Esercitazioni Ing.Sw

Gian Enrico Conti

Mar 2025

Towards Remote View

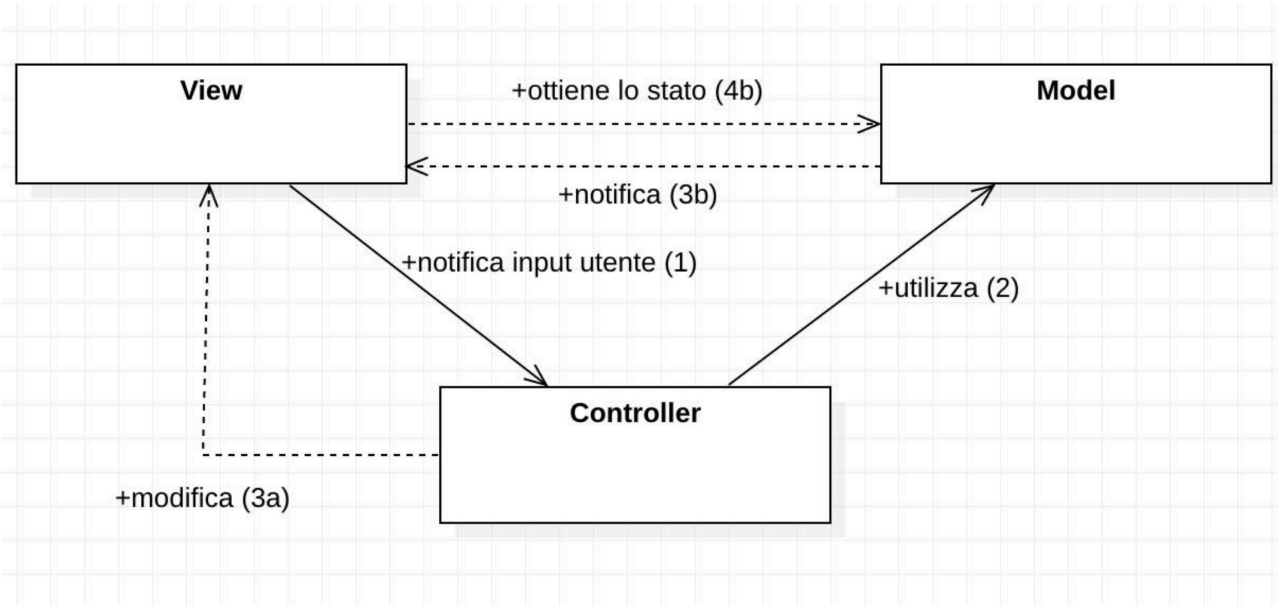
Code here:

https://github.com/ingconti/TowardsRemoteView.git

LOGIC:

- MVC con M V C in locale
- I button modificano il modello
  - View -- (evento) --> controller **modifica** modello

(invocazione di metodo)



- Aggiungiamo network...

LOGIC - Aggiungiamo network

- Modello su server!
- Client manda
- Listeners ANCORA su modello (i.e. sul server!)
- Listener propagano -> View (virtual) -> rete -> update (sul client)

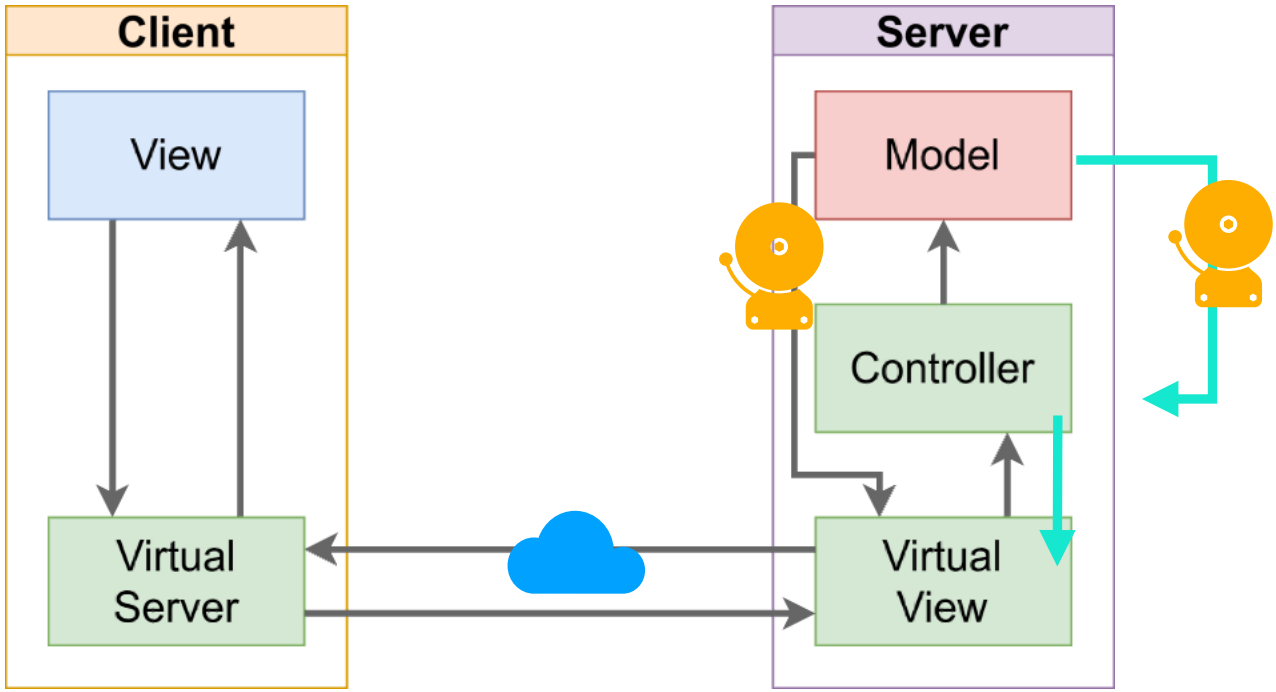
Full schema

Dal modello generano RISPOSTE e vanno lungo la rete:

Connessione "nera" o verde"?

Piu semplice "nera":

Listener su VV.



STEPS:

- app (client) in JavaFX ("TowardsRemoteViewClient")
- Modello ("*Restaurant*" *FSM*)

Testiamo in locale

..

STEPS con rete:

- Miniserver
- Testiamo "miniserver" con messaggi testuali da terminale
- Spostiamo modello su server
- Codifichiamo i cmd in messaggi
- Messaggi ---> comandi ---  -> nel modello -> Listeners

RISPOSTE

MVC LOCALE

app (client) in JavaFX: Modify Controller

1 start

add an attribute and setter:

...

```
private Stage myStage;
public void setStage(Stage stage) {
    myStage = stage;
}
```

And fix. POM..

Some fixes..

1 start


```
public void start(Stage stage) throws IOException {
    //Was: FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-
view.fxml"));
    FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));

    //Added:
    Parent root = fxmlLoader.load();
      HelloController controller = fxmlLoader.getController();
      controller.setStage(stage);

    //was: Scene scene = new Scene(fxmlLoader.load(), 320, 240);
    Scene scene = new Scene(root, 320, 240);

    stage.setTitle("Hello!");
    stage.setScene(scene);
    stage.show();
}
```

Il controller vede il suo stage, i.e. la VIEW

Modello

Abbiamo un semplice ristorante:

(specs &&& code for a simple network automaton here: https://github.com/ingconti/AutomatonFromNetwork)

Our FSM describes a lunch.
It starts from ENTREE and evolves sending "g" (GO!) command from client on TCP.

States are:

```
UNKNOWN,
ENTREE,
MAIN_COURSE,
SECOND_COURSE,
DESSERT,
THE_END_OF_LUNCH;
```

Modello

Allowed commands are:

- "G" (GO!)
- every initial char of every state, i.e. "E", "M"..."T"
- "P" to pay.

You cannot go back when specifying state.

NOTE: you cannot finish lunch without PAYING!
You can pay in any state you are in.

User cannot see why he cannot go on.

Note: for simplicity these commands are equal to states, (except "G").
to be precise we should pay more attention: cmd shoudl be:
"GoTo_E", "GoTo_M" and so on...

Try it!

To test against console:

(Dowload form git... run.. it will listen on pot 1234)

/usr/bin/nc 127.0.0.1 1234

and type in console:
server will receive and send back some info.

Live..

```
g
 new state: MAIN COURSE
g
 new state: SECOND COURSE
g
 new state: DESSERT
p
 new state: DESSERT
p
 new state: DESSERT
asd
 NOT MOVED FROM DESSERT
p
 NOT MOVED FROM DESSERT
p
 NOT MOVED FROM DESSERT
d
 NOT MOVED FROM DESSERT
m
 NOT MOVED FROM DESSERT
g
 new state: END OF YOUR LUNCH!
```

Modello: ENUM "DinnerPhase"

DinnerPhase

```java
enum DinnerPhase implements Comparable<DinnerPhase> {
    UNKNOWN,
    ENTREE,
    MAIN_COURSE,
    SECOND_COURSE,
    DESSERT,
    THE_END_OF_LUNCH;

    DinnerPhase next() {

        switch (this) {
            case ENTREE:
                return MAIN_COURSE;

            case MAIN_COURSE:
                return SECOND_COURSE;

            case SECOND_COURSE:
                return DESSERT;

            case DESSERT:
                return THE_END_OF_LUNCH;

            case THE_END_OF_LUNCH:
                return THE_END_OF_LUNCH;
        }

        return UNKNOWN;
    }

    static DinnerPhase fromString(String s) {

        switch (s.toUpperCase().charAt(0)) {
            case 'E':
                return ENTREE;
            case 'M':
                return MAIN_COURSE;
            case 'S':
                return SECOND_COURSE;
            case 'D':
                return DESSERT;
            case 'T':
                return THE_END_OF_LUNCH;
            case 'U':
                return UNKNOWN;
        }
        return UNKNOWN;
    }

    @Override
    public String toString() {
        switch (this) {
            case ENTREE:
                return "ENTREE";
            case MAIN_COURSE:
                return "MAIN COURSE";
            case SECOND_COURSE:
                return "SECOND COURSE";
            case DESSERT:
                return "DESSERT";
            case THE_END_OF_LUNCH:
                return "END OF YOUR LUNCH!";
        }
        return "UNKNOWN";
    }
}
```

App: add automaton


```java
public void start(Stage stage) throws IOException {

    FXMLLoader fxmlLoader = new
FXMLLoader(HelloApplication.class.getResource("hello-
view.fxml"));

    // Added:
    Parent root = fxmlLoader.load();
    HelloController controller = fxmlLoader.getController();
    //was: controller.setStage(stage, this.automaton);
    controller.setStage(stage, this.automaton);

    Scene scene = new Scene(root, 320, 240);

    stage.setTitle("Hello!");
    stage.setScene(scene);
    stage.show();
}
```

```java
// add model.
Automaton automaton;
public HelloApplication() {
    this.automaton = new Automaton();
}
```

