Διαχείριση Βάσης Δεδομένων με γραφική διεπαφή σε Java

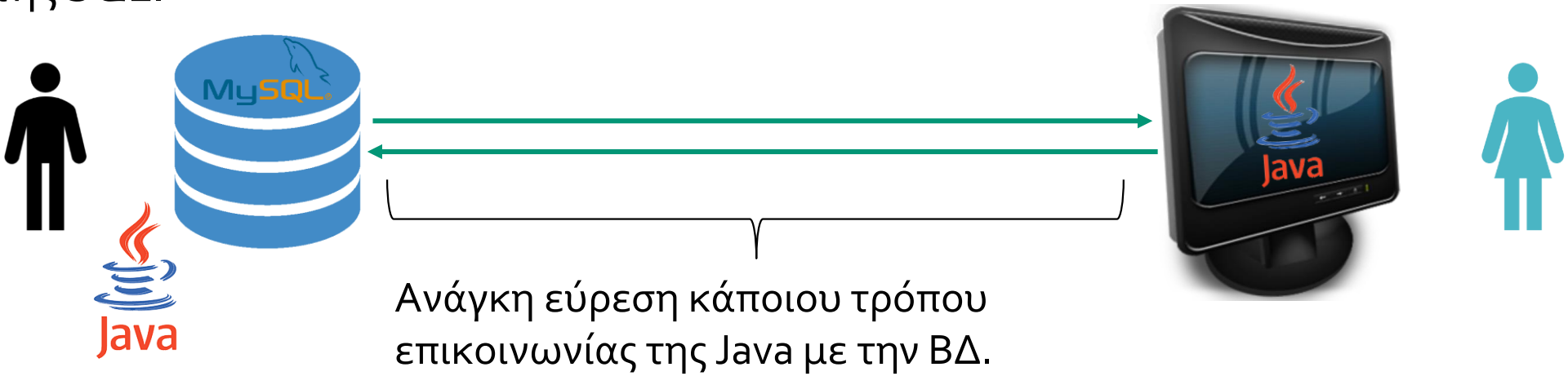
Γιατί γραφική διεπαφή;

- Γιατί να χρειάζεται να υλοποιήσουμε μία γραφική διεπαφή για να επικοινωνήσουμε με μία Βάση Δεδομένων (ΒΔ) αφού μπορούμε να την διαχειριστούμε άμεσα?
- Ας δούμε το ακόλουθο σενάριο:

Αυτός είναι ένας τριτοετής φοιτητής του ΤΜΗΥ&Π. Ας τον πούμε **Bob**.

Και αυτή είναι μία φίλη του Bob η οποία ξέρει να χρησιμοποιεί μόνο απλές εφαρμογές σε ένα PC. Ας την πούμε **Alice**.

- Η **Alice** εργάζεται στην οικογενειακή επιχείρηση ως οινοπαραγωγός και για την καλύτερη διαχείριση της επιχείρισης χρειάζεται να καταγράφει πληροφορία σχετικά με τους πελάτες, τα τιμολόγια, την παραγωγή, τις παραγγελίες κτλ.
- Ο **Bob** προτείνει στην **Alice** την δημιουργία μιας βάσης δεδομένων για τον σκοπό αυτό, αλλά η **Alice** δεν ξέρει SQL ούτε έχει ευχέρεια με το command line ούτε με κάποιο προγραμματιστικό περιβάλλον.
- Έτσι, ο **Bob** θα πρέπει να φτιάξει μια γραφική διεπαφή για να μπορεί η **Alice** να διαχειρίζεται την βάση δεδομένων της επιχείρησης χωρίς να απαιτείται γνώση της SQL.



Ανάγκη εύρεση κάποιου τρόπου επικοινωνίας της Java με την ΒΔ.

Ο **Bob**, ως Μηχανικός Η/Υ, έχει στην διάθεσή του μια πληθώρα εργαλείων για τις ανάγκες του, τα οποία μπορεί και πρέπει να χρησιμοποιήσει όταν πλέον γνωρίζει ένα συγκεκριμένο αντικείμενο.

Αφού ψάχνει τα εργαλεία του, καταλήγει στα ακόλουθα:

MySQL Workbench
ή
XAMPP

Και για την επικοινωνία αυτή, ανακάλυψε το **JDBC**

Γραφική διεπαφή;

