Object Oriented Programming TDA547

Krasimir Angelov, tel. 070-234-24-78

2015-08-21

The total number of points is 40. 20 points certainly guarantee a pass. 27p correspond to grade 4 and 32p to grade 5.

No other help materials except an English Dictionary are allowed. Write clean and readable Java code. Trivial syntax errors will be tolerated without affecting the grades. You don't have to comment your code unless if you really want to.

1. Read the following program:

```
public class Question1 {
    public static int[] reorder(int[] a) {
        int half = a.length/2;
        int[] b = new int[a.length];
        for (int i = 0; i < half; i ++) {
            b[i]      = a[i*2];
            b[i+half] = a[i*2+1];
        }
        return b;
    }

    public static void main(String[] args) {
        int[] a = new int[] {1,2,8,0,0,1,8,5};
        a = reorder(a);

        for (int i = 0; i < a.length; i++) {
            System.out.println(a[i]);
        }
    }
}
```

What will the program print when it is executed? (4p)

2. In this task we do simple array processing:

   - The range size of a sequence of numbers is the difference between the biggest and the smallest element in the sequence. For example the range size of 1 7 3 5 is 6 because 7-1 = 6. Implement the static method:

     ```
     public static int getRangeSize(int[] a)
     ```

     which returns the range size. The numbers in the sequence are arbitrary integers, i.e. they could be negative as well as positive. By definition the range size of an empty sequence is zero. (4p)

   - Implement a class `Question2` which can be used to test the method getRangeSize. The class should be possible to run like this:

     ```
     > java Question2 1 7 3 5
     6
     ```

     i.e. it takes the sequence of numbers from the command line arguments and prints the range size. (4p)

3. Some old computer graphics systems used the concept of turtle graphics. The "turtle" is a software object which can move over a plane and draw pictures with a pen. It can follow three commands: `turnLeft`, `turnRight` and `forward N`. The first two commands tell the turtle to turn to the left/right while the last tells it to go forward with `N` number of steps. Assuming that you have the enumeration type:

   ```
   enum Direction {NORTH, WEST, EAST, SOUTH};
   ```

   implement a class `Turtle`, which maintains the current coordinates and direction of the turtle. When a new turtle is created its initial coordinates must be X=0 and Y=0 and the direction must be NORTH. Implement the methods:

   - `void turnLeft();`
   - `void forward(int steps);`

   which correspond to two of the commands (`turnRight` is very similar and that is why we just skip it here).

   Note that, at all times, the turtle is facing a particular direction, and the behaviour of each of the commands depends on the current direction. For convenience the effect of each command depending on the current direction is shown in the table:

	turnLeft	turnRight	forward
NORTH	face WEST	face EAST	increase Y
SOUTH	face EAST	face WEST	decrease Y
EAST	face NORTH	face SOUTH	increase X
WEST	face SOUTH	face NORTH	decrease X

There are also two examples:

- `forward 10` will move it to (0,10) because initially it was at (0,0) and it was looking North. Movement to the north means increase in Y.
- `turnLeft; forward 10` will move it to (-10, 0). The difference is that now `turnLeft` has caused it to look to the West which means that `forward 10` will decrease the X coordinate instead of increasing Y as in the first example.

(8p)

4. In this task you have to implement Knuth's algorithm for shuffling a deck of cards. The deck is represented as an array of `N` integers. The algorithm proceeds in `N` steps where at each step the last `I` cards are properly shuffled while the first `(N-I)` are yet to be shuffled. This is illustrated with the example:

`1 2 3 4 5 6 7 8|`	swap 6 and 8
`1 2 3 4 5 8 7|6`	swap 7 and 2
`1 7 3 4 5 8|2 6`	swap 8 and 8
`1 7 3 4 5|8 2 6`	swap 5 and 1
`5 7 3 4|1 8 2 6`	swap 4 and 3
`5 7 4|3 1 8 2 6`	swap 4 and 4
`5 7|4 3 1 8 2 6`	swap 7 and 5
`7|5 4 3 1 8 2 6`	

Here each row represents a step in the shuffling process. The cards to the left of the bar symbol (`|`) are yet to be shuffled, while those to the right are already properly shuffled. To proceed from one step to another we pick a random card from the unshuffled part and we swap it with the card that is immediately before the bar. After that we move the bar to the left. When there is only one card before the bar then the deck is completely shuffled.

Implement this algorithm with the static method:

```
void shuffle(int[] a);
```

where the array `a` must be shuffled after the execution of the method.

(10p)

5. The file `students.txt` contains information for students and their grades in different courses. Each line contains the name of a student, followed by the number of courses that he/she has passed, and a list of codes and grades for the passed courses. For example:

   ```
   John 3 TDA547 5 DAT170 5 TDA233 4
   Eva  2 TDA547 4 DAT170 5
   ```

   Here John has passed three courses TDA547, DAT170 and TDA233 with grades 5, 5 and 4. Eva has passed two courses TDA547 and DAT170 with grades 4 and 5. Assuming that there is a class:

   ```
   public class Student {
      public Student(String name);
      public void addCourse(String code, int grade);
   }
   ```

   implement the static method:

   ```
   List loadStudents(File file) throws FileNotFoundException;
   ```

   which loads the file with students and returns a list with objects of type `Student`. In order to make things simple, you can assume that the student names consist of only a single word. The course codes are also single words. This means that it is enough to use the methods `next()` and `nextInt()` of `Scanner` to parse the content of the file.