Modello: Class "Automaton"


```java
public class Automaton {

    private Boolean paid = false;
    private DinnerPhase state = DinnerPhase.ENTREE;

    public DinnerPhase getState(){
        return state;
    }

    private Boolean canEvolve(){
        int currOrd = state.ordinal();
        int dessertOrd = DinnerPhase.DESSERT.ordinal();

        if (currOrd >= dessertOrd && !paid) {
            System.out.println("PAY BEFORE!!!");
            return false;
        }
        return true;
    }

    public void setPaid(){
        paid = true;
    }

    public Boolean evolve() {
     if (!canEvolve())
            return false;

        int currOrd = state.ordinal();
        int lastOrd = DinnerPhase.THE_END_OF_LUNCH.ordinal();

        if (currOrd < lastOrd) {
            state = state.next();
            return true;
        }
        return false;
    }


    public Boolean evolveTo(DinnerPhase toState){
        if (!canEvolve())
            return false;

        int toOrd = toState.ordinal();
        int currOrd = state.ordinal();

        if (toOrd>currOrd) {
            state = toState;
            return true;
        }
        return false;
    }

}
```

Send commands

Come detto, dal controller chiamata di metodo al modello, che è visto per ref:

*(Nota: il button è **già** nella view (stage..) e manda touch al controller)*

```
protected void onHelloButtonClick() {
   //was: welcomeText.setText("Welcome to JavaFX Application!");
    this.automaton.evolve();
    this.updateView();
}

void updateView(){
    String status = this.automaton.getState().toString();
    this.welcomeText.setText(status);
}
```

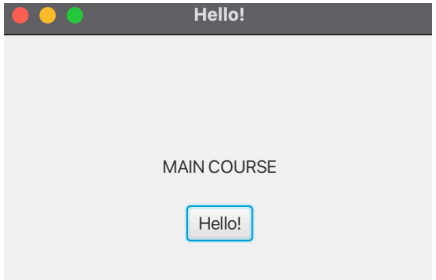
Run...

Send commands: Run

Potremmo fare un po' di refactor / renaming.. ai pulsanti...

aggiungere check su **canEvolve**,

Dare piu diagnostica..



Network!

```
/Users/ingconti/Library/Java/JavaVirtua
PAY BEFORE!!!
PAY BEFORE!!!
```

Network

Aggiungere network al client

Creare TCP server

Test scambio messaggi

Spostare model sul server

Network base code (server)

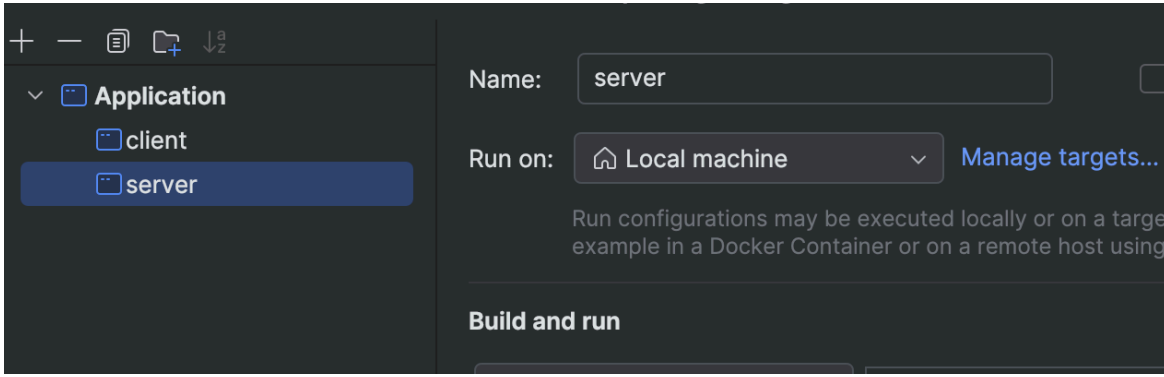
5 AddedServerAndRefactoring

(See: MinimalTCPServerAndClient2025)

- nuova classe ServerMain (copiata da MinimalTCPServerAndClient)

X ora NON threaded

- (rifattorizziamo HelloApplication in ClientMain)
- (rifattorizziamo HelloControlller in Controller)
- creiamo 2 config:



Network base code (client)

(See: MinimalTCPServerAndClient2025)

- Rimuoviamo model (andrà su server)
- Nuova classe VirtualServer
- Metodo "start()"
- Copiamo intero corpo di **main** di "ClientMain" / incolla dentro "start"

..

Network base code (client cont'd)


```java
public class VirtualServer {

    void start() {
        String hostName = "127.0.0.1";
        int portNumber = 1234;

        Socket echoSocket = null;
        try {
            echoSocket = new Socket(hostName, portNumber);
        } catch (IOException e) {
            System.err.println(e.toString() + " " + hostName);
            System.exit(1);
        }

        PrintWriter out = null;
        BufferedReader in = null;
        BufferedReader stdIn = null;
        try {
            out = new PrintWriter(echoSocket.getOutputStream(), true);
            in = new BufferedReader(
                    new InputStreamReader(echoSocket.getInputStream()));

            stdIn = new BufferedReader(
                    new InputStreamReader(System.in));

        } catch (IOException e) {
            System.err.println(e.toString() + " " + hostName);
            System.exit(1);
        }

        String userInput = "";
        while (true) {
            try {
                if (!((userInput = stdIn.readLine()) != null)) break;
                out.println(userInput);
                System.out.println("echo: " + in.readLine());
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        } // while
    }
}
```

..

Network base code (client cont'd)

- Instanziamo VirtualServer dentro client:

```
public class ClientMain extends Application {

    @Override
    public void start(Stage stage) throws IOException {
        FXMLLoader fxmlLoader = new
                FXMLLoader(ClientMain.class.getResource("hello-view.fxml"));

        Parent root = fxmlLoader.load();
        Controlller controller = fxmlLoader.getController();
        // removed controller.setStage(stage, this.automaton);
      controller.setStage(stage);
      Scene scene = new Scene(root, 320, 240);
      stage.setTitle("Hello!");
      stage.setScene(scene);
      stage.show();

      // added:
      this.virtualServer.start()
    }

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


    /*removed
     Automaton automaton;
    public ClientMain() {
        //removed this.automaton = new Automaton();
    }
     */

     VirtualServer virtualServer;
    public ClientMain() {
         this.virtualServer = new VirtualServer();
    }
}
```

Lanciamo QUI

Network base code (client cont'd)

AddedClientNetworkCodeRemovedModelFromClient

6

- Rimuoviamo istanza Model da Controller

```
public class Controlller {
    @FXML
    private Label welcomeText;

    @FXML
    protected void onHelloButtonClick() {
       /*removed:
        welcomeText.setText("Welcome to JavaFX Application!");
        this.automaton.evolve();
        */
        this.updateView();
    }

    void updateView(){
        /* removed
        String status = this.automaton.getState().toString();
        this.welcomeText.setText(status);
        */
    }

    private Stage stage;
    //was: public void setStage(Stage stage, Automaton automaton) {
        public void setStage(Stage stage) {
        stage = stage;
        //removed: this.automaton = automaton;
    }
    //removed: private Automaton automaton;
}
```

RUN...

POLITECNICO MILANO 1863

Network Client RUN

AddedClientNetworkCodeRemovedModelFromClient

6

- Crash!

```
/Users/ingconti/Library/Java/JavaVirtualMachines/openjdk-
java.net.ConnectException: Connection refused 127.0.0.1

Process finished with exit code 1
```

Prima va lanciato server...

Run again.. Good.

Network Client: sending commands..

VirtualServerPassedDownToController

Passiamo virtualServer al controller:

```
public void start(Stage stage) throws IOException {
    FXMLLoader fxmlLoader = new
            FXMLLoader(ClientMain.class.getResource("hello-view.fxml"));

    Parent root = fxmlLoader.load();
    Controlller controller = fxmlLoader.getController();
    //was: controller.setStage(stage);
     controller.setStage(stage, this.virtualServer);

    Scene scene = new Scene(root, 320, 240);
    stage.setTitle("Hello!");
    stage.setScene(scene);
    stage.show();

    // added:
    this.virtualServer.start();
}
```

Cosi dal pulsante possiamo invocare metodi su VV ...

```
protected void onHelloButtonClick() {
    String msg = new Date().toString();
    this.virtualServer.sendCmd(msg);
    this.updateView();
}
```

Network Client: send 1st command

Controller_Send_CMD1

8

All code for VirtualServer:

```java
public class VirtualServer {

    PrintWriter out = null;
    BufferedReader in = null;

    void start() {

        String hostName = "127.0.0.1";
        int portNumber = 1234;

        Socket echoSocket = null;
        try {
            echoSocket = new Socket(hostName, portNumber);
        } catch (IOException e) {
            System.err.println(e.toString() + " " + hostName);
            System.exit(1);
        }
        /* moved up, class instances:
        PrintWriter out = null;
        BufferedReader in = null;*=
        BufferedReader stdIn = null;
        try {
            out = new PrintWriter(echoSocket.getOutputStream(), true);
            in = new BufferedReader(
                    new InputStreamReader(echoSocket.getInputStream()));

            stdIn = new BufferedReader(
                    new InputStreamReader(System.in));

        } catch (IOException e) {
            System.err.println(e.toString() + " " + hostName);
            System.exit(1);
        }
        /* removed:
        String userInput = "";
        while (true) {
            try {
                if (!((userInput = stdIn.readLine()) != null)) break;
                out.println(userInput);
                System.out.println("echo: " + in.readLine());
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        } // while*/
    }

    void sendCmd(String cmd){
        out.println(cmd);
    }
}
```

All code Controlller:

```java
public class Controlller {
    @FXML
    private Label welcomeText;

    @FXML
    protected void onHelloButtonClick() {
        String msg = new Date().toString();
        this.virtualServer.sendCmd(msg);
        this.updateView();
    }

    void updateView(){

    }

    VirtualServer virtualServer;
    private Stage stage;
        public void setStage(Stage stage, VirtualServer virtualServer) {
        this.stage = stage;
        this.virtualServer = virtualServer;
    }

}
```

POLITECNICO MILANO 1863

Network Client: send 1st command: RUN

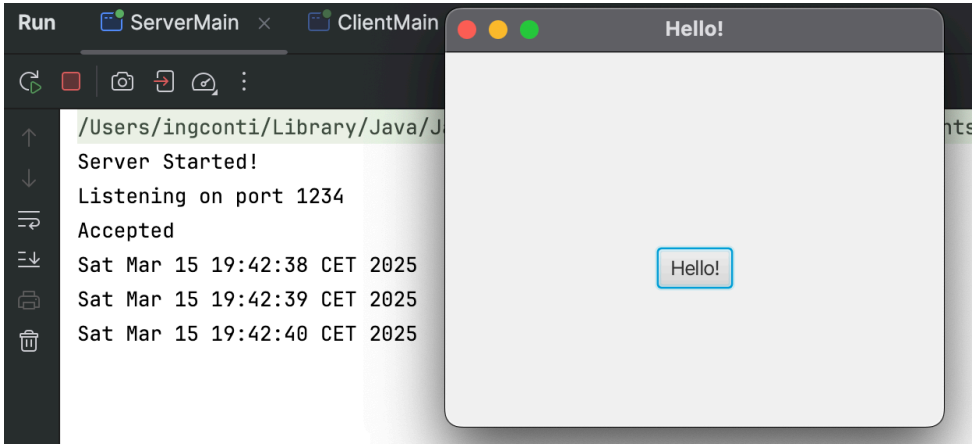
Controller_Send_CMD1

8

Run server... run client..