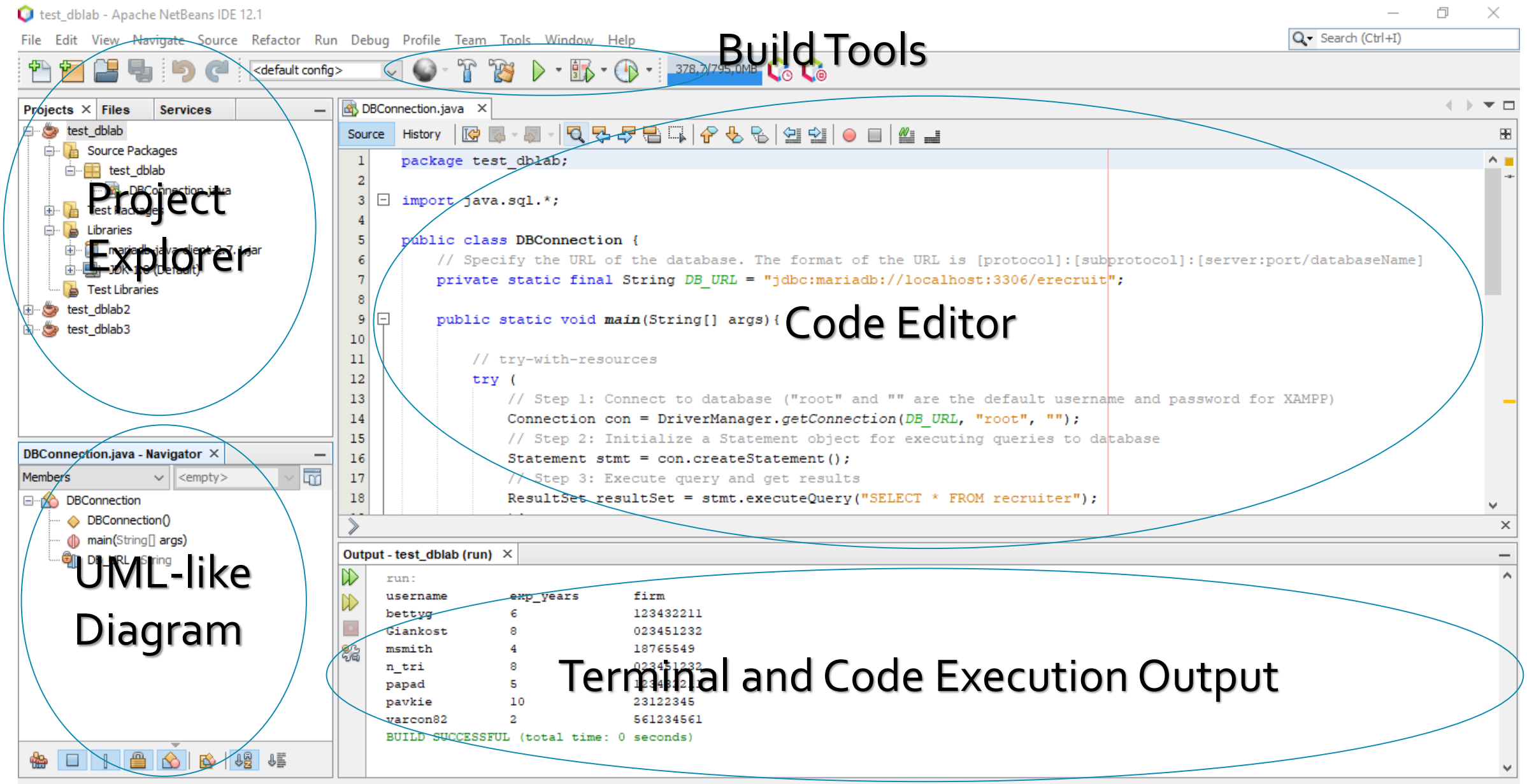
Netbeans IDE

<u>Τι είναι το **JDBC**?</u>

Το **J**ava **D**ata**B**ase **C**onnectivity είναι μια βιβλιοθήκη της Java που περιέχει κλάσεις και μεθόδους (τα εργαλεία σας) για την σύνδεση και διαχείριση μιας ΒΔ.

