71	0	64	15	82	33	9	80	98
66	39	91	90	7	53	91	45	85	30	83	71	**0***	64	15	82	33	9	80	98
—	39	91	90	**7***	53	91	45	85	30	83	71	66	64	15	82	33	9	80	98
—	—	91	90	39	53	91	45	85	30	83	71	66	64	15	82	33	**9***	80	98
0	7	9	90	39	53	91	45	85	30	83	71	66	64	15	82	33	91	80	98

「0*」が最小値。この表示のあと、先頭の「66」と入れ替わる。

● JKad28B「選択ソート②（完成！）」

配列データが昇順に並ぶように選択ソートを完成させよ。

選択ソート（完成版）

書式	仕様
public static void selectionSort(int[] array)	配列 array の要素を昇順に並べ替える。 for 文で JKad28C の selectionSort メソッドを呼び出せば OK。

リスト１:「選択ソート②」（ファイル「JKad28B.java」）

```
public class JKad28B {
    public static void main(String[] args) {
        int[] array = new int[20];
        JKad28D.initArray(array);
        JKad28D.showArray(array);
        selectionSort(array);                    // すべてのデータを昇順に並べ替える
        JKad28D.showArray(array);
    }
    public static void selectionSort(int[] array) {
        for ループを使って JKad28C.selectionSort を呼び出す
    }
}
```

課題完成時の画面（太字はその時点での最小値）

27	79	94	67	70	77	47	55	70	38	8	19	97	17	84	48	58	22	68	31
27	79	94	67	70	77	47	55	70	38	**8***	19	97	17	84	48	58	22	68	31
—	79	94	67	70	77	47	55	70	38	27	19	97	**17***	84	48	58	22	68	31
—	—	94	67	70	77	47	55	70	38	27	**19***	97	79	84	48	58	22	68	31
—	—	—	67	70	77	47	55	70	38	27	94	97	79	84	48	58	**22***	68	31
—	—	—	—	70	77	47	55	70	38	**27***	94	97	79	84	48	58	67	68	31
—	—	—	—	—	77	47	55	70	38	70	94	97	79	84	48	58	67	68	**31***
—	—	—	—	—	—	47	55	70	**38***	70	94	97	79	84	48	58	67	68	77
—	—	—	—	—	—	—	55	70	**47***	70	94	97	79	84	48	58	67	68	77
—	—	—	—	—	—	—	—	70	55	70	94	97	79	84	**48***	58	67	68	77
—	—	—	—	—	—	—	—	—	**55***	70	94	97	79	84	70	58	67	68	77
—	—	—	—	—	—	—	—	—	—	70	94	97	79	84	70	**58***	67	68	77
—	—	—	—	—	—	—	—	—	—	—	94	97	79	84	70	70	**67***	68	77
—	—	—	—	—	—	—	—	—	—	—	—	97	79	84	70	70	94	**68***	77
—	—	—	—	—	—	—	—	—	—	—	—	—	79	84	**70***	70	94	97	77
—	—	—	—	—	—	—	—	—	—	—	—	—	—	84	79	**70***	94	97	77
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	79	84	94	97	**77***
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	84	94	97	**79***
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	94	97	**84***
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	97	**94***
8	17	19	22	27	31	38	47	48	55	58	67	68	70	70	77	79	84	94	97

● JKad28A「選択ソート③（再帰版、クイックソートに向けて）」

for ループではなく再帰呼び出しを使って、選択ソートを作成せよ。

選択ソート（再帰版）

書式	仕様
public static void selectionSort(int[] array)	↓の selectionSort を呼び出す。
public static void selectionSort(int[] array, int start)	start 番目から最後尾まで昇順に並べ替える。 for 文ではなく再帰呼び出しを使うこと。

課題完成時の画面（JKad28B と同じ）

● JKad28S「クイックソート①（ピボットによる分割）」

クイックソートを使って昇順に並べ替える処理を作成する。まずは start 番目以上、end 番目未満のデータをピボットによって分割する処理を作成せよ。

ピボットによる分割

書式	仕様
public static int partition(　int[] array, 　int start, 　int end)	配列 array の start 番目以上、end 番目未満のデータをピボット未満のグループとピボット以上のグループに分割する。戻り値はピボットのインデックス。 （詳細は下記参照）

partition メソッドの処理　※先頭から順にピボットより小さい値をピボットの前に持って行く

① start 番目をピボットとする（変数 pivotNum と pivotVal を宣言し、ピボットのインデックスと値を保存する）。
② start+1 番目から end 番目（未満）まで、pivotVal より小さい値（array[i]とする）があれば以下の処理をする。

　array[pivotNum]に array[i]を代入し、pivotNum をインクリメントする。
　array[i]に array[pivotNum]を代入する。