Click multiple times and see server..

(We do not process yet answer from server to client..)



Process answer:

POLITECNICO MILANO 1863

Network Client: manage reading

Nel client NON possiamo semplicemente riabilitare loop:

```
String userInput = "";
while (true) {
    try {
        if (!((userInput = stdIn.readLine()) != null)) break;
        out.println(userInput);
        System.out.println("echo: " + in.readLine());
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
} // while
```

Perchè bloccante *(provateci..)*

Thread..

Creiamo classe ClientThread:

```
public class ClientThread extends Thread {
    private BufferedReader reader;

    public ClientThread(BufferedReader reader) {
        this.reader = reader;
    }

    public void run() {
        System.out.println("ClientThread started");
       int i = 0;

        while (true) {
            try {
                String answer = this.reader.readLine();
                System.out.println(answer);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Passiamo reader, rifastorrizzare con Socket?

...

Network Client: manage reading: Thread (client cont'd)

09ThreadedClient

Nel VV:

...

```
  ClientThread clientThread = new ClientThread(in);
    clientThread.start();
} // end of start

void sendCmd(String cmd){
    out.println(cmd);
}
```

Run...

```
/Users/ingconti/Library/Java/
ClientThread started
SUN MAR 16 07:56:15 CET 2025
SUN MAR 16 07:56:16 CET 2025
```

Full code:

```
public class VirtualServer {

    PrintWriter out = null;
    BufferedReader in = null;

    void start() {

        String hostName = "127.0.0.1";
        int portNumber = 1234;

        Socket echoSocket = null;
        try {
            echoSocket = new Socket(hostName, portNumber);
        } catch (IOException e) {
            System.err.println(e.toString() + " " + hostName);
            System.exit(1);
        }

        BufferedReader stdIn = null;
        try {
            out = new PrintWriter(echoSocket.getOutputStream(), true);
            in = new BufferedReader(
                    new InputStreamReader(echoSocket.getInputStream()));
        } catch (Exception e) {
            System.err.println(e.toString());
            System.exit(1);
        }

        ClientThread clientThread = new ClientThread(in);
        clientThread.start();
    } // end of start

    void sendCmd(String cmd){
        out.println(cmd);
    }
}
```

Network Client: send "evolve" cmd

10-SendEvolveCmd

Nel controller:

```
@FXML
protected void onEvolveButtonClick() {
    // was: String msg = new Date().toString();
    String msg = "g"; // as per AutomatonFromNetwork.
    this.virtualServer.sendCmd(msg);
    this.updateView();
}

@FXML
protected void onPayButtonClick() {
    // was: String msg = new Date().toString();
    String msg = "p"; // as per AutomatonFromNetwork.
    this.virtualServer.sendCmd(msg);
    this.updateView();
}
```

Fxml:

```
<Button text="Evolve!"
onAction="#onEvolveButtonClick"/>
<Button text="Pay!"
onAction="#onPayButtonClick"/>
```

Run...

Network Server: process and answer

11-ProcessEvolveCmdOnServer

Nel server:

```
    String s = "";
    try {
        while ((s = in.readLine()) != null) {
            System.out.println(s);
            //was: out.println(s.toUpperCase());
            out.println(processCmd(s)); // write back to client.
        }
        System.out.println("done");
    } catch (IOException e) {
        e.printStackTrace();
    }
    // we should close..
}

    static Automaton model = new Automaton();

    // from: static Boolean readLoop(... ) on Automamton code.
    static String processCmd(String s){

        String stateString;
        Boolean goOn = false;
        s = s.toUpperCase();
        System.out.println(s);
        if (s.equals("G")){
            goOn = model.evolve();
        }else if (s.equals("P")){
            model.setPaid();
        }else{
            DinnerPhase ph = DinnerPhase.fromString(s);
            goOn = model.evolveTo(ph);
        }

        stateString = model.getState().toString();
        return stateString;
    }
```

Run...

All code:

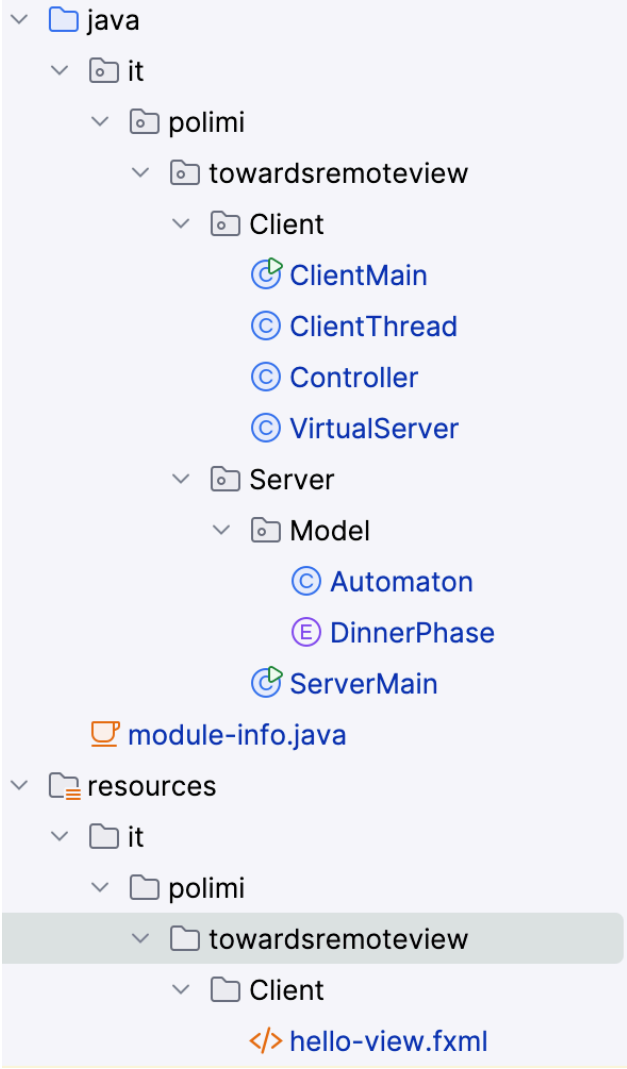
```
public class ServerMain {
    static int portNumber = 1234;

    public static void main(String[] args) {

        System.out.println("Server Started!");

        ServerSocket serverSocket = null;
        try {
            serverSocket = new ServerSocket(portNumber);
        } catch (IOException e) {
            e.printStackTrace();
        }

        System.out.println("Listening on port " + portNumber);

        Socket clientSocket = null;
        try {
            clientSocket = serverSocket.accept();
        } catch (IOException e) {
            e.printStackTrace();
        }

        System.out.println("Accepted");

        BufferedReader in = null;
        PrintWriter out = null;
        try {
            in = new BufferedReader(
                    new InputStreamReader(clientSocket.getInputStream()));
            out = new PrintWriter(clientSocket.getOutputStream(), true);

        } catch (IOException e) {
            e.printStackTrace();
        }

        String s = "";
        try {
            while ((s = in.readLine()) != null) {
                System.out.println(s);
                //was: out.println(s.toUpperCase());
                out.println(processCmd(s));

            }
            System.out.println("done");
        } catch (IOException e) {
            e.printStackTrace();
        }

        // we should close..
    }


        static Automaton model = new Automaton();

        // from: static Boolean readLoop(BufferedReader in, PrintWriter out ) on Automamton code.
        static String processCmd(String s){

            String stateString;
            Boolean goOn = false;

            s = s.toUpperCase();
            System.out.println(s);
            if (s.equals("G")){
                goOn = model.evolve();
            }else if (s.equals("P")){
                model.setPaid();
            }else{
                DinnerPhase ph = DinnerPhase.fromString(s);
                goOn = model.evolveTo(ph);
            }

            stateString = model.getState().toString();
            return stateString;
        }