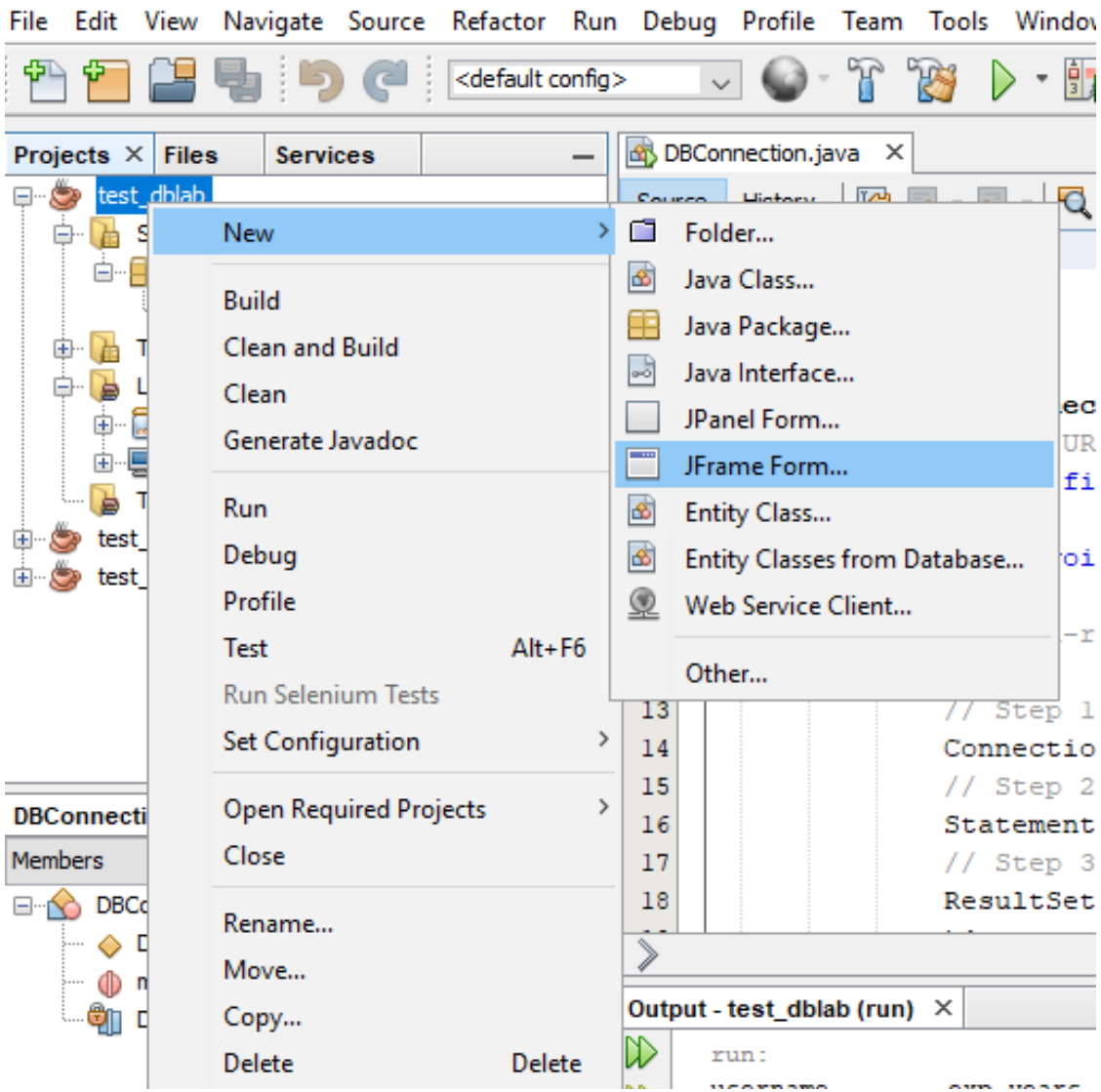
Netbeans

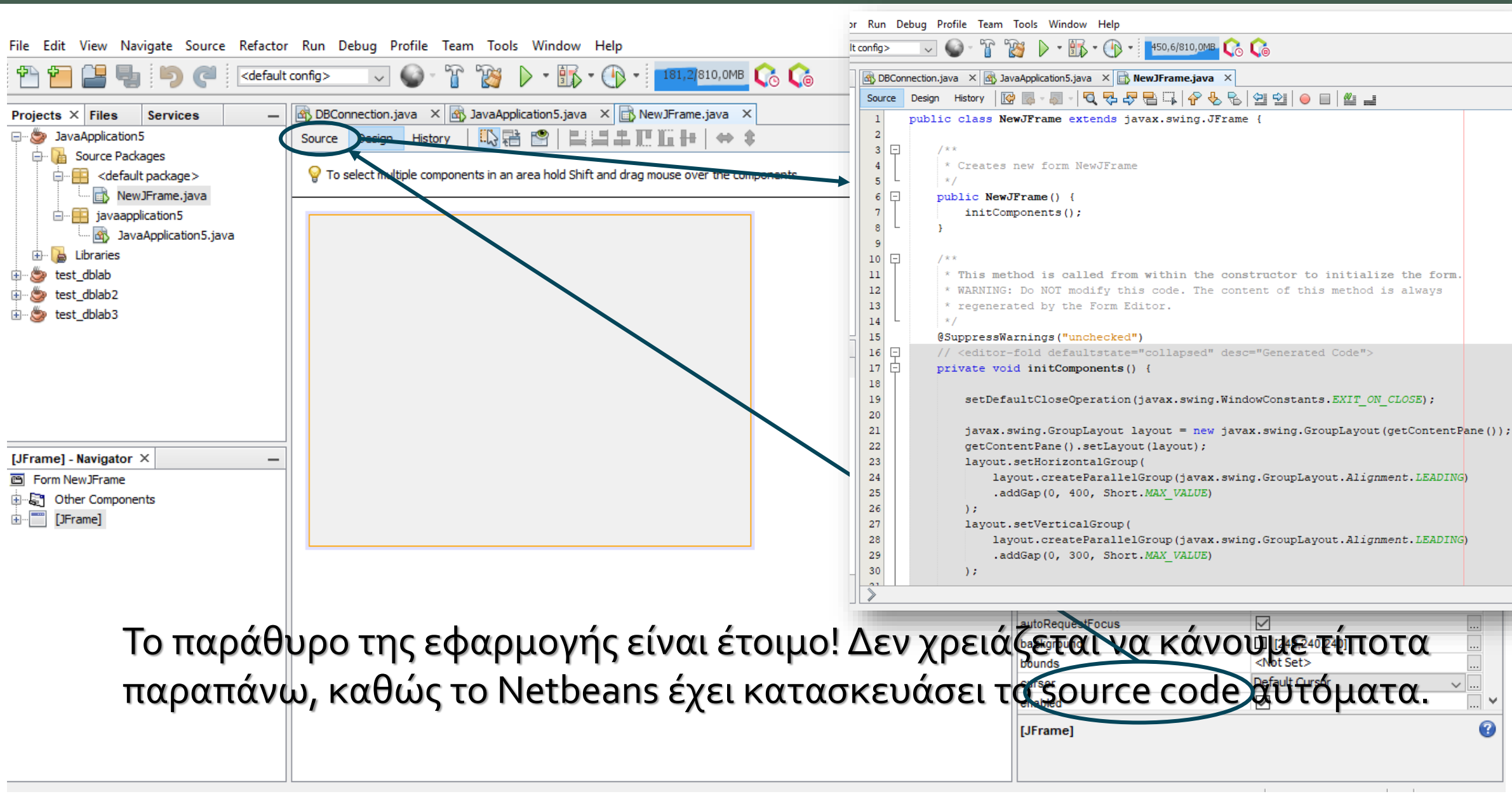
- Είναι ένα **ολοκληρωμένο περιβάλλον ανάπτυξης** (IDE) εφαρμογών με την γλώσσα προγραμματισμού **Java.**
- Έχει επεκτάσεις και για άλλες γλώσσες προγραμματισμού, όπως **PHP, HTML5, CSS** και **JavaScript**.
- Περιέχει ένα εργαλείο σχεδίασης γραφικής διεπαφής (GUI) στο οποίο το ακριβές μέγεθος και η θέση των συστατικών στοιχείων (components) καθορίζονται με οπτικό τρόπο.
- Κάνοντας **drag n drop** τα components μέσα στο παράθυρο του GUI, ο κώδικας παράγεται **αυτόματα**.


test_dblab - Apache NetBeans IDE 12.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
<default config>
Build Tools
Projects Files Services
test_dblab
Source Packages
test_dblab
DBConnection.java
Libraries
Test Libraries
test_dblab2
test_dblab3
Project Explorer
DBConnection.java - Navigator
Members
<empty>
DBConnection
DBConnection()
main(String[] args)
UML-like Diagram
DBConnection.java
Source History
1 package test_dblab;
3 import java.sql.*;
5 public class DBConnection {
6 // Specify the URL of the database. The format of the URL is [protocol]:[subprotocol]:[server:port/databaseName]
7 private static final String DB_URL = "jdbc:mariadb://localhost:3306/erecruit";
9 public static void main(String[] args){
11 // try-with-resources
12 try (
13 // Step 1: Connect to database ("root" and "" are the default username and password for XAMPP)
14 Connection con = DriverManager.getConnection(DB_URL, "root", "");
15 // Step 2: Initialize a Statement object for executing queries to database
16 Statement stmt = con.createStatement();
17 // Step 3: Execute query and get results
18 ResultSet resultSet = stmt.executeQuery("SELECT * FROM recruiter");
Code Editor
Output - test_dblab (run)
run:
username exp_years firm
bettyg 6 123432211
Giankost 8 023451232
msmith 4 18765549
n_tri 8 023451232
papad 5 123432211
pavkie 10 23122345
varcon82 2 561234561
BUILD SUCCESSFUL (total time: 0 seconds)
Terminal and Code Execution Output
1:1 INS

Δημιουργία GUI στο Netbeans

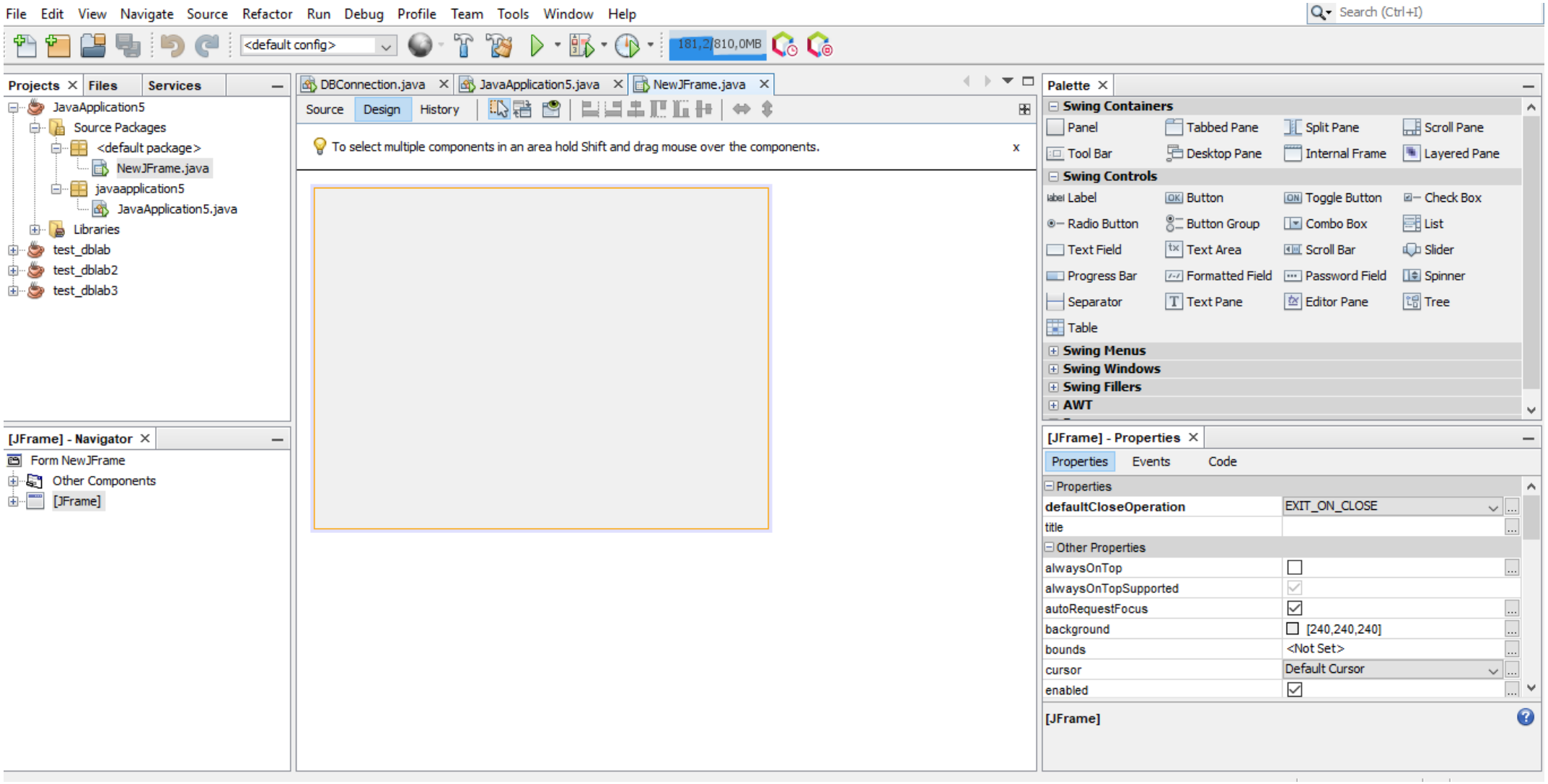
- Εάν σε ένα υπάρχον Project Java επιθυμούμε να κατασκευάσουμε μια εφαρμογή με γραφική διεπαφή, τότε πατώντας δεξί κλικ στον φάκελο του project στον project explorer θα πρέπει να επιλέξουμε την επιλογή JFrame Form.



