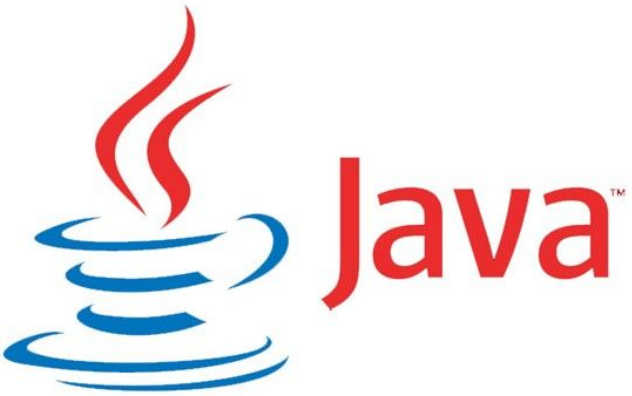
Java Project : Lorann



Summary

Reminder of the subject & Organization

Reminder of the subject

Objective and context

The final goal of this project is to partially rewrite in Java the code of an old game named Lorann that was initially coded for the Thomson MO5 and TO7.

Lorann is a PacMan like game, which means we control a character (Lorann) that moves from tile to tile, trying to gather as much items as possible without getting caught by 1 to 4 monsters through 101 different levels. Lorann's only way to defend himself is to throw spells that can kill his enemies.

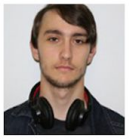
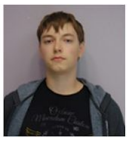
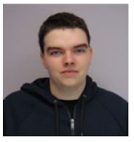
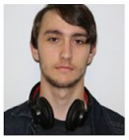
Required work

- Set up 5 levels easily accessible by modifying the program settings
- Store the 5 maps in the database our program will access
- Produce several UML diagrams
- Display a functional game

Constraints

- Using Java, Maven, Git, JUnit
- Not using a graphical Framework other than Swing
- Not writing SQL requests in the Java code
- All the calls must be done through stored procedures

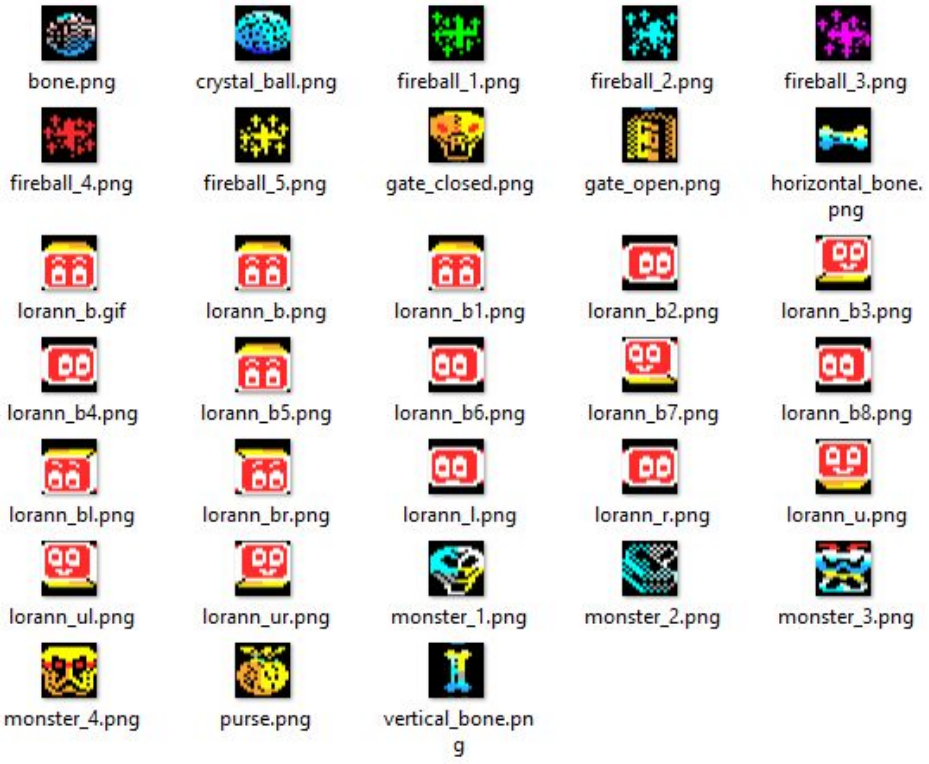
Planned organization

	Antoine Soulaire	Dorian Buquet	Clément Blin	Pierre Oudin
Code Java				
BDD				
Livrable				
UML				
Maps				