    }
```

POLITECNICO MILANO 1863

Just a bit of refactor..

12-Refactored

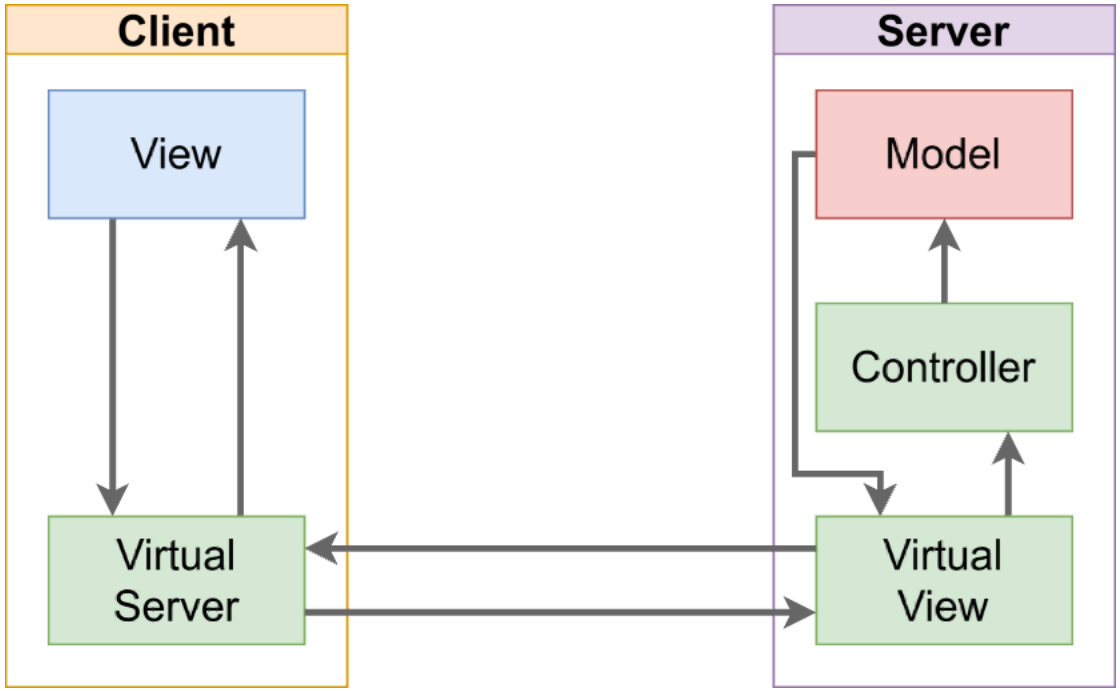
Separiamo in due package:

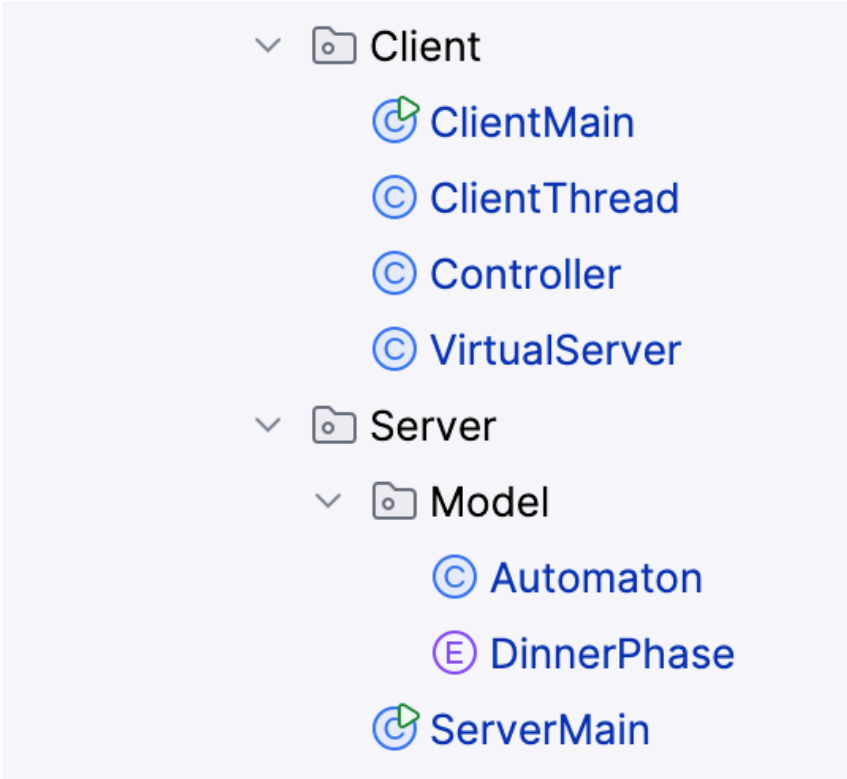


POLITECNICO MILANO 1863

Just a moment..

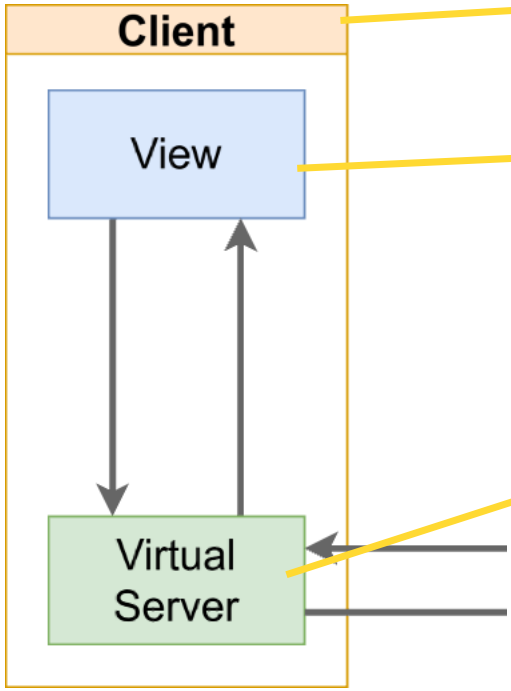
Si era detto:

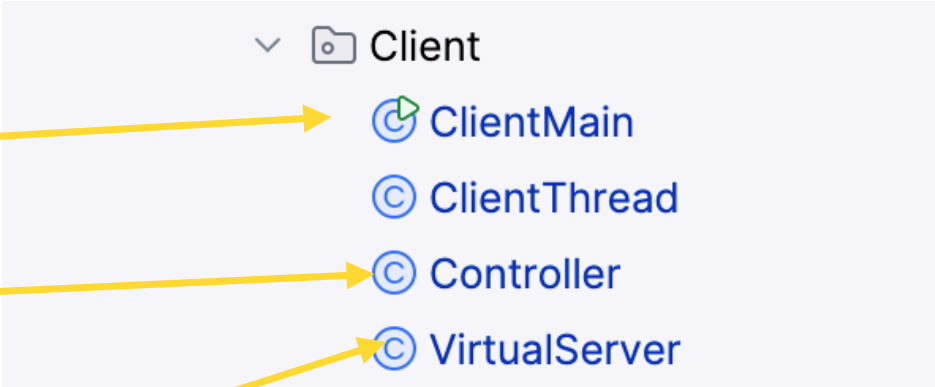




...

CLient ok...





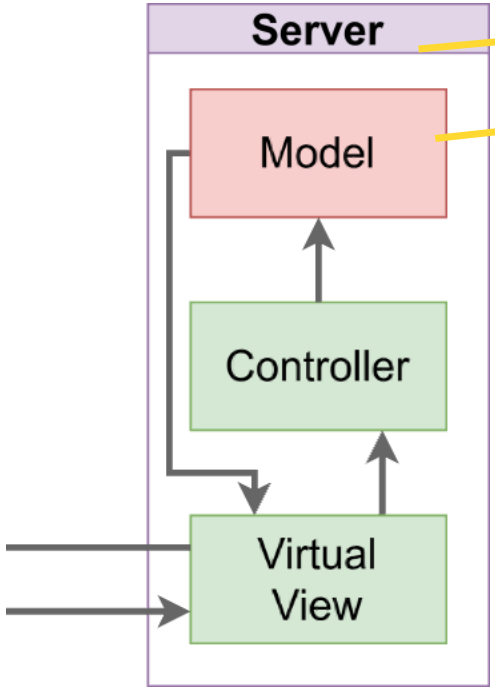
- "Controller" e' il componente JavaFX

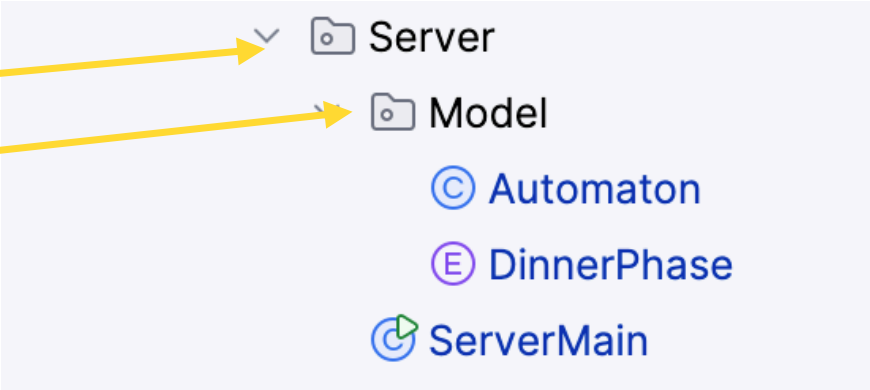
Che visualizza la UI e manda user cmds

- Il file Controller contiene implicitamente

la View

Just a moment..



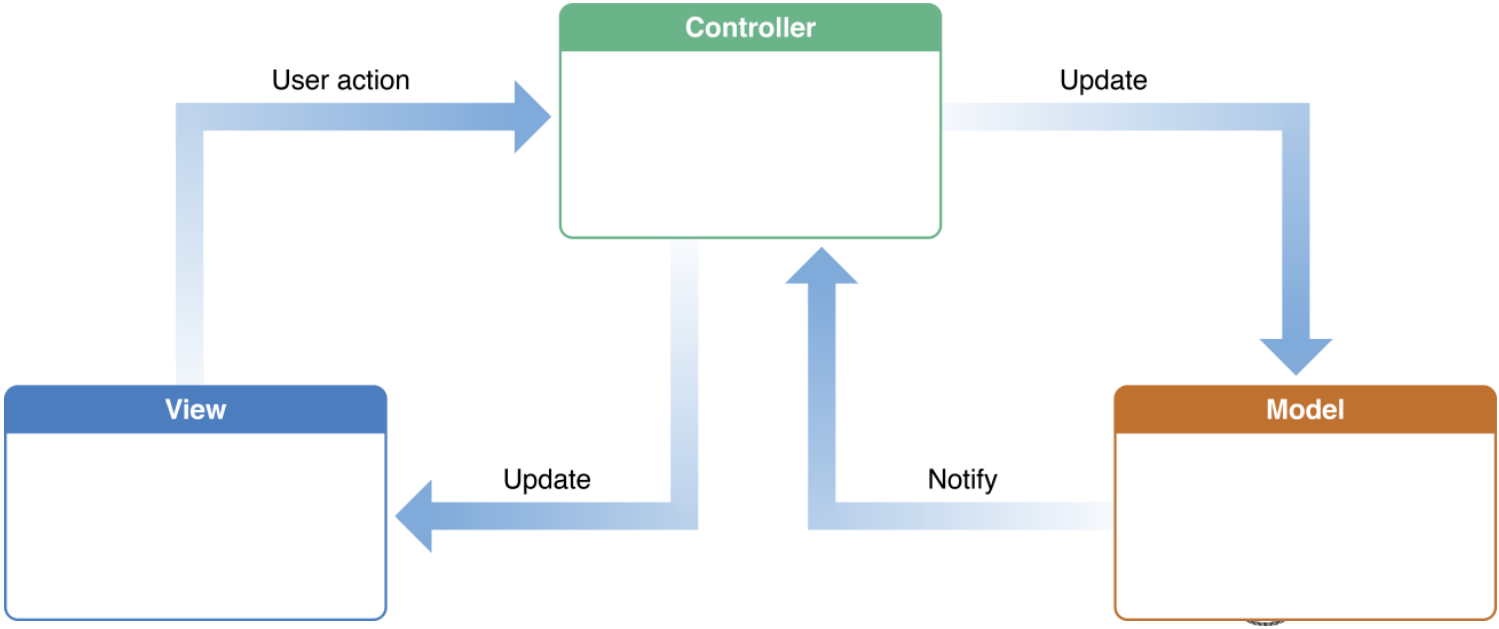


Non abbiamo separato opportunamente

Il Server...

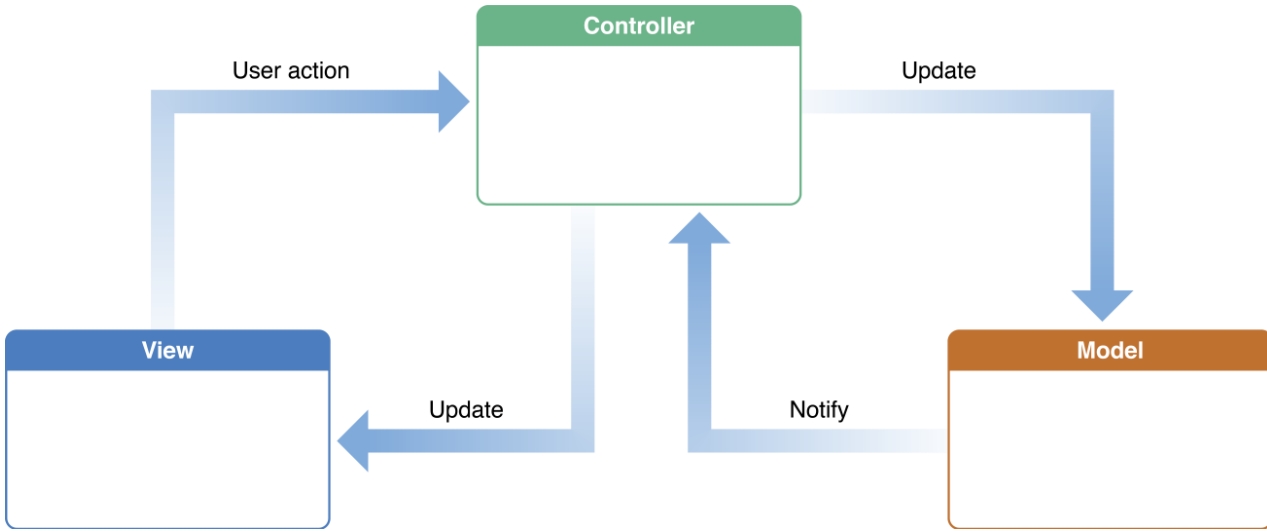
Server 1' variante

- instanzia:
- View
- Model
- Controller (a cui passa rif. a V e M)



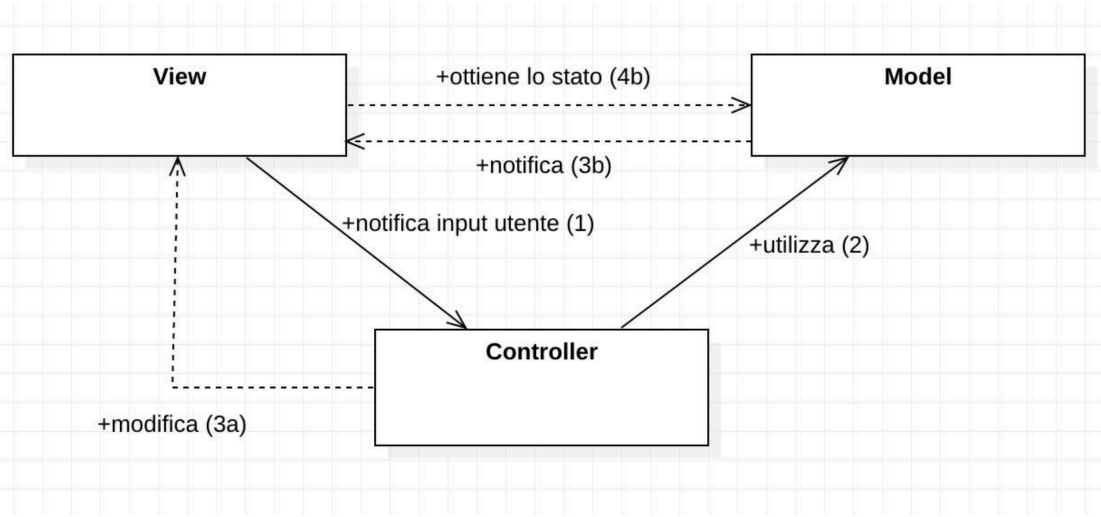
Server 1' variante: il Controller

- business logic
- Riceve input dalla View (qui dalla VV, i.e. rete, che è "virtuale")
- Modifica Modello
- É notificato dal modello (listeners) e dalla view



Server 2' variante

- instanzia:
- Model
- Controller (riceve rif. a M)
- View (riceve rif. a C)
- Listener propagano -> View (virtual) -> rete -> update (sul client) *



Nota: non avendo piu eventi UI,
facciamo chiamata diretta V->C
(andrebbe usati listener su UI, javaFX: public class
Controller implements ActionListener { ...)

* slide 4

Server 2' variante: steps

- Codice di rete nella VV
- Instanziamo VV nella App
- Nel costruttore C passiamo rif. al model (invocazione diretta)
- Nel costruttore VV passiamo rif. al controller
- **La VV non processa piu il cmd!** *(Useremo listeners)*

..

Server 2' codice VV


