**JAVA REPORT:**

1. **SOFTWARE’S SYSTEM:**
2. **MVC:**

The MVC (model-view-controller), is a data pattern used in JAVA to build efficiently softwares. It has the property of separate the program in three categories.

1. **MODEL:**

The model part represents the software’s core, it shall treat the data and tell to the view what it should show to the user.

In our case the model has multiple functions:

* To get data from the database:
  + DBProperties class
  + DBConnection class
  + Model class
* To build the level to play, set the elements to their location and define their properties:
  + Element class and all its derivatives
  + Location class
  + Level class
  + Dimention class
  + Direction enumeration
* To select what sprite will appear on screen
  + Sprite
  + AnimatedSprite

(to see a graphic view of the model unit please report to the “model.jpg” file)

1. **VIEW:**

The view is the visual interface trough which one the user shall “see” the program.

It has to show the results of the calculations from the model unit and to “listen” every action from the user that may cause a change in the program (mouse click, key pressed etc…)

In our game the view part:

* Open a frame to see the game
  + Game Frame class
* Display graphic elements on the screen
  + GamePanel class
* Update what the user see, based on the model’s information
  + View class
* Keyes listener
  + GameFrame class

(to see a graphic view of the view unit please report to the “view.jpg” file)

1. **CONTROLLER:**

This part of the program is in charge of the event management. It has to synchronize every action in the software, in order update the view or the model correctly.

It receives events from the view unit and treat the information in order to trigger the appropriate reaction from the model part.

In our program the controller has to:

* Start the game:
  + Controller class
* Synchronize the software’s actions:
  + Clock class
* Manage the behavior of every element that will appear on screen:
  + HeroManager class
  + MoveManager class
  + CollisionManager class
  + AIManager class
* Inform the model of every order to perform and changes to apply:
  + Controller class

(to see a graphic view of the controller unit please report to the controller.jpg” file)

* **ABOUT THE CONTROLLER AND THE VIEW:**

In order to inform the controller unit about user actions, we have to set up a pattern observer.

We create an observer interface that will notice the view every possible changes and will make it react, so the information on screen update in real time.

1. **CONTRACT:**

A classic MVC pattern has a default, it creates a large number of couple between the three parts, that may cause program dysfunction and it decrease the code’s reusability.

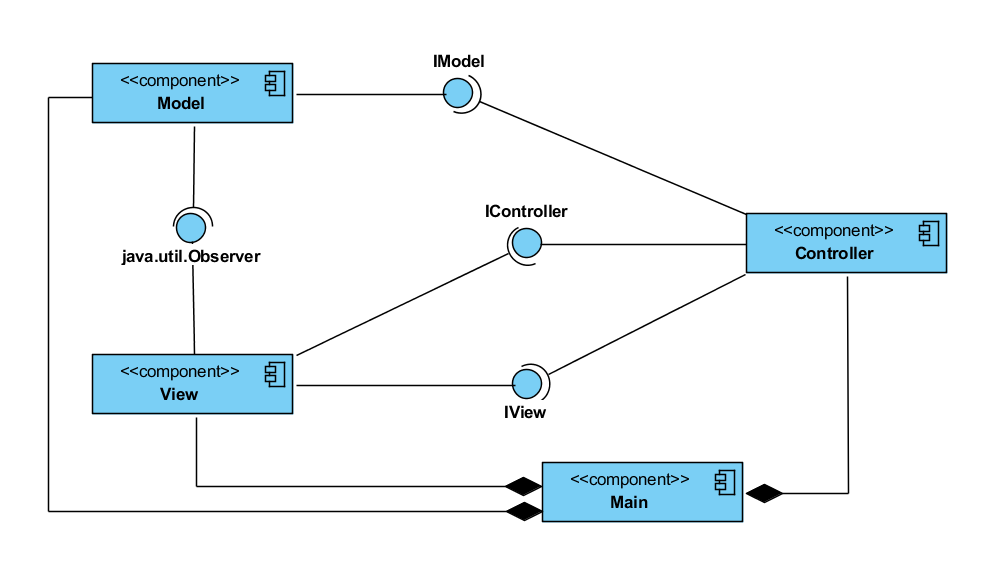
In order to face those problems, wet set a fourth unit in our program, the contract.

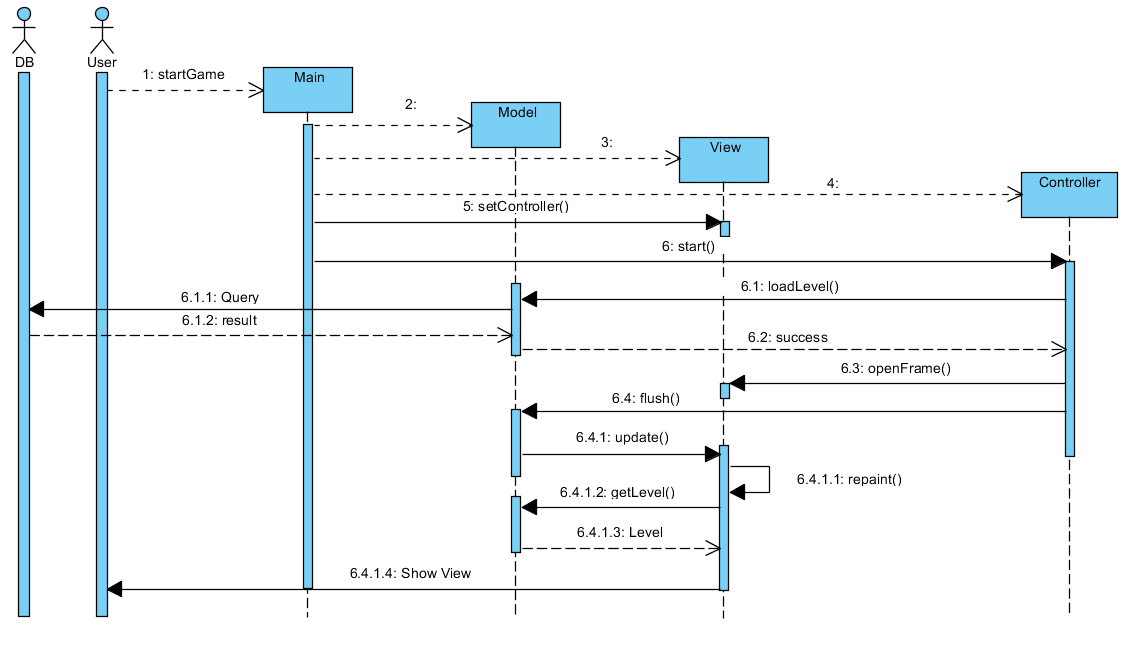
In this last part we create many interfaces within which contain operation from every class that has to be use in another part of the program than its native unit.

For example, the view needs information from the model, so it will use its interface, IModel.

(to see a graphic view of the controller unit please report to the controller.jpg” file)

Finally the program’s components are connecting this way



1. **SEQUENCE IN GAME:**

The program is obeing to the following diagram.

First the user starts the game, the main will activate the model, the view then the controller.

The controller will ask the model to load the first level, the level is charged from the data base and return to the model.

Next the controller will make the view open a frame that will the interface with the user.

At every player’s action the view will update by charging the model’s data.