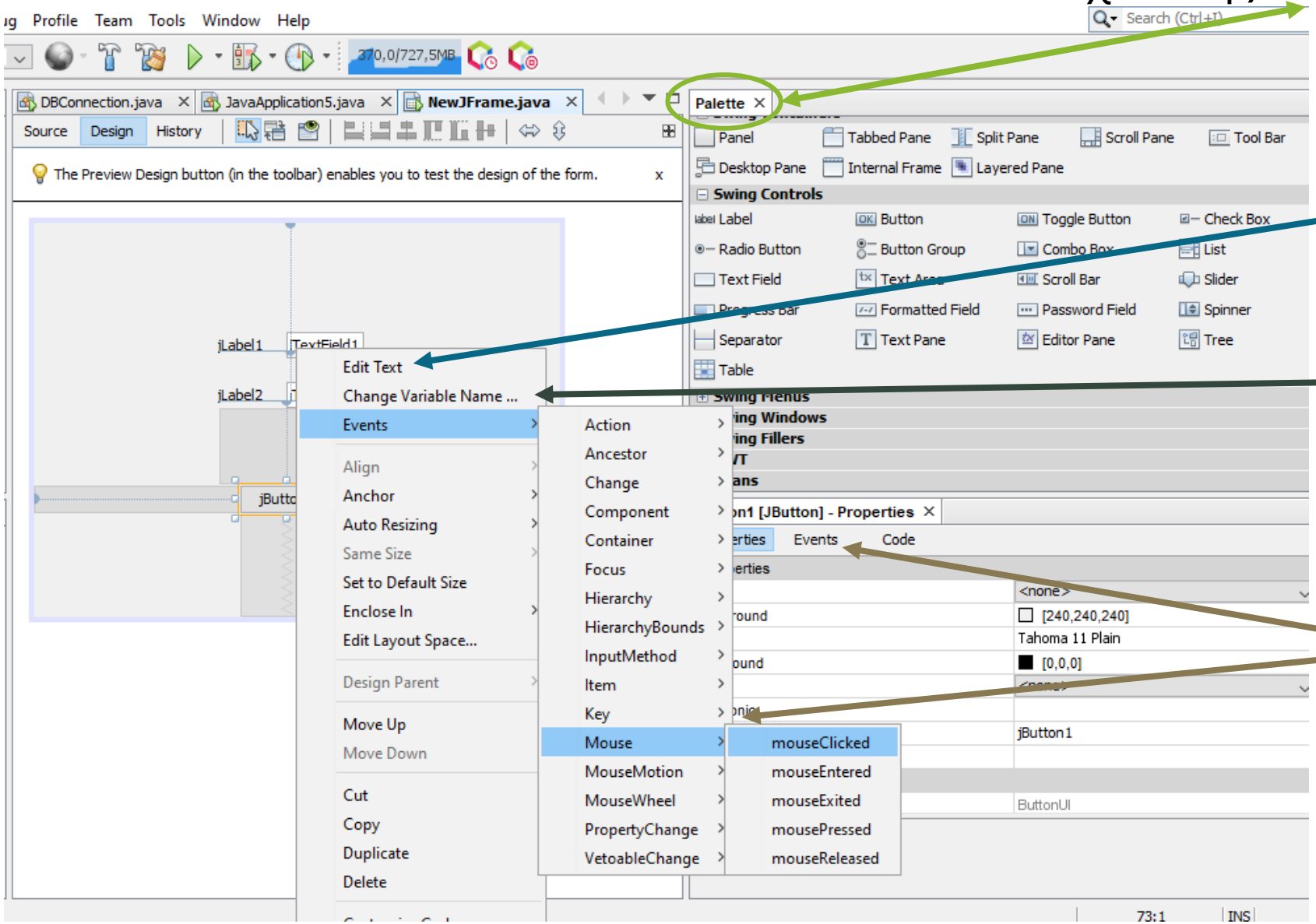


Το παράθυρο της εφαρμογής είναι έτοιμο! Δεν χρειάζεται να κάνουμε τίποτα παραπάνω, καθώς το Netbeans έχει κατασκευάσει το source code αυτόματα.

Για να αρχίσουμε να «γεμίζουμε» το παράθυρο με στοιχεία που χρειαζόμαστε αρκεί ένα **Drag n Drop**.