```java
public class ServerMain {
    static int portNumber = 1234;
    static Automaton model = new Automaton();

    public static void main(String[] args) {

        System.out.println("Server Started!");
        ServerSocket serverSocket = null;
        try {
            serverSocket = new ServerSocket(portNumber);
        } catch (IOException e) {
            e.printStackTrace();
        }

        System.out.println("Listening on port " + portNumber);
        Socket clientSocket = null;
        try {
            clientSocket = serverSocket.accept();
        } catch (IOException e) {
            e.printStackTrace();
        }

        System.out.println("Accepted");

        Controller controller = new Controller(model);
        VirtualView virtualView = new VirtualView(clientSocket, controller);
        virtualView.networkEventLoop();
        // we should close..
    }
}
```

Server 2' codice Controller


```java
public class Controller {
    Automaton model = null;

    public Controller(Automaton model) {
        this.model = model;
    }

    String processCmd(String s){ // Business logic
        String stateString;
        Boolean goOn = false;

        s = s.toUpperCase();
        System.out.println(s);
        if (s.equals("G")){
            goOn = model.evolve();
        }else if (s.equals("P")){
            model.setPaid();
        }else{
            DinnerPhase ph = DinnerPhase.fromString(s);
            goOn = model.evolveTo(ph);
        }

        stateString = model.getState().toString();
        return stateString;
    }
}
```

Server 2' codice Server

13-AddedControllerAndVVOnServer

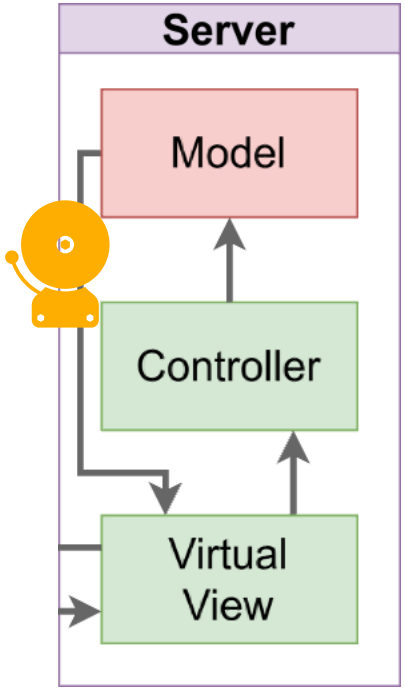
```java
public class ServerMain {
    static int portNumber = 1234;
    static Automaton model = new Automaton();

    public static void main(String[] args) {

        System.out.println("Server Started!");
        ServerSocket serverSocket = null;
        try {
            serverSocket = new ServerSocket(portNumber);
        } catch (IOException e) {
            e.printStackTrace();
        }

        System.out.println("Listening on port " + portNumber);

        Socket clientSocket = null;
        try {
            clientSocket = serverSocket.accept();
        } catch (IOException e) {
            e.printStackTrace();
        }

        System.out.println("Accepted");
        Controller controller = new Controller(model);
        VirtualView virtualView = new VirtualView(clientSocket, controller);
        virtualView.networkEventLoop();
        // we should close..
    }
}
```

Run...

Server 2' codice Server: note

- Server riceve ma NON risponde (log ok)
- Listeners! On model
- Instanziati su VV

Listener logic...



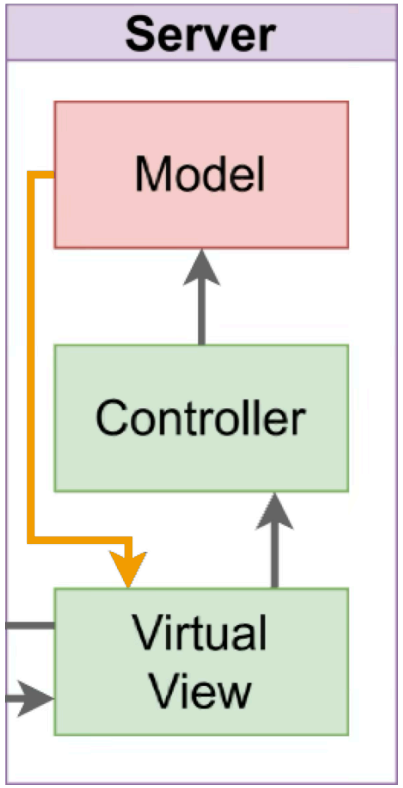
Server 2' codice Server: note

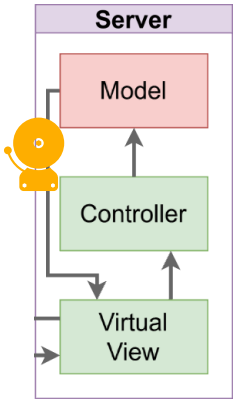
- NOTA:

Sarebbe possible anche avere:

Chiamata diretta,

ma **viola** modello ad observers/listeners





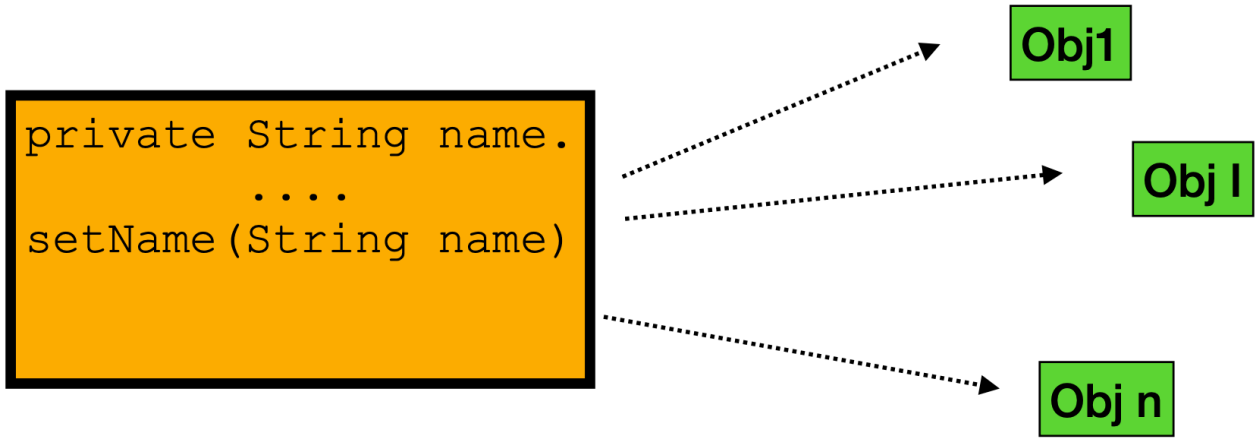
(Avremmo rif. incrociati,

M ha rif a VV, VV a M via C)

Let's listen!

Vogliamo un meccanismo che permetta a PIU oggetti di essere notificati su un cambiamento.

Perchè?
A) Per esempio vogliamo poter aggiungere **ascoltatori senza modificare il codice della sorgente** delle modifiche.
Sul "set" notifico.



B) non è necessario tenere una lista dei potenziali "interessati alle modifiche"
C) disaccoppiare le modifiche al modello dalle view che devono mostrare le modifiche.

(See also at: https://github.com/ingconti/JAVAPropertyListeners)

Let's listen!

- make attributes private *(so You are forces to pass via **Setters**)*
- Instantiate a PropertyChangeListener listener in VV
- Pass PropertyChangeListener listener to model
  (aggiungere attributo PropertyChangeListener listener; al modello)
- Set it as listener or changes in model.

Nota: by design la VV non vede il modello.. passeremo dal controller..

Model e controller

Model:

