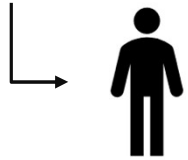
Διαχείριση Βάσης Δεδομένων με γραφική διεπαφή σε Java

- 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.

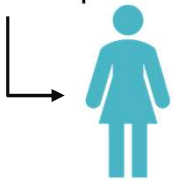
Αυτός είναι ένας τριτοετής φοιτητής του τμήματος Μηχανικών Η/Υ κ Πληροφορικής.



Ας τον πούμε **Bob**.

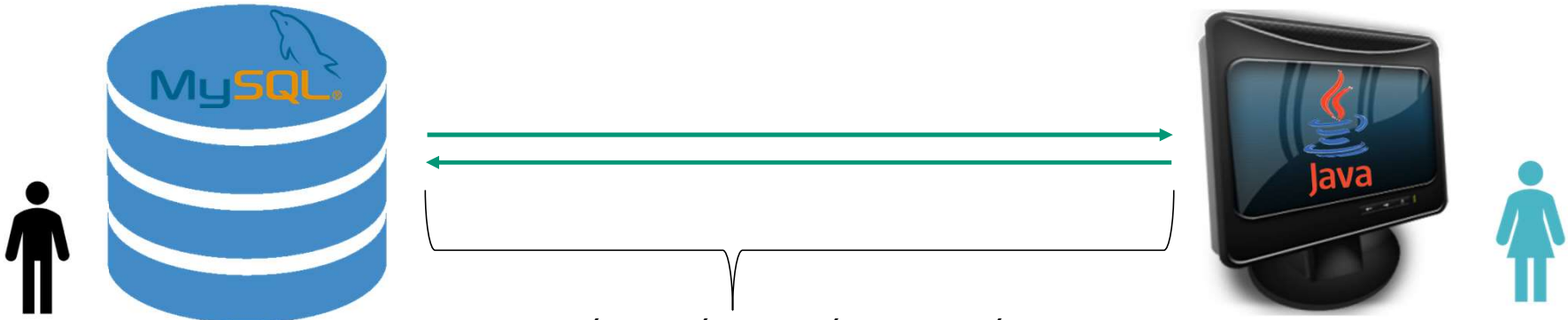
Ο **Bob** έχει μάθει τα απαραίτητα για την κατασκευή μιας Βάσης Δεδομένων με MySQL

Και αυτή είναι η **Alice** που ξέρει να χειρίζεται μόνο απλά προγράμματα σε έναν υπολογιστή.



Ο **Bob** θέλει να κατασκευάσει ένα πρόγραμμα για να βοηθήσει την **Alice** να χρησιμοποιήσει μια ΒΔ και σαν πρώτη επιλογή σκέφτηκε την Java την οποία την έχει διδαχτεί και *μπορεί εύκολα να βρει πολλές πληροφορίες και βοήθεια εάν χρειαστεί κάτι…*

Συνεπώς μέχρι τώρα στο σενάριο υπάρχουν τα εξής στοιχεία:



Η MySQL ΒΔ

Και η ανάγκη εύρεση κάποιου τρόπου επικοινωνίας της Java με την ΒΔ.

Η ανάγκη δημιουργίας ενός Java GUI για την Alice

Ο Bob, ως Μηχανικός Υπολογιστών, 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