Realization

Maps

The first thing we planned to do was the translation and the realization of the 5 maps that we'll use during the game presentation.
First, we did some simple researches and found out that the maps of the actual game were 20x12 tiles.



We had a few sprites in the resources that we must use in our maps. So we established this translation document :

```
0 = vide
1 = purse
2 = lorann
3 = crystal ball
4 = fireball
5 = gate closed
6 = gate open
7 = vertical bone
8 = horizontal bone
9 = bone
A = monster 1
B = monster 2
C = monster 3
D = monster 4
```

And we wrote the map in .txt documents, here's an example (our third map) :

```
98898888888888888889 98898888888888888889
70070000030001100007 70070000030001100007
70070000000001100007 70070000000001100007
70098888888888900007 70098888888888900007
70000000000000000007 70000000000000000007
70009890989888900007 70009890989888900007
70107000007000000007 70107000007000000007
70007000007001001007 70007000007001001007
70507002007000000d07 70507002007000000d07
70007000007000010007 70007000007000010007
70007000007000000007 70007000007000000007
98889888889888888889 98889888889888888889
```

For an easier comprehension of the left map (txt document), I copied and pasted it in a microsoft word document in order to add color in order to visualize the walls an elements. The zeros represent void tiles.

The finale result is the last screenshot under the two others, once the translation from characters to sprites is done.

Once our five maps were done, we needed to put them in a database as it's one of the project's constraints. The next goal will be to implement the maps on the database and display them on the game's window with the Java code.

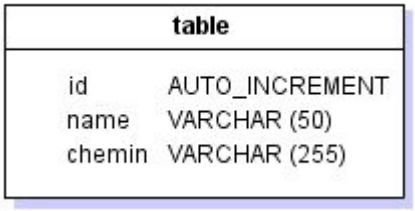
Database

The next step was to implement the “text maps” into the Database, not much to do here, we just had to change the document and merge all 12 lines into 1 like so :

+ Options

	id	name	chemin
Éditer Copier Supprimer	1	niveau 1	98888888888888888898000000000000000000088090009000...
Éditer Copier Supprimer	2	niveau 2	98888888888888888897110700000000007003771107000000...
Éditer Copier Supprimer	3	niveau 3	98898888888888888897007000003000110000770070000000...
Éditer Copier Supprimer	4	niveau 4	00998888888888888890990000000000000000799000000000...
Éditer Copier Supprimer	5	niveau 5	98888898888888898997100009000100099037970000099000...

Here’s the DCM we’ve used :



Then we get the map of the database in Java with the code “String letter = this.getModel().getExampleById(1).toString();”

```
    int o = 0;
    for (int i = 0 ; i < 12 ;i++)
    {
        System.out.println("");
        for (int y = 0; y <20; y++) {

        switch (this.getModel().getExampleById(1).toString().charAt(o)) {

    case '0':
        System.out.print("nothing");
        break;
    case '1':
        System.out.print("purse");
        break;
    case '2':
        System.out.print("lorann");
        break;
    case '3':
        System.out.print("crystall ball");
        break;
    case '4':
        System.out.print("fireball");
        break;
    case '5':
        System.out.print("gate closed");
        break;
    case '6':
        System.out.print("gate open");
        break;
    case '7':
        System.out.print("vertical bone");
        break;
    case '8':
        System.out.print("horizontal bone");
        break;
    case '9':
        System.out.print("bone");
        break;
    case 'A':
        System.out.print("monster 1");
        break;
    case 'B':
        System.out.print("monster 2");
        break;
    case 'C':
        System.out.print("monster 3");
        break;
    case 'D':
        System.out.print("monster 4");
        break;

    default:
        System.out.print("error");

      }
        o++;
    }

    }


}
```