③ ②の処理が終わったら、array[pivotNum]に pivotVal を代入する（この時点で pivotNum より前に pivotVal 未満の値が、後ろに pivotVal 以上の値が並んでいる）。
④ （今回だけの処理）配列 array の start から end までを表示する（ピボットに「*」が付くようにすること）。
⑤ pivotNum を戻り値として終了する。

リスト１：「クイックソート①」（ファイル「JKad28S.java」）

```
public class JKad28S {
    public static void main(String[] args) {
        int[] array = new int[20];
        JKad28D.initArray(array);
        JKad28D.showArray(array);
        partition(array, 0, array.length);
    }
    public static int partition(int[] array, int start, int end) {
        int pivotNum = start;                          // ピボットのインデックス
        int pivotVal = array[pivotNum];                // ピボットの値
        作成すること
    }
}
```

課題完成時の画面（太字はピボット）

```
55  4   68  39  21  2   86  81  78  57  2   11  52  30  79  45  28  55  63  15
4   39  21  2   2   11  52  30  45  28  15  55* 81  78  79  57  68  55  63  86
```

ピボット（55）未満のグループ　　ピボット（55）以上のグループ

● JKad28X1「クイックソート②（完成！）」

JKad28S で作成した partition メソッドを使って、クイックソートを完成させよ（main メソッドは JKad28B とほぼ同じ。selectionSort を quickSort に変更するだけ）。

クイックソート（完成版）

書式	仕様
public static void quickSort(int[] array)	配列 array の先頭から最後尾までを対象に並べ替えを行う。
public static void quickSort(　int[] array, 　int start, 　int end)	配列 array の start 番目以上、end 番目未満を対象に並べ替えを行う。

ヒント：
partition メソッドでは start から ennd で指定した区間を以下の 3 つに分割する。

① ピボット未満の値のグループ
② ピボット
③ ピボット以上の値のグループ

よって①と③に関してさらに partition メソッドを適用するとさらに分割（並べ替えが進む）。

課題完成時の画面（太字はピボット）

12	3	39	99	5	16	20	9	54	5	76	40	66	85	13	72	1	9	41	28
3	5	9	5	1	9	**12***	99	54	39	76	40	66	85	13	72	16	20	41	28
1	**3***	9	5	5	9	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	5	5	**9***	9	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	**5***	5	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	54	39	76	40	66	85	13	72	16	20	41	28	**99***
—	—	—	—	—	—	—	39	40	13	16	20	41	28	**54***	66	85	76	72	—
—	—	—	—	—	—	—	13	16	20	28	**39***	41	40	—	—	—	—	—	—
—	—	—	—	—	—	—	**13***	16	20	28	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	**16***	20	28	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	**20***	28	—	—	—	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	40	**41***	—	—	—	—	—	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	**66***	85	76	72	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	76	72	**85***	—
—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	72	**76***	—	—
1	3	5	5	9	9	12	13	16	20	28	39	40	41	54	66	72	76	85	99

● JKad28X2「クイックソート③（Ver.2）」※JKad28X1 をコピペして作成

あらたに partition メソッドを作成し、この partition メソッドを使ってクイックソートを行え。

ピボットによる分割（Ver.2）

書式	仕様
public static int partition(　int[] array, 　int start, 　int end)	配列 array の start 番目以上、end 番目未満のデータをピボット未満のグループとピボット以上のグループに分割する。戻り値はピボットのインデックス。 （詳細は下記参照）

partition メソッドの処理（Ver.2）　※先頭と最後尾からピボットより大きい値と小さい値を探して入れ替える

① start 番目をピボットとし、変数 left（初期値 start+1）と right（初期値 end-1）を宣言する。
② array[left]がピボットより小さい間、left をインクリメントする（ただし left が right を越えないように注意すること）。
③ array[right]がピボットより大きい間、right をデクリメントする。
④ right が left 以下になっていたら⑦へ。
⑤ array[left]と array[right]を入れ替える。
⑥ left をインクリメント、right をデクリメントし②へ戻る。
⑦ array[start]に array[right]を代入し、array[right]にピボットの値を代入する。
⑧ （今回だけの処理）配列 array の start から end までを表示する（ピボットに「*」が付くようにすること）。
⑨ right を戻り値として終了する。

リスト１:「クイックソート①」（ファイル「JKad28S.java」）

```
public class JKad28S {
    public static void main(String[] args) {
        ⋮
    }
    public static int partition(int[] array, int start, int end) {
        int pivotVal = array[start];
        int left = start + 1;              // ピボット以下グループの最後尾
        int right = end - 1;               // ピボット以上グループの先頭
        作成すること
    }
}
```

課題完成時の画面（JKad28X1 と同じ）