Κάθε ένα από τα στοιχεία της **παλέτας** περιέχει τα εξής ΠΟΛΥ βασικά στοιχεία:



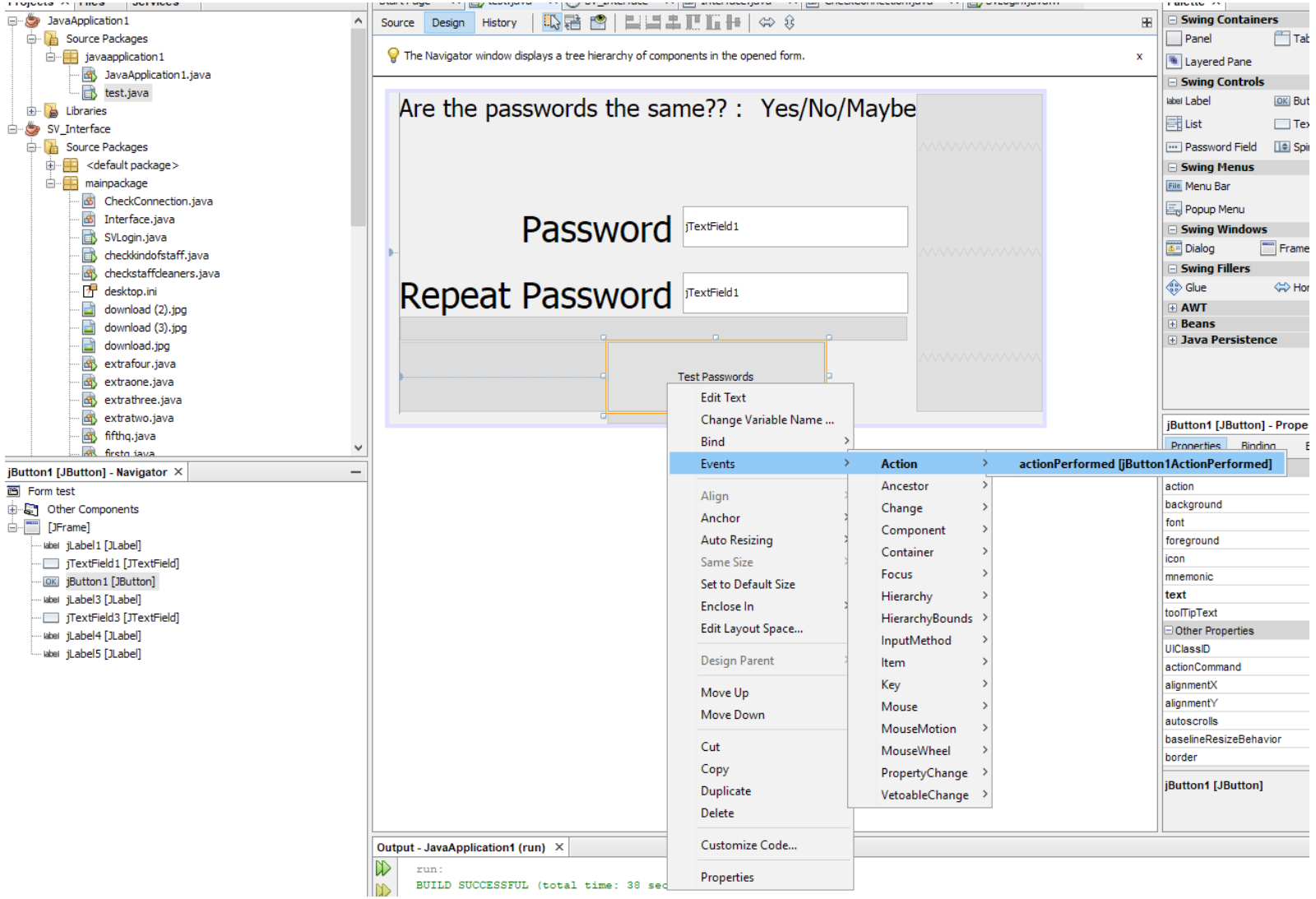
Text: Η ετικέτα του στοιχείου στην γραφική διεπαφή

Variable Name: Το όνομα που θα έχει ως μεταβλητή πλέον το στοιχείο στο **source code**

Πυροδότηση στοιχείου (Event Listener): Εκτέλεση επιθυμητής ενέργειας όταν...

Ένα απλό παράδειγμα

Έστω το ακόλουθο παράθυρο που ελέγχει αν δύο κωδικοί είναι ίδιοι.



Εάν θέλουμε το Yes/No/Maybe που είναι ένα απλό label στοιχείο της παλέτας, να μας λέει εάν οι κωδικοί είναι ίδιοι στο πάτημα του κουμπιού, αυτό που θέλουμε είναι να εκτελεστεί κάποιος κώδικα όταν γίνει κάποιο action στο button με ετικέτα ***Test Passwords***

Ένα απλό παράδειγμα

```
        pack();
    }// </editor-fold>

    private void jTextField1ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        String str1=jTextField1.getText();
        String str2=jTextField3.getText();
        if(str1.equals(str2)){
            jLabel5.setText("Passwords Match");
        }else{
            jLabel5.setText("Passwords do not Match");
        }
    }

    private void jTextField3ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }

    /**
     * @param args the command line arguments
```

Και στην πορεία θα γράψουμε με source code την ενέργεια που θέλουμε να εκτελεστεί, ακριβώς μέσα στην μέθοδο που μας παραπέμπει το Netbeans.

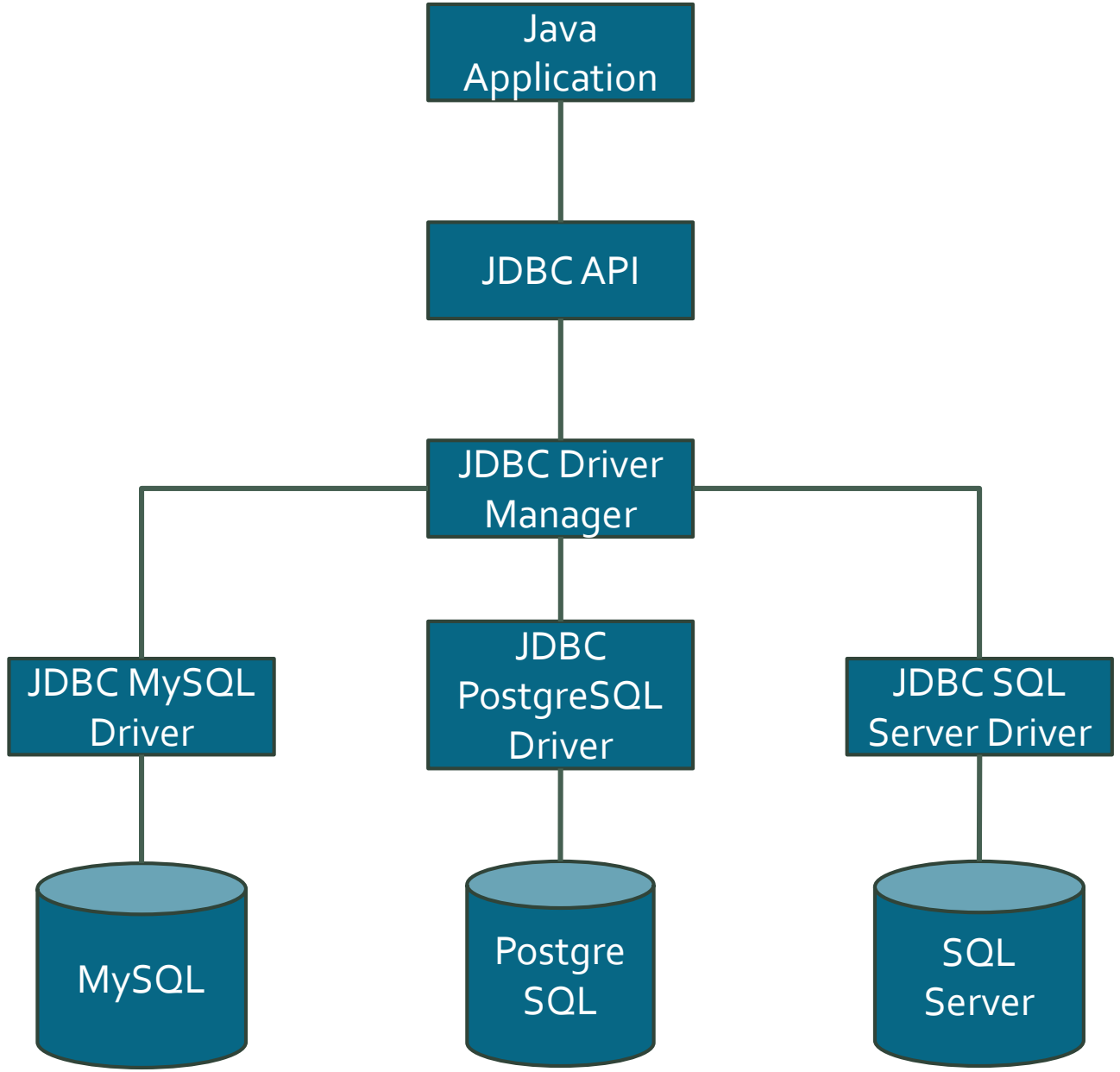
Java DataBase Connectivity API (**JDBC API**)