Then we wrote the above code, the :

```
for (int i = 0 ; i < 12 ;i++)
      {
         System.out.println("");
         for (int y = 0; y <20; y++) {
```

Means our map, as we wanted, is gonna display with a 20x12 size because it's gonna skip a line each 20 characters 12 times.
Then the map is gonna display with what we wrote in the text document, with

```
      case '2':
         System.out.print("lorann");
         break;
```

In the above example, the program will display the character Lorann where we wrote "2" on the map.

Java Coding

We also had to do some programming in Java and do some JUnit tests, all the Java code is written and commented in our GitHub.

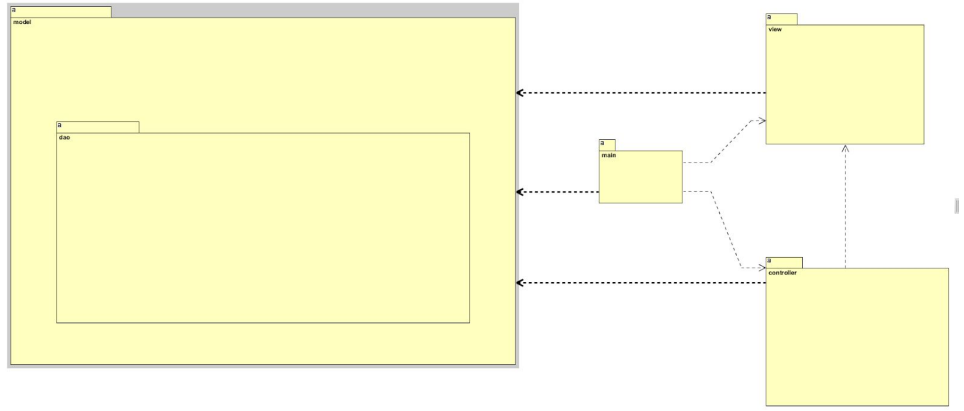
```
public void start() throws SQLException {

    //String letter = this.getView().displayMessage()

    String letter = this.getModel().getExampleById(1).toString();


    int o = 0;


    JFrame frmM = new JFrame();
    frmM.setTitle("Lorran");
    frmM.setSize(new Dimension(770,500));
    frmM.setLocationRelativeTo(null);
    frmM.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frmM.setVisible(true);
    frmM.setResizable(false);
    JPanel pnlM = new JPanel();
    pnlM.setBackground(Color.BLACK);
    frmM.setContentPane(pnlM);



    for (int i = 0 ; i < 12 ;i++)
    {
        System.out.println("");
        for (int y = 0; y <20; y++) {

            |
        switch (this.getModel().getExampleById(5).toString().charAt(o)) {

        case '0':

            JLabel space = new JLabel();
            space.setIcon(new ImageIcon("C:\\Users\\pierr\\Desktop\\sprite\\space.png"));
            pnlM.add(space);

            break;
        case '1':

            JLabel purse = new JLabel();
            purse.setIcon(new ImageIcon("C:\\Users\\pierr\\Desktop\\sprite\\purse.png"));
            pnlM.add(purse);

            break;

        case '2':

            JLabel lorann = new JLabel();
            lorann.setIcon(new ImageIcon("C:\\Users\\pierr\\Desktop\\sprite\\lorann_r.png"));
            pnlM.add(lorann);

            break;
```