```
private PropertyChangeListener listener;

public void setListener(PropertyChangeListener listener)
{
    this.listener = listener;
}
```

Controller:

```
public void setListener(PropertyChangeListener listener)
{
    this.model.setListener(listener);
}
```

Model e controller full code

Model:

```java
public class Automaton {

    private Boolean paid = false;
    private DinnerPhase state = DinnerPhase.ENTREE;

    public DinnerPhase getState(){
        return state;
    }

    private Boolean canEvolve(){
        int currOrd = state.ordinal();
        int dessertOrd = DinnerPhase.DESSERT.ordinal();

        if (currOrd >= dessertOrd && !paid) {
            System.out.println("PAY BEFORE!!!");
            return false;
        }
        return true;
    }

    public void setPaid(){
        paid = true;
    }


    public Boolean evolve() {
        if (!canEvolve())
            return false;

        int currOrd = state.ordinal();
        int lastOrd = DinnerPhase.THE_END_OF_LUNCH.ordinal();

        if (currOrd < lastOrd) {
            state = state.next();
            return true;
        }
        return false;
    }


    public Boolean evolveTo(DinnerPhase toState){
        if (!canEvolve())
            return false;
        int toOrd = toState.ordinal();
        int currOrd = state.ordinal();

        if (toOrd>currOrd) {
            state = toState;
            return true;
        }
        return false;
    }

    private PropertyChangeListener listener;
    public void setListener(PropertyChangeListener listener) {
        this.listener = listener;
    }
}
```

Controller:

```java
public class Controller {
    Automaton model = null;

    public Controller(Automaton model) {
        this.model = model;
    }

    String processCmd(String s){ // Business logic
        String stateString;
        Boolean goOn = false;

        s = s.toUpperCase();
        System.out.println(s);
        if (s.equals("G")){
            goOn = model.evolve();
        }else if (s.equals("P")){
            model.setPaid();
        }else{
            DinnerPhase ph = DinnerPhase.fromString(s);
            goOn = model.evolveTo(ph);
        }

        stateString = model.getState().toString();
        return stateString;
    }

    public void setListener(PropertyChangeListener listener) {
        this.model.setListener(listener);
    }
}
```

Listener code 2

- *(make methods private (to model better..))*
- Instantiate a PropertyChangeListener listener in VV
- Pass PropertyChangeListener listener to model
- Set it as listener or changes in model.

Listener code


```
public class ModelListener implements PropertyChangeListener {
    @Override
    public void propertyChange(PropertyChangeEvent evt) {
        String debugStr = evt.getPropertyName()  + " from: " +
                evt.getOldValue() + " to: "  + evt.getNewValue();
        System.out.println(debugStr);
    }
}
```

VV code

14-AddingListeners

```java
public class VirtualView {

    BufferedReader in = null;
    PrintWriter out = null;
    Socket clientSocket = null;
    Controller controller = null;
    ModelListener listener;

    public VirtualView(Socket clientSocket, Controller controller) {
        this.clientSocket = clientSocket;
        this.listener = new ModelListener();
        this.controller = controller;
        controller.setListener(this.listener);

        try {
            in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
            out = new PrintWriter(clientSocket.getOutputStream(), true);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }


    void networkEventLoop(){
        String s = "";
        try {
            while ((s = in.readLine()) != null) {
                System.out.println(s);
                // no more...out.println(processCmd(s));
                String status = this.controller.processCmd(s);
                System.out.println(status); // only for debug. we do NOT send back!
            }
            System.out.println("done");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

Run...

POLITECNICO MILANO 1863

Adding Listener: RUN

14-AddingListeners

Nota: non abbiamo ancora invio indietro al client... ma sul server si vede:

```
/Users/ingconti/Library/Java/J
Server Started!
Listening on port 1234
Accepted
g
G
MAIN COURSE
g
G
SECOND COURSE
```

Ora aggiungiamo invocazione del listener sul model...

Listener - add triggering:


```
public class Automaton {

   ......
        if (currOrd < lastOrd) {
            DinnerPhase next = state.next();
            tellToListener(state, next);
            // and update:
            state = next;
            return true;
        }
        return false;
    }

    public Boolean evolveTo(DinnerPhase toState){
           ....
        if (toOrd>currOrd) {
            tellToListener(state, toState);
            // and update:
            state = toState;
            return true;
        }
        return false;
    }

    private void tellToListener(DinnerPhase from, DinnerPhase to ){
        PropertyChangeEvent evt = new PropertyChangeEvent( this, "PHASE_CHANGED", from, to);
            listener.propertyChange(evt);
    }

    private PropertyChangeListener listener;
    public void setListener(PropertyChangeListener listener) {
        this.listener = listener;
    }
}
```

Listener - add triggering: full code

15-TriggeringListeners

```java
public class Automaton {

    private Boolean paid = false;
    private DinnerPhase state = DinnerPhase.ENTREE;

    public DinnerPhase getState(){
        return state;
    }

    private Boolean canEvolve(){
        int currOrd = state.ordinal();
        int dessertOrd = DinnerPhase.DESSERT.ordinal();

        if (currOrd >= dessertOrd && !paid) {
            System.out.println("PAY BEFORE!!!");
            return false;
        }
        return true;
    }

    public void setPaid(){
        paid = true;
    }


    public Boolean evolve() {
        if (!canEvolve())
            return false;

        int currOrd = state.ordinal();
        int lastOrd = DinnerPhase.THE_END_OF_LUNCH.ordinal();

        if (currOrd < lastOrd) {
            DinnerPhase next = state.next();
            tellToListener(state, next);
            // and update:
            state = next;
            return true;
        }
        return false;
    }


    public Boolean evolveTo(DinnerPhase toState){
        if (!canEvolve())
            return false;

        int toOrd = toState.ordinal();
        int currOrd = state.ordinal();

        if (toOrd>currOrd) {
            tellToListener(state, toState);
            // and update:
            state = toState;
            return true;
        }
        return false;
    }


    private void tellToListener(DinnerPhase from, DinnerPhase to ){
        PropertyChangeEvent evt = new PropertyChangeEvent(this,"PHASE_CHANGED", from, to);
            listener.propertyChange(evt);
    }


    private PropertyChangeListener listener;

    public void setListener(PropertyChangeListener listener) {
        this.listener = listener;
    }
}
```

Run...

POLITECNICO MILANO 1863

Adding Listener: RUN && listeners

15-TriggeringListeners

Nota: non abbiamo ancora invio indietro al client...

sul server si vede:

```
Server Started!
Listening on port 1234
Accepted
g
G
PHASE_CHANGEDENTREEMAIN COURSE
```

Ora a bit of functional to send back...

Adding Listener: RUN && listeners

15-TriggeringListeners

Nota: non abbiamo ancora invio indietro al client...

sul server si vede:

```
Server Started!
Listening on port 1234
Accepted
g
G
PHASE_CHANGEDENTREEMAIN COURSE
```

Ora a bit of functional to send back...

Sending Back.. functional

Interfaccia:

```
public interface CallBack {
    void gotEvent(PropertyChangeEvent evt);
}
```

Sul Listener:

```
public class ModelListener implements PropertyChangeListener {
    @Override
    public void propertyChange(PropertyChangeEvent evt) {
        /*
        String debugStr = evt.getPropertyName()  + " from: " +
                evt.getOldValue() + " to: "  + evt.getNewValue();
        System.out.println(debugStr);*/
        this.callBack.gotEvent(evt);
    }

    private CallBack callBack;
    public ModelListener(CallBack callBack) {
        this.callBack = callBack;
    }
}
```

Sending Back.. functional

Interfaccia:

```
public interface CallBack {
    void gotEvent(PropertyChangeEvent evt);
}
```

Sul Listener:

```
public class ModelListener implements PropertyChangeListener {
    @Override
    public void propertyChange(PropertyChangeEvent evt) {
        /*
        String debugStr = evt.getPropertyName()  + " from: " +
                evt.getOldValue() + " to: "  + evt.getNewValue();
        System.out.println(debugStr);*/
        this.callBack.gotEvent(evt);
    }

    private CallBack callBack;
    public ModelListener(CallBack callBack) {
        this.callBack = callBack;
    }
}
```

Sending Back.. VV

Added:

```
public VirtualView(Socket clientSocket, Controller controller) {
    this.clientSocket = clientSocket;

    CallBack callBack = (PropertyChangeEvent evt) -> {
        String debugStr = evt.getPropertyName()  + " from: " +
                evt.getOldValue() + " to: "  + evt.getNewValue();
        System.out.println(debugStr);
    };

    this.listener = new ModelListener(callBack);
...
```

Sending Back.. VV full code.