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

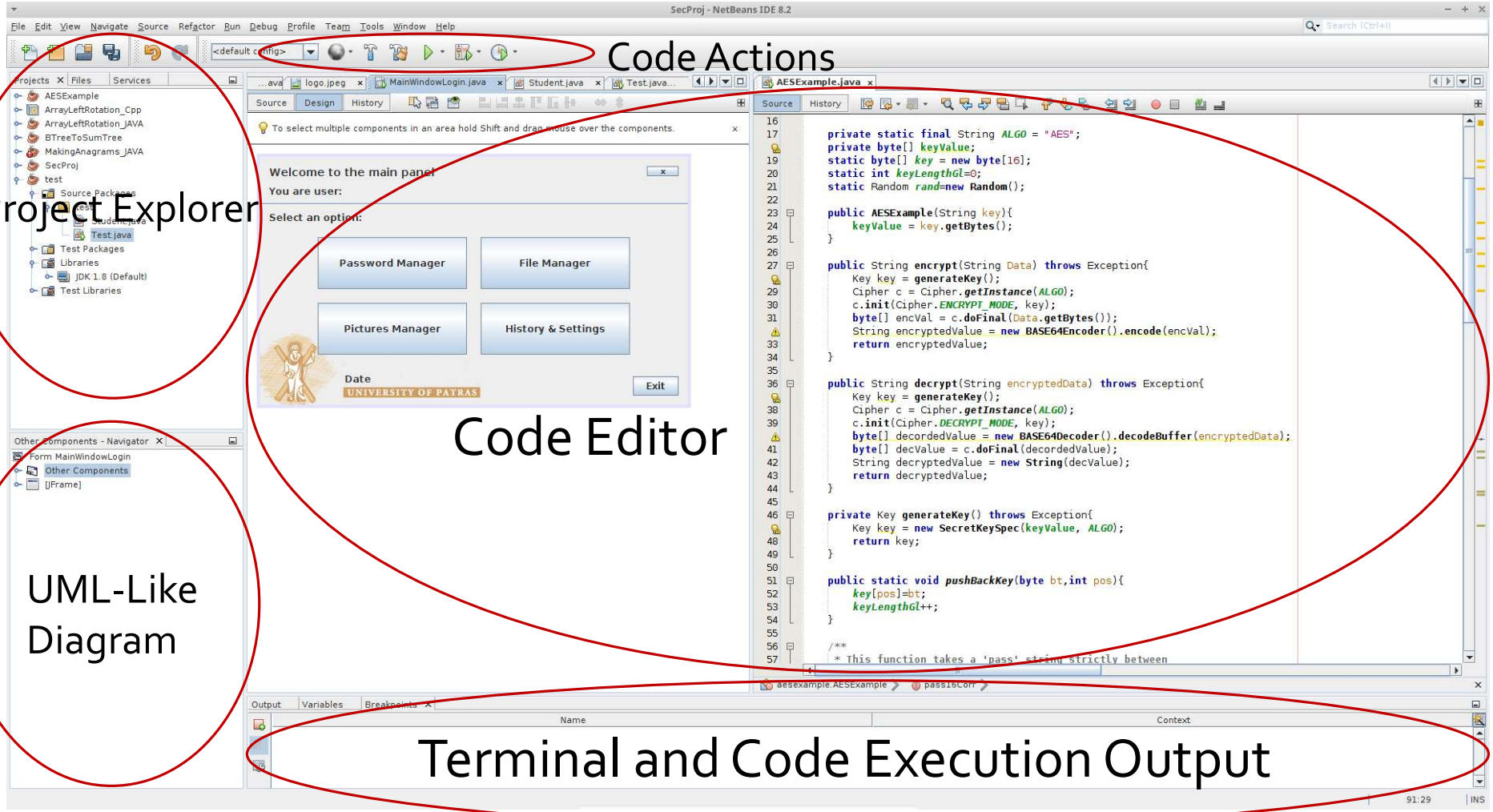
Netbeans

➢ Τι είναι το **JDBC**?