Java DataBase Connectivity API (JDBC API)

- Βιβλιοθήκη της Java που επιτρέπει την **επικοινωνία** μίας Java εφαρμογής με μία βάση δεδομένων.
- Περιλαμβάνει διασυνδέσεις (interfaces) για την σύνδεση, υποβολή ερωτημάτων και επεξεργασία αποτελεσμάτων από μία βάση δεδομένων.
- Είναι μεταφέρσιμο (**portable**), δηλαδή το ίδιο API μπορεί να χρησιμοποιηθεί για την διασύνδεση με οποιοδήποτε RDBMS αρκεί να χρησιμοποιείται το κατάλληλο πρόγραμμα οδήγησης (**driver**) για αυτό το RDBMS.

Αρχιτεκτονική του JDBC

- Αποτελείται από δύο επίπεδα:
  - **JDBC API**: Χρησιμοποιεί έναν *Driver Manager* για την διασύνδεση της Java εφαρμογής με ετερογενείς ΒΔ.
  - **JDBC Driver Manager**:
    - Διασφαλίζει ότι χρησιμοποιείται ο σωστός driver για την διασύνδεση με κάθε ΒΔ.
    - Επίσης, υποστηρίζει ταυτόχρονη διασύνδεση με ετερογενείς ΒΔ.



Προγράμματα οδήγησης για κάθε RDBMS

- MySQL
  https://dev.mysql.com/downloads/connector/j/
- MariaDB
  https://mariadb.com/kb/en/about-mariadb-connector-j/
- Microsoft SQL Server
  https://docs.microsoft.com/en-us/sql/connect/jdbc/microsoft-jdbc-driver-for-sql-server?view=sql-server-ver15
- PostgreSQL
  https://jdbc.postgresql.org/

Σύνδεση στην Βάση Δεδομένων (1)

- Ορίζουμε ένα αντικείμενο τύπου **Connection**
  - Διαχειρίζεται την σύνδεση ανάμεσα στην Java εφαρμογή και την ΒΔ
  - Επιτρέπει την δημιουργία SQL προτάσεων (statements) από την εφαρμογή Java στην ΒΔ

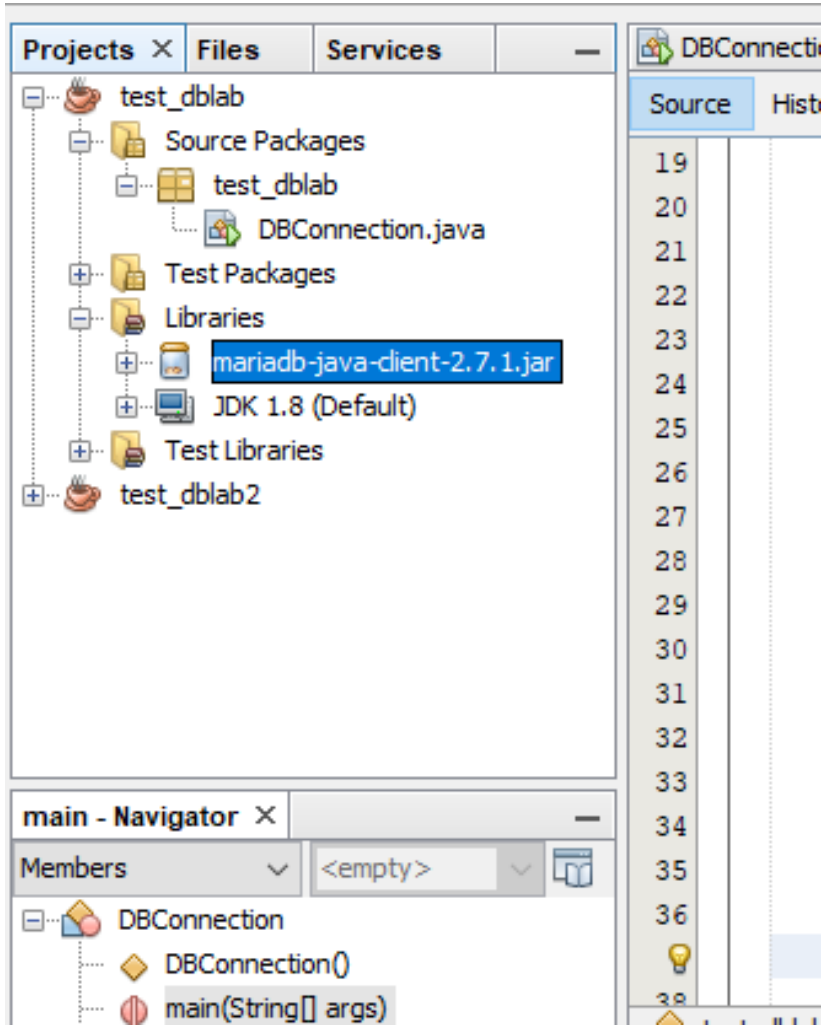
```
Class.forName("com.mysql.jdbc.Driver"); //driver initialization for mysql
Connection connection = DriverManager.getConnection(db_url, username,
                                                    password);
```

Από την έκδοση JDBC API 4.0 η εντολή Class.forName(…) δεν είναι απαραίτητη

db_url = [protocol]:[subprotocol]:[server:port/databaseName]

```
MySQL:      "jdbc:mysql://hostname:portNumber/databaseName"
MariaDB:    "jdbc:mariadb://hostname:portNumber/databaseName"
SQL Server: "jdbc:sqlserver://hostname:portNumber/databaseName"
```

Σύνδεση στην Βάση Δεδομένων (2)



- **Προσοχή**: Θα πρέπει να κατεβάσετε τον JDBC driver (.jar) για την αντίστοιχο RDBMS(π.χ. MySQL) που θα χρησιμοποιήσετε και να τον προσθέσετε στα libraries (classpath) του project.