16-sendingBack

```java
public class VirtualView {

    BufferedReader in = null;
    PrintWriter out = null;
    Socket clientSocket = null;
    Controller controller = null;
    ModelListener listener;

    public VirtualView(Socket clientSocket, Controller controller) {
        this.clientSocket = clientSocket;

        CallBack callBack = (PropertyChangeEvent evt) -> {
            String debugStr = evt.getPropertyName()  + " from: " + evt.getOldValue() + " to: "  + evt.getNewValue();
            System.out.println(debugStr);
        };

        this.listener = new ModelListener(callBack);
        this.controller = controller;
        controller.setListener(this.listener);

        try {
            in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
            out = new PrintWriter(clientSocket.getOutputStream(), true);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    void networkEventLoop(){
        String s = "";
        try {
            while ((s = in.readLine()) != null) {
                System.out.println(s);
                // no more...out.println(processCmd(s));
                String status = this.controller.processCmd(s);
                System.out.println(status); // only for debug. we do NOT send back!
            }
            System.out.println("done");
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

POLITECNICO MILANO 1863

Sending Back.. network

17-sendingBack-nertwork

```
CallBack callBack = (PropertyChangeEvent evt) -> {
    /*
    String debugStr = evt.getPropertyName()  + " from: " +
            evt.getOldValue() + " to: "  + evt.getNewValue();
    System.out.println(debugStr);
    */
    DinnerPhase state = this.controller.model.getState();
    String answer = state.toString();
    out.println(answer);
};
```

In client:

```
/Users/ingconti/Library/Java/JavaVirt
Connected to the target VM, address:
ClientThread started
ENTREE
```

Now is time to update JavaFx UI

POLITECNICO MILANO 1863

Update JavaFx UI

Dovremmo usare listeners sul modello, oppure (per semplicità..) avere rif. alla V (lo stage / al ctl. ..) dentro il thread.. (un po' sporco..)

Per **semplicità** / modularità:

Altra callBack:

```
public interface UpdateUICallBack {
    void process(String cmd);
}
```

Nel controller un po' di modifiche:

Update JavaFx UI: controller


```java
public void setStage(Stage stage, VirtualServer virtualServer) {
    this.stage = stage;
    this.virtualServer = virtualServer;

    // add call back:
    this.updateUICallBack = new UpdateUICallBack() {
        @Override
        public void process(String cmd) {
            System.out.println("Call back " + cmd);
        }
    };

    virtualServer.setUICallBack(this.updateUICallBack);
}
```

ALL CODE:

```java
public class Controller {
    @FXML
    private Label welcomeText;

    @FXML
    protected void onEvolveButtonClick() {
        String msg = "g"; // as per AutomatonFromNetwork.
        this.virtualServer.sendCmd(msg);
    }

    @FXML
    protected void onPayButtonClick() {
        String msg = "p"; // as per AutomatonFromNetwork.
        this.virtualServer.sendCmd(msg);
    }

/*
    void updateView(String msg){
        this.welcomeText.setText(msg);
    }*/


    VirtualServer virtualServer;
    private Stage stage;

    UpdateUICallBack updateUICallBack;

    public void setStage(Stage stage, VirtualServer
virtualServer) {
        this.stage = stage;
        this.virtualServer = virtualServer;

        // add call back:
        this.updateUICallBack = new UpdateUICallBack() {
            @Override
            public void process(String cmd) {
                System.out.println("Call back " + cmd);
            }
        };

        virtualServer.setUICallBack(this.updateUICallBack);
    }

}
```

POLITECNICO MILANO 1863

Update JavaFx UI: ClientThread


```java
// add call back:
private UpdateUICallBack updateUICallBack;

void setUICallBack(
    UpdateUICallBack updateUICallBack){
    this.updateUICallBack = updateUICallBack;
}
```

ALL CODE:

```java
public class ClientThread extends Thread {
    private BufferedReader reader;


    public ClientThread(BufferedReader reader) {
        this.reader = reader;
    }

    public void run() {
        System.out.println("ClientThread started");

        while (true) {
            try {
                String answer = this.reader.readLine();
                System.out.println(answer);
                this.updateUICallBack.process(answer);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }


    // add call back:
    private UpdateUICallBack updateUICallBack;

    void setUICallBack(UpdateUICallBack updateUICallBack){
        this.updateUICallBack = updateUICallBack;
    }
}
```

Update JavaFx UI: VirtualServer


```
  //was: ClientThread clientThread = new ClientThread(in);
    this.clientThread = new ClientThread(in);
    //added:
    clientThread.setUICallBack(this.updateUICallBack);
    clientThread.start();
} // end of start

public void sendCmd(String cmd){
    out.println(cmd);
}


ClientThread clientThread;
private UpdateUICallBack updateUICallBack;

void setUICallBack(UpdateUICallBack updateUICallBack){
    this.updateUICallBack = updateUICallBack;
}
```

ALL CODE:

```
public class VirtualServer {

    PrintWriter out = null;
    BufferedReader in = null;

    public void start() {

        String hostName = "127.0.0.1";
        int portNumber = 1234;

        Socket echoSocket = null;
        try {
            echoSocket = new Socket(hostName, portNumber);
        } catch (IOException e) {
            System.err.println(e.toString() + " " + hostName);
            System.exit(1);
        }

        BufferedReader stdIn = null;
        try {
            out = new PrintWriter(echoSocket.getOutputStream(), true);
            in = new BufferedReader(
                    new InputStreamReader(echoSocket.getInputStream()));
        } catch (Exception e) {
            System.err.println(e.toString());
            System.exit(1);
        }

        //was: ClientThread clientThread = new ClientThread(in);
        this.clientThread = new ClientThread(in);
        //added:
        clientThread.setUICallBack(this.updateUICallBack);
        clientThread.start();
    } // end of start

    public void sendCmd(String cmd){
        out.println(cmd);
    }


    ClientThread clientThread;
    private UpdateUICallBack updateUICallBack;

    void setUICallBack(UpdateUICallBack updateUICallBack){
        this.updateUICallBack = updateUICallBack;
    }
}
```

POLITECNICO MILANO 1863

Update JavaFx UI: run!

18-updateStage

```
/Users/ingconti/Library/Java/JavaVirtualMachines/openj
ClientThread started
ENTREE
Call back ENTREE
MAIN COURSE
Call back MAIN COURSE
SECOND COURSE
Call back SECOND COURSE
```

(Too many debug infos....)

Update JavaFx UI: GUI

Modifichiamo callBack:

```
// add call back:
this.updateUICallBack = new UpdateUICallBack() {
    @Override
    public void process(String answer) {
        System.out.println("Call back " + answer);
        updateView(answer);
    }
};
```

Aggiungiamo:

```
void updateView(String msg){
    this.welcomeText.setText(msg);
}
```

RUN...

ALL CODE:

```
public class Controller {
    @FXML
    private Label welcomeText;

    @FXML
    protected void onEvolveButtonClick() {
        String msg = "g"; // as per AutomatonFromNetwork.
        this.virtualServer.sendCmd(msg);
    }

    @FXML
    protected void onPayButtonClick() {
        String msg = "p"; // as per AutomatonFromNetwork.
        this.virtualServer.sendCmd(msg);
    }

    void updateView(String msg){
        this.welcomeText.setText(msg);
    }


    VirtualServer virtualServer;
    private Stage stage;

    UpdateUICallBack updateUICallBack;

    public void setStage(Stage stage, VirtualServer virtualServer) {
        this.stage = stage;
        this.virtualServer = virtualServer;

        // add call back:
        this.updateUICallBack = new UpdateUICallBack() {
            @Override
            public void process(String answer) {
                System.out.println("Call back " + answer);
                updateView(answer);
            }
        };

        virtualServer.setUICallBack(this.updateUICallBack);
    }

}
```

Update JavaFx UI: GUI **CRASH**


```
FIX: RunLater!
```

Update JavaFx UI: GUI

19-updateStage-GUI-fixed_crash

FIX:

```java
private void safeUpdateView(String msg){
    Platform.runLater(new Runnable() {
        @Override
        public void run() {
            updateView(msg);
        }
    });

}

private void updateView(String msg){
    this.welcomeText.setText(msg);
}


...
public void process(String answer) {
    System.out.println("Call back " + answer);
    safeUpdateView(answer);
}
```

RUN.. ok

(Non gestiamo x per ora errore da automa)

ALL CODE:

```java
public class Controller {
    @FXML
    private Label welcomeText;

    @FXML
    protected void onEvolveButtonClick() {
        String msg = "g"; // as per AutomatonFromNetwork.
        this.virtualServer.sendCmd(msg);
    }

    @FXML
    protected void onPayButtonClick() {
        String msg = "p"; // as per AutomatonFromNetwork.
        this.virtualServer.sendCmd(msg);
    }


    private void safeUpdateView(String msg){
        Platform.runLater(new Runnable() {
            @Override
            public void run() {
                updateView(msg);
            }
        });

    }

    private void updateView(String msg){
        this.welcomeText.setText(msg);
    }


    VirtualServer virtualServer;
    private Stage stage;

    UpdateUICallBack updateUICallBack;

    public void setStage(Stage stage, VirtualServer virtualServer) {
        this.stage = stage;
        this.virtualServer = virtualServer;

        // add call back:
        this.updateUICallBack = new UpdateUICallBack() {
            @Override
            public void process(String answer) {
                System.out.println("Call back " + answer);
                safeUpdateView(answer);
            }
        };

        virtualServer.setUICallBack(this.updateUICallBack);
    }

}
```

POLITECNICO MILANO 1863

Update JavaFx UI : Drawing in graphics

FXML, aggiungiamo una hBox:

```
<HBox fx:id="hbox" >

</HBox>
```

Controller, aggiungiamo una hBox:

```
@FXML
private Label welcomeText;

@FXML
private HBox hbox;
```

Update JavaFx UI : Drawing in graphics (2)

Controller:

```
private void updateView(String msg){
    this.welcomeText.setText(msg);
    this.addCircle(msg);
}
```

+ altri metodi..

```
private void addCircle(String answer){
    double centerX = xForAnswer(answer);
    double centerY = 100;
    double radius = 30;

    Circle c1 = new Circle(centerX, centerY, radius, Color.RED);
    this.hbox.getChildren().add(c1);
}
```

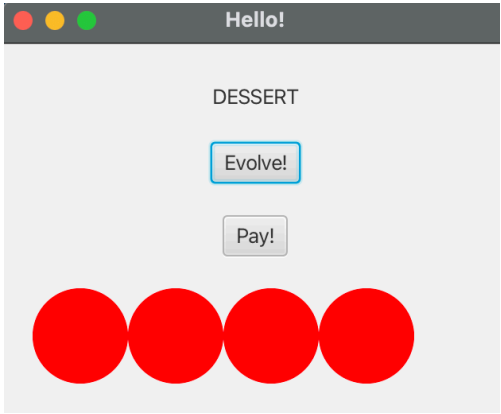
Faremo apparire dei cerchi, uno x fase....

Update JavaFx UI : Drawing in graphics full code

120-updateStage-GUI-Circles

```java
public class Controller {
    @FXML
    private Label welcomeText;
    @FXML
    private HBox hbox;

    @FXML
    protected void onEvolveButtonClick() {
        String msg = "g"; // as per AutomatonFromNetwork.
        this.virtualServer.sendCmd(msg);
    }

    @FXML
    protected void onPayButtonClick() {
        String msg = "p"; // as per AutomatonFromNetwork.
        this.virtualServer.sendCmd(msg);
    }

   private int xForAnswer(String answer) {     // todo: You will process answer..
        switch (answer) {
            case "ENTREE":return 30;
            case "MAIN COURSE":return 60;
            case "SECOND COURSE":return 90;
            case "DESSERT":return 120;
            case "END OF YOUR LUNCH!":return 30;
        }
        return 0;
    }

    private void addCircle(String answer){
        double centerX = xForAnswer(answer), centerY = 100, radius = 30;
        Circle c1 = new Circle(centerX, centerY, radius, Color.RED);
        this.hbox.getChildren().add(c1);
    }

    private void safeUpdateView(String msg){
        Platform.runLater(new Runnable() {
            @Override
            public void run() {
                updateView(msg);
            }
        });