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

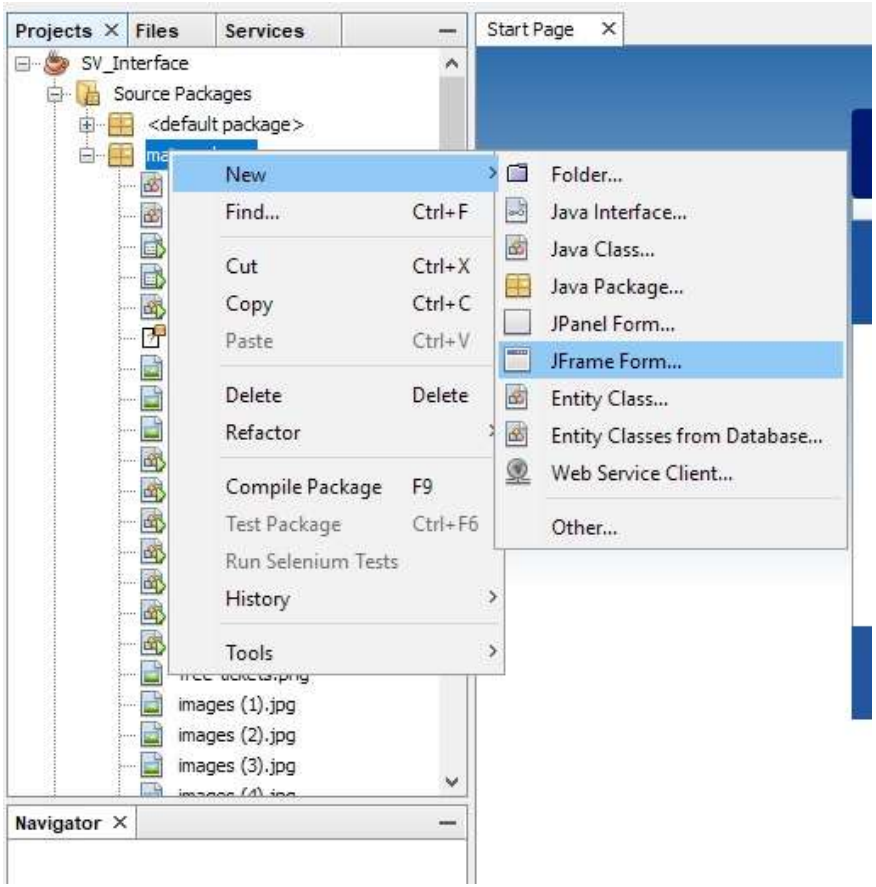
Netbeans

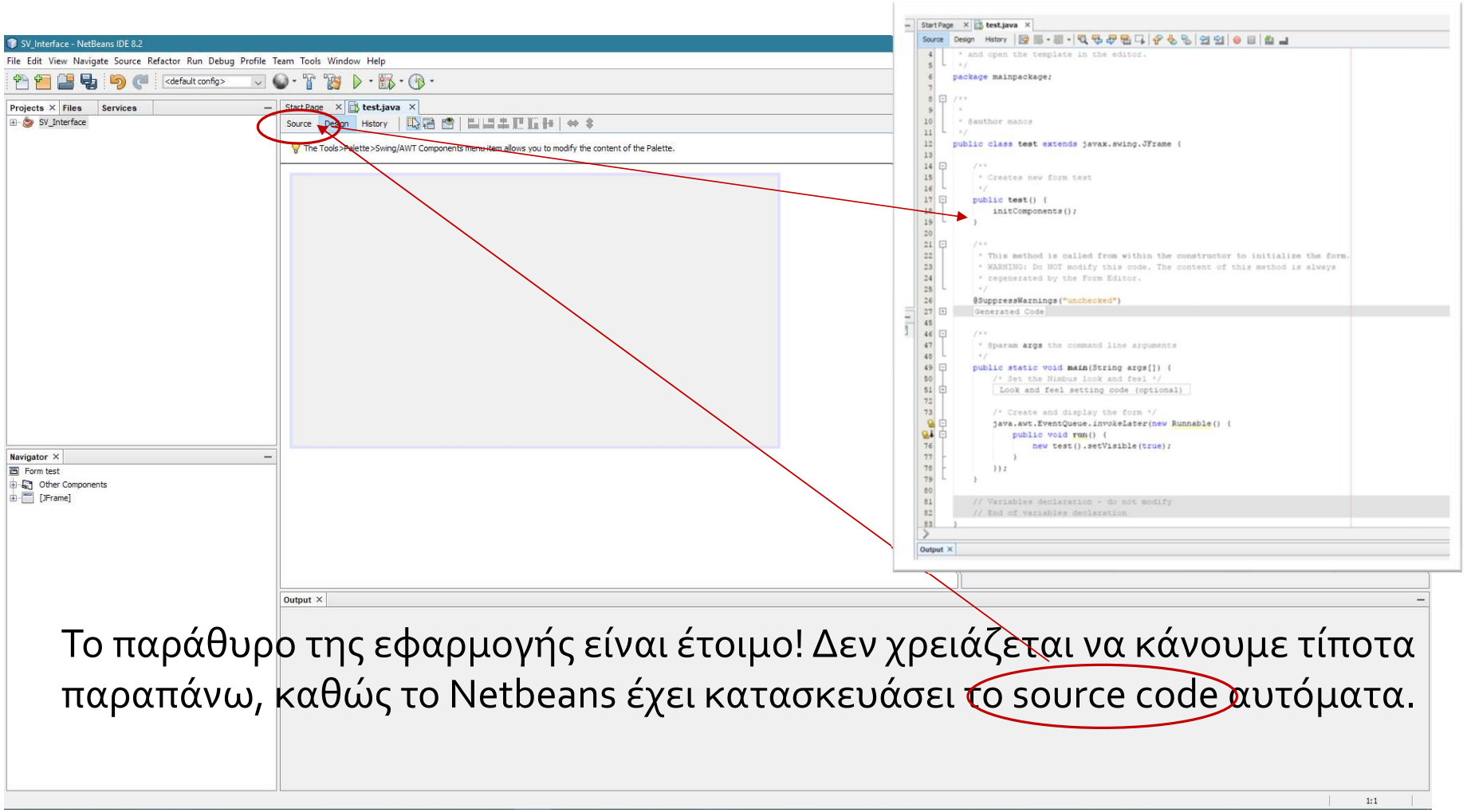
- Είναι ένα **ολοκληρωμένο περιβάλλον ανάπτυξης** (IDE) εφαρμογών με την γλώσσα προγραμματισμού **Java.**
- Έχει επεκτάσεις και για άλλες γλώσσες προγραμματισμού, όπως **PHP, HTML5, CSS** και **JavaScript**.
- Περιέχει ένα εργαλείο σχεδίασης γραφικής διεπαφής (GUI) στο οποίο το ακριβές μέγεθος και η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


