July 7th, 2020

Exercises for Programming Practice 2

Note:
- ✓ Do not create “module-info.java”, when you create a Java Project.
- ✓ Do not set Package name in the window “New Java Class”.
- ✓ Do not use the title of exercise for “contents of the program”. Think about “contents of the program” yourself.

The deadline for submitting the programs is 18:00 on July 9th, 2020.

The class “Exercise20.java” (although incomplete program) can be obtained from Week 10 page Resource of “Programming Practice 2” course, manaba+R. The field contains int array “list3” used by Exercise 21 and 22.

Exercise 20 (file name “Exercise20.java”)

Create the merge method (of Exercise20.java) which merges two previously sorted sequences “list1” and “list2”, with the output copied into “list”. Refer Code Fragment 12.1 using Python.

Table 1 shows the comparison between Java program and Code Fragment 12.1.

```
1  def merge(S1, S2, S):
2    """Merge two sorted Python lists S1 and S2 into properly sized list S."""
3    i = j = 0
4    while i + j < len(S):
5      if j == len(S2) or (i < len(S1) and S1[i] < S2[j]):
6        S[i+j] = S1[i]              # copy ith element of S1 as next item of S
7        i += 1
8      else:
9        S[i+j] = S2[j]              # copy jth element of S2 as next item of S
10       j += 1
```

Code Fragment 12.1: An implementation of the merge operation for Python’s array-based list class.

Table 1 the comparison between Java program and Code Fragment 12.1

Java program	Code Fragment 12.1
int[] list1	S1
int[] list2	S2
int[] list	S

The outline of “Exercise20.java” is as follows:

```
/* comments */

public class Exercise20 {
    int[] list1 = {2, 5, 8};
    int[] list2 = {1, 4, 7, 9};
    int[] list3 = {6, 8, 2, 1, 5, 3, 7}; // data for Exercise21, 22

    public static void main(String[] args) {
        new Exercise20();
    }

    public Exercise20() {
        int[] list = new int[list1.length + list2.length];
        printList("list1", list1);
        printList("list2", list2);
        merge(list1, list2, list);
        printList("list", list);
    }

    public void merge(int[] list1, int[] list2, int[] list) {
        int i = 0;
        int j = 0;

        // Complete this part

    }

    public void printList(String str, int[] list) {
        System.out.print(str + ": ");
        for(int i = 0; i < list.length; i++) {
            System.out.print(list[i] + " ");
        }
        System.out.println();
    }
}
```

Expected Output:

```
list1: 2 5 8
list2: 1 4 7 9
list: 1 2 4 5 7 8 9
```

Exercise 21 (file name “Exercise21.java”)

Create a Java program “Exercise21.java” that implements the merge-sort algorithm given in Code Fragment 12.2. Refer Table 1 to compare between Java program and the algorithm. In order to copy part of list list1 (S1) and list2 (S2), it is done after creating new list1 and list2 of appropriate size.

```
1  def merge_sort(S):
2    """Sort the elements of Python list S using the merge-sort algorithm."""
3    n = len(S)
4    if n < 2:
5      return                    # list is already sorted
6    # divide
7    mid = n // 2
8    S1 = S[0:mid]               # copy of first half
9    S2 = S[mid:n]               # copy of second half
10   # conquer (with recursion)
11   merge_sort(S1)              # sort copy of first half
12   merge_sort(S2)              # sort copy of second half
13   # merge results
14   merge(S1, S2, S)            # merge sorted halves back into S
```

Code Fragment 12.2: An implementation of the recursive merge-sort algorithm for Python’s array-based list class (using the merge function defined in Code Fragment 12.1.

The outline of “Exercise21.java” is as follows:

```
/* comments */

public class Exercise21 {
    int[] list3 = {6, 8, 2, 1, 4, 3, 7};

    public static void main(String[] args) {
        new Exercise21();
    }

    public Exercise21() {
        printList("list", list3);
        mergeSort(list3);
        printList("result", list3);
    }

    public void mergeSort(int[] list) {

        // Complete this part

        printList("list", list);
    }
    /* Copy the methods “merge” and “printList” from Exercise20*/
}
```

Expected Output:

```
list: 6 8 2 1 4 3 7
list: 2 8
list: 2 6 8
list: 1 4
list: 3 7
list: 1 3 4 7
list: 1 2 3 4 6 7 8
result: 1 2 3 4 6 7 8
```

Exercise 22 (file name “Exercise22.java”)

Create a Java program “Exercise22.java” that implements the quick-sort algorithm given in Code Fragment 12.5. Table 2 shows the comparison between Java program and Code Fragment 12.5.

```
1  def quick_sort(S):
2    """Sort the elements of queue S using the quick-sort algorithm."""
3    n = len(S)
4    if n < 2:
5      return                          # list is already sorted
6    # divide
7    p = S.first( )                    # using first as arbitrary pivot
8    L = LinkedQueue( )
9    E = LinkedQueue( )
10   G = LinkedQueue( )
11   while not S.is_empty( ):          # divide S into L, E, and G
12     if S.first( ) < p:
13       L.enqueue(S.dequeue( ))
14     elif p < S.first( ):
15       G.enqueue(S.dequeue( ))
16     else:                           # S.first() must equal pivot
17       E.enqueue(S.dequeue( ))
18   # conquer (with recursion)
19   quick_sort(L)                     # sort elements less than p
20   quick_sort(G)                     # sort elements greater than p
21   # concatenate results
22   while not L.is_empty( ):
23     S.enqueue(L.dequeue( ))
24   while not E.is_empty( ):
25     S.enqueue(E.dequeue( ))
26   while not G.is_empty( ):
27     S.enqueue(G.dequeue( ))
```

Code Fragment 12.5: Quick-sort for a sequence *S* implemented as a queue.

Table 2 the comparison between Java program and Code Fragment 12.1

Java program	Code Fragment 12.1
ArrayDeque<Integer>	LinkedQueue
less	L
equal	E
greater	G
isEmpty()	is_empty()
getFirst()	first()
add()	enqueue()
remove()	dequeue()

The outline of “Exercise22.java” is as follows:

```
/* comments */

import java.util.ArrayDeque;

public class Exercise22 {
    int[] list = {6, 8, 2, 1, 4, 3, 7};
    ArrayDeque<Integer> data = new ArrayDeque<Integer>();

    public static void main(String[] args) {
        new Exercise22();
    }

    public Exercise22() {
        for(int i = 0; i < list.length; i++) {
            data.add(list[i]);
        }
        System.out.println("original is " + data);
        quickSort(data);
        System.out.println("result is " + data);
    }

    public void quickSort(ArrayDeque<Integer> sequence) {

        // Complete this part

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

Expected Output:

```
original is [6, 8, 2, 1, 4, 3, 7]
[3, 4]
[1, 2, 3, 4]
[7, 8]
[1, 2, 3, 4, 6, 7, 8]
result is [1, 2, 3, 4, 6, 7, 8]
```