- Αν ο DriverManager δεν μπορέσει να συνδεθεί στην ΒΔ προκαλείται **SQLException**.

Δημιουργία πρότασης για την εκτέλεση ερωτημάτων

- Ορίζουμε ένα αντικείμενο τύπου **Statement**
  - Υποβάλλει SQL προτάσεις στην ΒΔ

```
Statement statement = connection.createStatement();
```

- Εκτέλεση ερωτήματος
  - Χρησιμοποιούμε την μέθοδο *executeQuery* της κλάσης Statement
  - Ένα αντικείμενο ResultSet αποθηκεύει το αποτέλεσμα του SQL ερωτήματος.

```
String SELECT_QUERY = "SELECT * FROM … ";
ResultSet resultSet = statement.executeQuery(SELECT_QUERY);
```

Επεξεργασία του ResultSet ενός ερωτήματος (1)

- Λήψη μετα-δεδομένων των αποτελεσμάτων του ερωτήματος

```
ResultSetMetaData metadata = resultSet.getMetaData();
```

- Ανάκτηση πλήθους και ονομάτων στηλών (attributes)

```
int numberOfColumns = metaData.getColumnCount();
for (int i = 1; i <= numberOfColumns; i++)
   System.out.printf("%-8s\t", metadata.getColumnName(i));
System.out.println();
```

Οι αριθμοί των στηλών ξεκινούν από το 1!

Επεξεργασία του ResultSet ενός ερωτήματος (2)

- Προσπέλαση των αποτελεσμάτων του ResultSet
  - μέθοδος **next()**: τοποθετεί τον cursor στην επόμενη γραμμή των αποτελεσμάτων

```
while (resultSet.next())
{
   for (int i = 1; i <= numberOfColumns; i++)
      System.out.printf("%-8s\t", resultSet.getObject(i));
   System.out.println();
}
```

Τερματισμός σύνδεσης με την ΒΔ (1)

- Η σύνδεση με την ΒΔ θα πρέπει να **τερματίζεται** όταν ολοκληρωθούν οι οποιεσδήποτε διεργασίες έτσι ώστε να ελευθερωθούν οι πόροι του **λειτουργικού συστήματος** (πχ sockets) αλλά και αυτοί του **server** (πχ. cursors).
- Αν η σύνδεση δεν τερματιστεί ρητά από την Java εφαρμογή τότε ο **garbage collector** της Java θα τερματίσει την σύνδεση όταν θα καθαρίσει τα αντίστοιχα αντικείμενα. **poor programming practice!**
- Μπορούμε με δύο τρόπους να τερματίσουμε ρητά την σύνδεση με την ΒΔ.

Τερματισμός σύνδεσης με την ΒΔ (2)

```
try {

    /* Connection to DB, SQL queries */

} catch (SQLException e){

    /* Exception handling */

} finally {

    if (resultSet != null){
        try {
            resultSet.close();
        } catch (SQLException e) { }
    }

    if (statement != null){
        try {
            statement.close();
        } catch (SQLException e) { }
    }

    if (connection != null){
        try {
            connection.close();
        } catch (SQLException e) { }
    }
}
```

Α΄ τρόπος

Εφαρμογή της μεθόδου close() στα αντικείμενα τύπου **ResultSet**, **Statement** και **Connection** στο **finally** μπλοκ

Τερματισμός σύνδεσης με την ΒΔ (3)

```
try (
    Connection connection =
DriverManager.getConnection(DATABASE_URL, username, password);
    Statement statement = connection.createStatement();
    ResultSet resultSet = statement.executeQuery(SELECT_QUERY);
) {
    /* Process resultSet */
} catch (SQLException e){
    /* Exception handling */
}
```

Β΄ τρόπος (Java SE 7 -)

- Τα αντικείμενα JDBC που χρησιμοποιούν πόρους υλοποιούν το **AutoCloseable Interface**
- Χρήση του μπλοκ ***try-with-resources***
- Η μέθοδος **close()** καλείται **αυτόματα** μόλις η ροή του προγράμματος εξέλθει από το try μπλοκ

```java
import java.sql.*;

public class DBConnection {
    // Specify the URL of the database. The format of the URL is [protocol]:[subprotocol]:[server:port/databaseName]
    private static final String DB_URL = "jdbc:mariadb://localhost:3306/erecruit";

    public static void main(String[] args){
        // try-with-resources
        try (
            // Step 1: Connect to database ("root" and "" are the default username and password for XAMPP)
            Connection con = DriverManager.getConnection(DB_URL, "root", "");
            // Step 2: Initialize a Statement object for executing queries to database
            Statement stmt = con.createStatement();
            // Step 3: Execute query and get results
            ResultSet resultSet = stmt.executeQuery("SELECT * FROM recruiter");){
            // Step 4: Process and print results
            ResultSetMetaData metaData = resultSet.getMetaData();
            int numberOfColumns = metaData.getColumnCount();
            // Print column names
            for (int i = 1; i <= numberOfColumns; i++)
                System.out.printf("%-8s\t", metaData.getColumnName(i));
            System.out.println();
            // Move cursor to the next row of the results and print the value of each column
            while (resultSet.next()){
                for (int i = 1; i <= numberOfColumns; i++)
                    System.out.printf("%-8s\t", resultSet.getObject(i));
                System.out.println();
            }
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
    }
}
```

PreparedStatement Interface (1)

- Η διασύνδεση PreparentStatement χρησιμοποιείται για τον ορισμό και την εκτέλεση ερωτημάτων με παραμέτρους.
- Μία PreparedStatement είναι πιο αποτελεσματική από την Statement:
  - Το SQL ερώτημα που περιέχει είναι ήδη μεταγλωττισμένο (pre-compiled)
  - Το ίδιο ερώτημα μπορεί να εκτελεστεί επανειλλημένα με διαφορετικά ορίσματα
  - Μπορούν να εμποδίσουν τις επιθέσεις μέσω SQL προτάσεων (*SQL injection attacks*). Τέτοιες επιθέσεις πραγματοποιούνται από κακόβουλους χρήστες οι οποίοι δίνουν ως είσοδο ειδικά διαμορφωμένα αλφαριθμητικά για να αποκτήσουν πρόσβαση σε απαγορευμένα δεδομένα της βάσης. Στις PreparedStatement η είσοδος του χρήστη περιορίζεται μόνο στις παραμέτρους.

