

Object Oriented Programming TDA547

Krasimir Angelov, tel. 031 772 10 19

2016-03-16

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 void set(int k, int[] a) {
        int tmp = a[k];
        a[k] = 0;
        k += tmp;
    }

    public static void main(String[] args) {
        int[] a = new int[] {1,2,8,3,0};

        int k = 0;
        set(k,a);
        set(k,a);
        set(k,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:

- Implement the method:

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

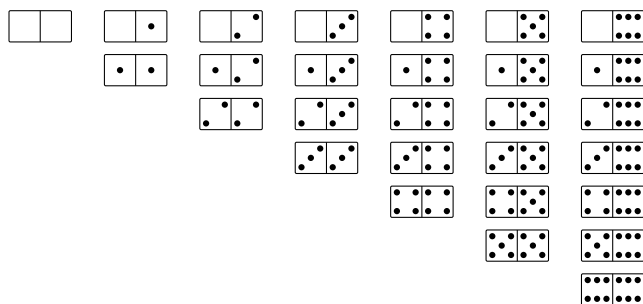
which receives in the array `a` a sequence of increasing integers. The method should return the value of the largest difference between two consecutive numbers. For example if the method is called with the array `{2,5,9,10,15}` then it should return 5 since the largest difference is $15-10=5$. (4p)

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

```
> java Question2 2 5 9 10 15
5
```

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

3. We will now develop methods in the class `DominoModel` that can be used for playing the game of domino. The game is played with 28 tiles where each tile is marked with two numbers in the range from 0 to 6. The full set of tiles is shown below:



As you can see there is one tile for every combination of numbers.



We will assume that there is already a class `DominoTile`:

```
public class DominoTile {
    public DominoTile(int left, int right) { ... }
    public int getLeft() { ... };
    public int getRight() { ... };
}
```

which represents a single tile. Here the variables `left` and `right` in the constructor represent correspondingly the left and the right number on

the tile. You can also get the values of those variables for an existing tile by using the methods `getLeft()` and `getRight()`.

Implement a class `DominoModel` with the following methods and instance variables:

- An instance variable called `set` which represents a list of `DominoTile`.
- A constructor which initializes the variable `set` and fills it in with the full set of tiles. You must fill the set by using a loop and you should make sure that every tile is added only once. Note that if you have already added for example , then you should not add . This is the same tile which is simply rotated.







- A method:

```
public DominoTile pickTile();
```

which picks up a random tile from the `set` and returns it. The method should remove the tile from the `set`. If there are no tiles left in the `set` then it should return `null`.

- A method:

```
public boolean match(DominoTile tile1, DominoTile tile2);
```

which returns true if the pair `tile1` and `tile2` share at least one number. For example the pairs   and   both match. The pair   doesn't match.

Hint: Since you have to remove elements from the list `set`, it would be easier to use the class `ArrayList` and the method `remove()`. Check the cheatsheet.

(8p)

4. Implement the method:

```
public static int[][] shift(int n)
```

which given a number `n` generates an $n \times n$ matrix of integers such that:

- the first row in the matrix is always the numbers from 0 to `n-1`.
- each of the following rows contains the same sequence but shifted cyclically with one position to the left.

For example `shift(5)` should return:

```
0 1 2 3 4
1 2 3 4 0
2 3 4 0 1
3 4 0 1 2
4 0 1 2 3
```

(10p)

5. Implement the method:

```
public static boolean login(String userName, String password)
```

which takes a user name and password and returns true if there is a registered user with the corresponding password. The list of registered users is stored in the file `accounts.txt`.

For simplicity we will assume that each user name and password is just a single word. This means that they could be read by using the method `next()` in `Scanner`. The format of the file is simply a sequence of words where the first word is the user name and the next is the password. For example the following is a valid file:

```
John 123 Mary m12 Jane 1238
Karl admf Erik w23hf
```

(10p)