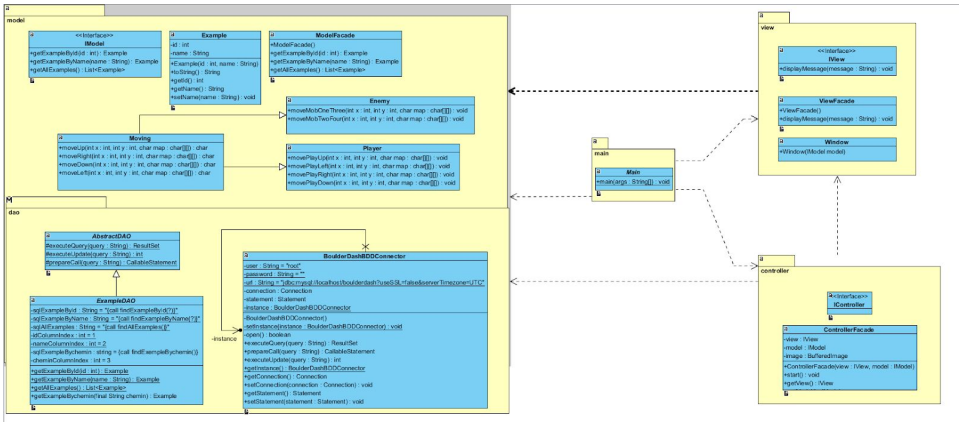
Here's an example of code that displays the sprites that form the maps from the databases.

Diagrams

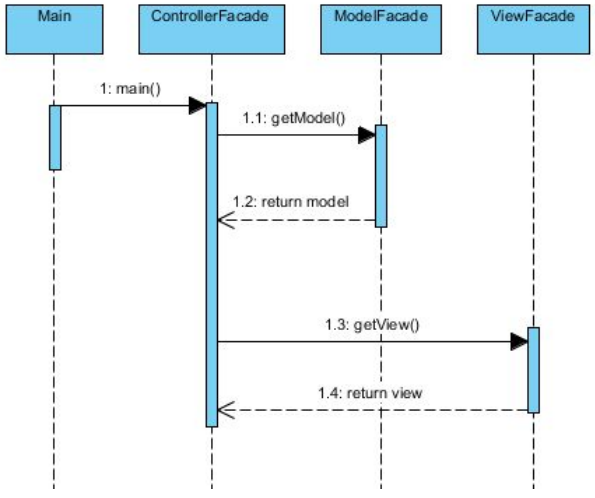
Finally, we had to do several diagrams, everything is also on the GitHub, here's an example of some diagrams we realized throughout this project :

Package Diagrams (with and without classes) :



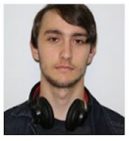
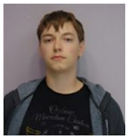
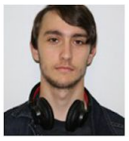
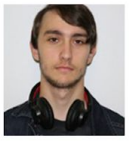
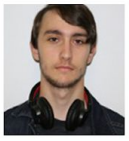
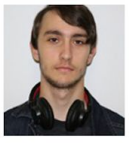


Sequence Diagram :



Conclusion

Organization performed

	Antoine Soulaire	Dorian Buquet	Clément Blin	Pierre Oudin
Code Java				
BDD				
Livrable				
UML				
Maps				

Group & Individual conclusions

Antoine : This project was interesting, and I felt like it was easy to know what every member was doing thanks to the regular sharing of our progression on GitHub and Discord. The final goal wasn't easy but we've made some good progress.

Dorian : It was an interesting project which allowed us to see the evolution of our skill in Java. While having a real objective allowing us to have a good view about the progress of the project. We had to realize the game named Lorann. Despite the different difficulties I had in this domain, I've successfully brought my part of the work to the final project.

Clément : In this group we all had difficulties, but we worked hard and do what we can. One of us do an excellent job despite his difficulties. In the beginning we take time to correctly distribute the work and see clearly what we have to do.

Pierre : The project was complicated and very little guided compared to the previous project, our group did its best and unfortunately did not manage to finish the job. Each member of the group did the research and work at home to try to finish the project but didn't succeed.

Improvements track (global)

We could've done more things with a better organization and more time but we started the project pretty slow and lost some time there. We should've implemented movement to the game and a game menu to switch between the different maps easier.