PreparedStatement Interface (2)

- Ορισμός ερωτήματος με την διασύνδεση PreparedStatement:

```
Connection con = DriverManager.getConnection(db_url, username,
                                             password);
PreparedStatement selectJobs = con.prepareStatement("SELECT * FROM
job WHERE edra LIKE ? AND salary > ?");
selectJobs.setString(1, "%Patra%");
selectJobs.setFloat(2, 1500.0);
ResultSet resultSet = selectJobs.executeQuery();
```

- Τα ερωτηματικά «**?**» ορίζουν σε ποιες θέσεις της SQL πρότασης θα εισαχθούν οι τιμές των παραμέτρων από τον χρήστη (**μόνο για τιμές πεδίων**)
- Ο καθορισμός των τιμών αυτών γίνεται με τις μεθόδους **set** της διασύνδεσης PreparedStatement πριν εκτελεστεί το ερώτημα

PreparedStatement Interface (3)

- Για κάθε τύπο μεταβλητών της SQL υπάρχει και μία μέθοδος set της διασύνδεσης PreparedStatement:
  - `setObject(int parameterIndex, Object x)`
  - `setString(int parameterIndex, String x)`
  - `setInt(int parameterIndex, int x)`
  - `setFloat(int parameterIndex, float x)`
  - `setDouble(int parameterIndex, double x)`
  - `setBoolean(int parameterIndex, boolean x)`
  - `setDate(int parameterIndex, Date x)`
  - `setTimestamp(int parameterIndex, Timestamp x)`
- Το πρώτο ερωτηματικό στην πρόταση έχει `parameterIndex = 1`, όχι `0`!

PreparedStatement Interface (4)

- Για την εκτέλεση των SQL ερωτημάτων η διασύνδεση PreparedStatement περιλαμβάνει τις εξής μεθόδους:
  - **`execute()`**:
    - Χρησιμοποιείται για την εκτέλεση οποιουδήποτε SQL ερωτήματος (`SELECT, INSERT, UPDATE, DELETE…`)
    - Επιστρέφει μία **boolean** τιμή: **True** αν η πρώτη γραμμή του αποτελέσματος είναι ένα αντικείμενο ResultSet, **False** διαφορετικά (πλήθος εγγραφών που επηρεάστηκαν ή καθόλου αποτέλεσμα)
  - **`executeUpdate()`**:
    - Χρησιμοποιείται για την εκτέλεση ερωτημάτων `INSERT, UPDATE, DELETE` ή οποιουδήποτε ερωτήματος που δεν επιστρέφει κάποιο αποτέλεσμα (`CREATE, ALTER`)
    - Επιστρέφει έναν **ακέραιο** που δηλώνει το πλήθος των εγγραφών που επηρεάστηκαν από το ερώτημα
  - **`executeQuery()`**:
    - Χρησιμοποιείται για την εκτέλεση ερωτημάτων `SELECT`
    - Επιστρέφει ένα **αντικείμενο ResultSet** που περιλαμβάνει τα δεδομένα του αποτελέσματος

Οι παραπάνω μέθοδοι χρησιμοποιούνται και για ερωτήματα τύπου Statement, αλλά πρέπει να περαστεί ως όρισμα το SQL ερώτημα!

CallableStatement Interface (1)

- Το JDBC API επιτρέπει την κλήση stored procedures χρησιμοποιώντας αντικείμενα που υλοποιούν την διασύνδεση **CallableStatement**.
- Η σύνταξη του SQL ερωτήματος για την κλήση μιας stored procedure ορίζεται από το JDBC ως εξής:

```
{call procedure_name(<arg1>,<arg2>, ...)}
```

- Για παράδειγμα αν έχουμε αποθηκεύσει στην βάση μία stored procedure με το εξής πρωτότυπο: **`countJobs(IN arg1 VARCHAR(12), OUT arg2 INT)`** τότε ορίζουμε ένα αλφαριθμητικό χρησιμοποιώντας την παραπάνω σύνταξη:

```
String stored = "{call countJobs(?, ?)}";
```

CallableStatement Interface (2)

- Η αρχικοποίηση ενός αντικειμένου CallableStatement πραγματοποιείται με την εφαρμογή της μεθόδου **prepareCall** της διασύνδεσης Connection εισάγοντας ως όρισμα το αλφαριθμητικό της κλήσης της stored procedure.

```
CallableStatement callCountJobs = con.prepareCall(stored);
```

- Το πέρασμα τιμών στις **IN** παραμέτρους της stored procedure γίνεται χρησιμοποιώντας με τον ίδιο τρόπο τις μεθόδους **set** της διασύνδεσης PreparedStatement.

```
callCountJobs.setString(1, "n_tri");
```

CallableStatement Interface (3)

- Πριν την εκτέλεση του CallableStatement πρέπει να προσδιορίσουμε τις παραμέτρους OUT (αν υπάρχουν) και τον SQL τύπο τους.
- Χρησιμοποιούμε την μέθοδο **registerOutParameter** ως εξής:

```
callCountJobs.registerOutParameter(2, java.sql.Types.INTEGER);
```

- Στην συνέχεια εκτελούμε το CallableStatement αντικείμενο με την κατάλληλη μέθοδο **execute** της διασύνδεσης PreparedStatement.
- Για να αποθηκεύσουμε τις τιμές των OUT παραμέτρων σε μεταβλητές της Java χρησιμοποιούμε τις μεθόδους **get** της διασύνδεσης CallableStatement.

```
int count = callCountJobs.getInt(2);
```

```
import java.sql.*;

public class DBConnection {

    private static final String DB_URL = "jdbc:mariadb://localhost:3306/erecruit";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "";
    private static Connection con;

    public static void main(String[] args) {
        try {
            con = DriverManager.getConnection(DB_URL, USERNAME, PASSWORD);
            // countJobs(IN rec_username VARCHAR(12), OUT totalJobs INT)
            String stored = "{call countJobs (?, ?)}";
            CallableStatement callCountJobs = con.prepareCall(stored);
            String rec_username = "n_tri";
            callCountJobs.setString(1, rec_username);
            callCountJobs.registerOutParameter(2, java.sql.Types.INTEGER);
            callCountJobs.execute();
            int count = callCountJobs.getInt(2);
            System.out.println("Recruiter with username " + rec_username + " is responsible for " + count + " jobs.");
        } catch (SQLException ex) {
            ex.printStackTrace();
        }
        finally{
            if (con != null)
                try { con.close();}
                 catch (SQLException ex) {  ex.printStackTrace();}
        }
    }
}
```

Αναφορές

- Για **MySQL Workbench**:
  https://docs.oracle.com/cd/E19078-01/mysql/mysql-workbench/wb-getting-started-tutorial.html

- Για **Netbeans - Swing/AWT** και **JDBC**:
  https://netbeans.org/
  https://docs.oracle.com/javase/tutorial/uiswing/index.html
  https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html
  https://www.tutorialspoint.com/jdbc/index.htm