    }

    private void updateView(String msg){
        this.welcomeText.setText(msg);
        this.addCircle(msg);
    }


    VirtualServer virtualServer;
    private Stage stage;
    UpdateUICallBack updateUICallBack;

    public void setStage(Stage stage, VirtualServer virtualServer) {
        this.stage = stage;
        this.virtualServer = virtualServer;

        // add call back:
        this.updateUICallBack = new UpdateUICallBack() {
            @Override
            public void process(String answer) {
                System.out.println("Call back " + answer);
                safeUpdateView(answer);
            }
        };

        virtualServer.setUICallBack(this.updateUICallBack);
    }
}
```

Run...



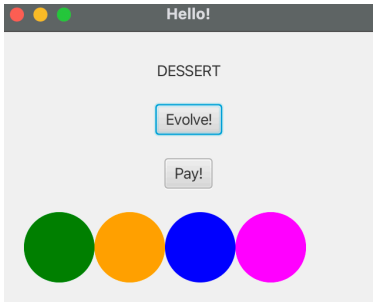
POLITECNICO MILANO 1863

Update JavaFx UI : Drawing.. colors..

21-updateStage-GUI-ColoredCircles

```java
public class Controller {
    @FXML
    private Label welcomeText;
    @FXML
    private HBox hbox;

    @FXML
    protected void onEvolveButtonClick() {
        String msg = "g"; // as per AutomatonFromNetwork.
        this.virtualServer.sendCmd(msg);
    }

    @FXML
    protected void onPayButtonClick() {
        String msg = "p"; // as per AutomatonFromNetwork.
        this.virtualServer.sendCmd(msg);
    }

    // todo: You will process answer..
    private ColorsAndCoord ColorsAndCoordForAnswer(String answer) {
        switch (answer) {
            case "ENTREE":return new ColorsAndCoord(30, Color.GREEN);
            case "MAIN COURSE":return new ColorsAndCoord(60, Color.ORANGE);
            case "SECOND COURSE":return new ColorsAndCoord(90, Color.BLUE);
            case "DESSERT":return new ColorsAndCoord(120, Color.MAGENTA);
            case "END OF YOUR LUNCH!":return new ColorsAndCoord(150, Color.YELLOW);
        }
        return new ColorsAndCoord(30, Color.BLACK);
    }

    private void addCircle(String answer){
        ColorsAndCoord colorsAndCoord = ColorsAndCoordForAnswer(answer);
        double centerX = colorsAndCoord.x, centerY = 100, radius = 30;
        Circle c1 = new Circle(centerX, centerY, radius, colorsAndCoord.c);
        this.hbox.getChildren().add(c1);
    }

    private void safeUpdateView(String msg){
        Platform.runLater(new Runnable() {
            @Override
            public void run() {
                updateView(msg);
            }
        });

    }

    private void updateView(String msg){
        this.welcomeText.setText(msg);
        this.addCircle(msg);
    }

    VirtualServer virtualServer;
    private Stage stage;
    UpdateUICallBack updateUICallBack;

    public void setStage(Stage stage, VirtualServer virtualServer) {
        this.stage = stage;
        this.virtualServer = virtualServer;

        // add call back:
        this.updateUICallBack = new UpdateUICallBack() {
            @Override
            public void process(String answer) {
                System.out.println("Call back " + answer);
                safeUpdateView(answer);
            }
        };
        virtualServer.setUICallBack(this.updateUICallBack);
    }
}
```

Run...



POLITECNICO MILANO 1863

Update JavaFx UI: all by code...

21-updateStage-GUI-ColoredCircles

Potremmo anche non usare del tutto FXML e usare solo codice x disegnare:

Al messaggio "DESSERT" svuotiamo tutto *(Dessert appare solo dopo aver pagato.. 3 Evolve->Pay->Evolve)*

```
private void cleanUpAll() {
    Group root = new Group();
    Canvas canvas = new Canvas(300, 250);
    GraphicsContext gc = canvas.getGraphicsContext2D();
    drawShapes(gc);
    root.getChildren().add(canvas);
    stage.setScene(new Scene(root));
    stage.show();
}

private void safeCleanUpAll() {
    Platform.runLater(new Runnable() {
        @Override
        public void run() {
            cleanUpAll();
        }
    });
}

private void drawShapes(GraphicsContext gc) {
    gc.setFill(Color.GREEN);
    gc.setStroke(Color.BLUE);
    gc.setLineWidth(5);
    gc.strokeLine(40, 10, 10, 40);
    gc.fillOval(10, 60, 30, 30);
    gc.strokeOval(60, 60, 30, 30);
    gc.fillRoundRect(110, 60, 30, 30, 10, 10);
    gc.strokeRoundRect(160, 60, 30, 30, 10, 10);
    gc.fillArc(10, 110, 30, 30, 45, 240, ArcType.OPEN);
    gc.fillArc(60, 110, 30, 30, 45, 240, ArcType.CHORD);
    gc.fillArc(110, 110, 30, 30, 45, 240, ArcType.ROUND);
    gc.strokeArc(10, 160, 30, 30, 45, 240, ArcType.OPEN);
    gc.strokeArc(60, 160, 30, 30, 45, 240, ArcType.CHORD);
    gc.strokeArc(110, 160, 30, 30, 45, 240, ArcType.ROUND);
    gc.fillPolygon(new double[]{10, 40, 10, 40}, new double[]{210, 210, 240, 240}, 4);
    gc.strokePolygon(new double[]{60, 90, 60, 90},new double[]{210, 210, 240, 240}, 4);
    gc.strokePolyline(new double[]{110, 140, 110, 140},new double[]{210, 210, 240, 240}, 4);
}
```

```
// add call back:
this.updateUICallBack = new UpdateUICallBack() {
    @Override
    public void process(String answer) {
        System.out.println("Call back " + answer);
        safeUpdateView(answer);
        if (answer.equals("DESSERT")){
            safeCleanUpAll();
        }
    }
};
```

POLITECNICO MILANO 1863

Update JavaFx UI: all by code. All Code

21-updateStage-GUI-ColoredCircles

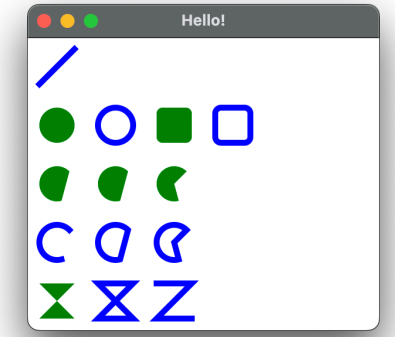
```java
public class Controller {
    @FXML
    private Label welcomeText;
    @FXML
    private HBox hbox;

    @FXML
    protected void onEvolveButtonClick() {
        String msg = "q"; // as per AutomatonFromNetwork.
        this.virtualServer.sendCmd(msg);
    }

    @FXML
    protected void onPayButtonClick() {
        String msg = "p"; // as per AutomatonFromNetwork.
        this.virtualServer.sendCmd(msg);
    }

   private ColorsAndCoord ColorsAndCoordForAnswer(String answer) {
        switch (answer) {
            case "ENTREE":return new ColorsAndCoord(30, Color.GREEN);
            case "MAIN COURSE":return new ColorsAndCoord(60, Color.ORANGE);
            case "SECOND COURSE":return new ColorsAndCoord(90, Color.BLUE);
            case "DESSERT":return new ColorsAndCoord(120, Color.MAGENTA);
            case "END OF YOUR LUNCH!":return new ColorsAndCoord(150, Color.YELLOW);
        }
        return new ColorsAndCoord(30, Color.BLACK);
    }

    private void addCircle(String answer){
        ColorsAndCoord colorsAndCoord = ColorsAndCoordForAnswer(answer);
        double centerX = colorsAndCoord.x, centerY = 100, radius = 30;
        Circle c1 = new Circle(centerX, centerY, radius, colorsAndCoord.c);
        this.hbox.getChildren().add(c1);
    }

    private void safeUpdateView(String msg){
        Platform.runLater(new Runnable() {
            @Override
            public void run() {
                updateView(msg);
            }
        });
    }

    private void updateView(String msg){
        this.welcomeText.setText(msg);
        this.addCircle(msg);
    }

    VirtualServer virtualServer;
    private Stage stage;
    UpdateUICallBack updateUICallBack;

    public void setStage(Stage stage, VirtualServer virtualServer) {
        this.stage = stage;
        this.virtualServer = virtualServer;
        // add call back:
        this.updateUICallBack = new UpdateUICallBack() {
            @Override
            public void process(String answer) {
                System.out.println("Call back " + answer);
                safeUpdateView(answer);
                if (answer.equals("DESSERT")){
                    safeCleanUpAll();
                }
            }
        };
        virtualServer.setUICallBack(this.updateUICallBack);
    }

    private void cleanUpAll() {
        Group root = new Group();
        Canvas canvas = new Canvas(300, 250);
        GraphicsContext gc = canvas.getGraphicsContext2D();
        drawShapes(gc);
        root.getChildren().add(canvas);
        stage.setScene(new Scene(root));
        stage.show();
    }

    private void safeCleanUpAll() {
        Platform.runLater(new Runnable() {
            @Override
            public void run() {
                cleanUpAll();
            }
        });
    }

    private void drawShapes(GraphicsContext gc) {
        gc.setFill(Color.GREEN);
        gc.setStroke(Color.BLUE);
        gc.setLineWidth(5);
        gc.strokeLine(40, 10, 10, 40);
        gc.fillOval(10, 60, 30, 30);
        gc.strokeOval(60, 60, 30, 30);
        gc.fillRoundRect(110, 60, 30, 30, 10, 10);
        gc.strokeRoundRect(160, 60, 30, 30, 10, 10);
        gc.fillArc(10, 110, 30, 30, 45, 240, ArcType.OPEN);
        gc.fillArc(60, 110, 30, 30, 45, 240, ArcType.CHORD);
        gc.fillArc(110, 110, 30, 30, 45, 240, ArcType.ROUND);
        gc.strokeArc(10, 160, 30, 30, 45, 240, ArcType.OPEN);
        gc.strokeArc(60, 160, 30, 30, 45, 240, ArcType.CHORD);
        gc.strokeArc(110, 160, 30, 30, 45, 240, ArcType.ROUND);
        gc.fillPolygon(new double[]{10, 40, 10, 40}, new double[]{210, 210, 240, 240}, 4);
        gc.strokePolygon(new double[]{60, 90, 60, 90},new double[]{210, 210, 240, 240}, 4);
        gc.strokePolyline(new double[]{110, 140, 110, 140}, new double[]{210, 210, 240, 240}, 4);
    }
}
```

POLITECNICO MILANO 1863

Update JavaFx UI: all by code...

Potremmo anche non usare del tutto FXML e usare solo codice x disegnare:

Al messaggio "DESSERT" svuotiamo tutto *(Dessert appare solo dopo aver pagato.. 3 Evolve->Pay->Evolve)*



End...