Code Actions
Project Explorer
Code Editor
UML-Like Diagram
Terminal and Code Execution Output

Netbeans

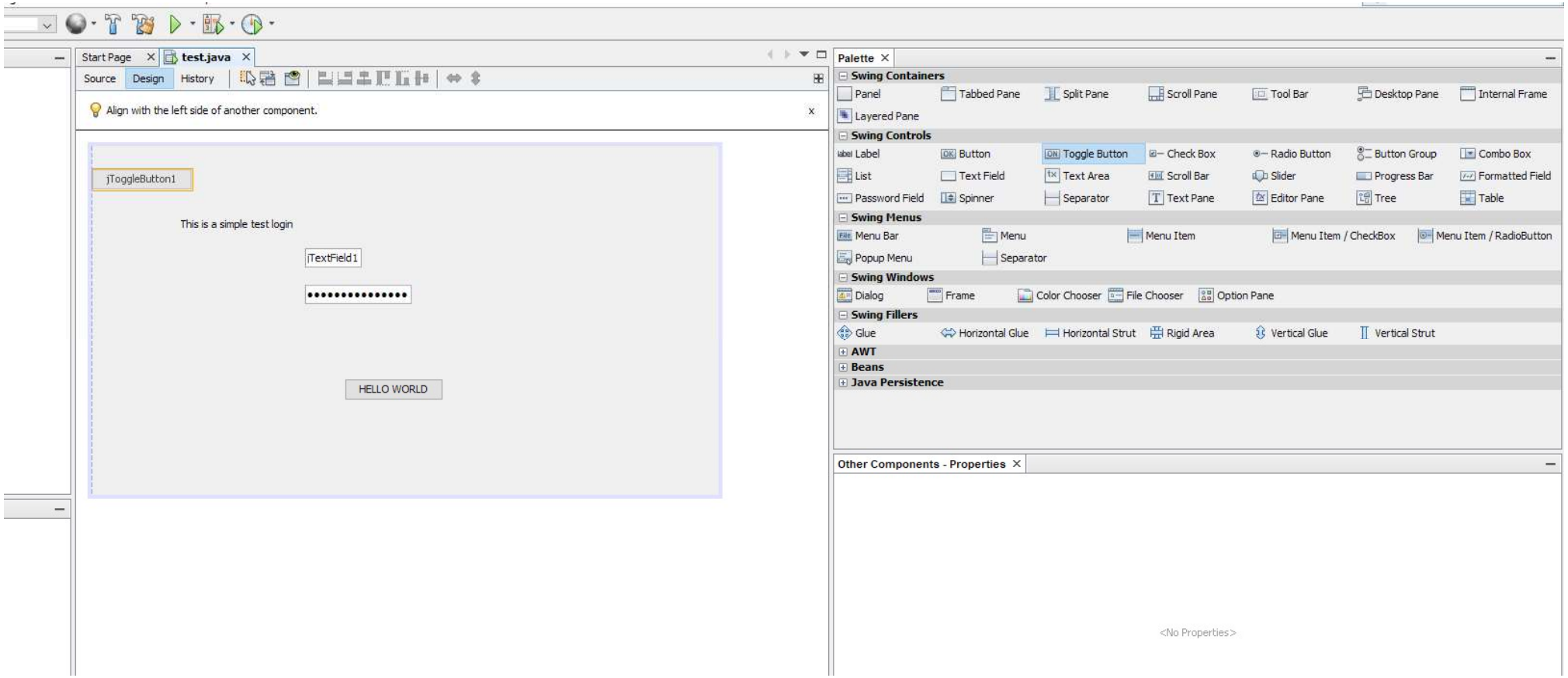
- Εάν σε ένα υπάρχον Project Java επιθυμούμε να κατασκευάσουμε μια εφαρμογή με γραφική διεπαφή τότε πρέπει απλά να επιλέξουμε το **JFrame Form**.



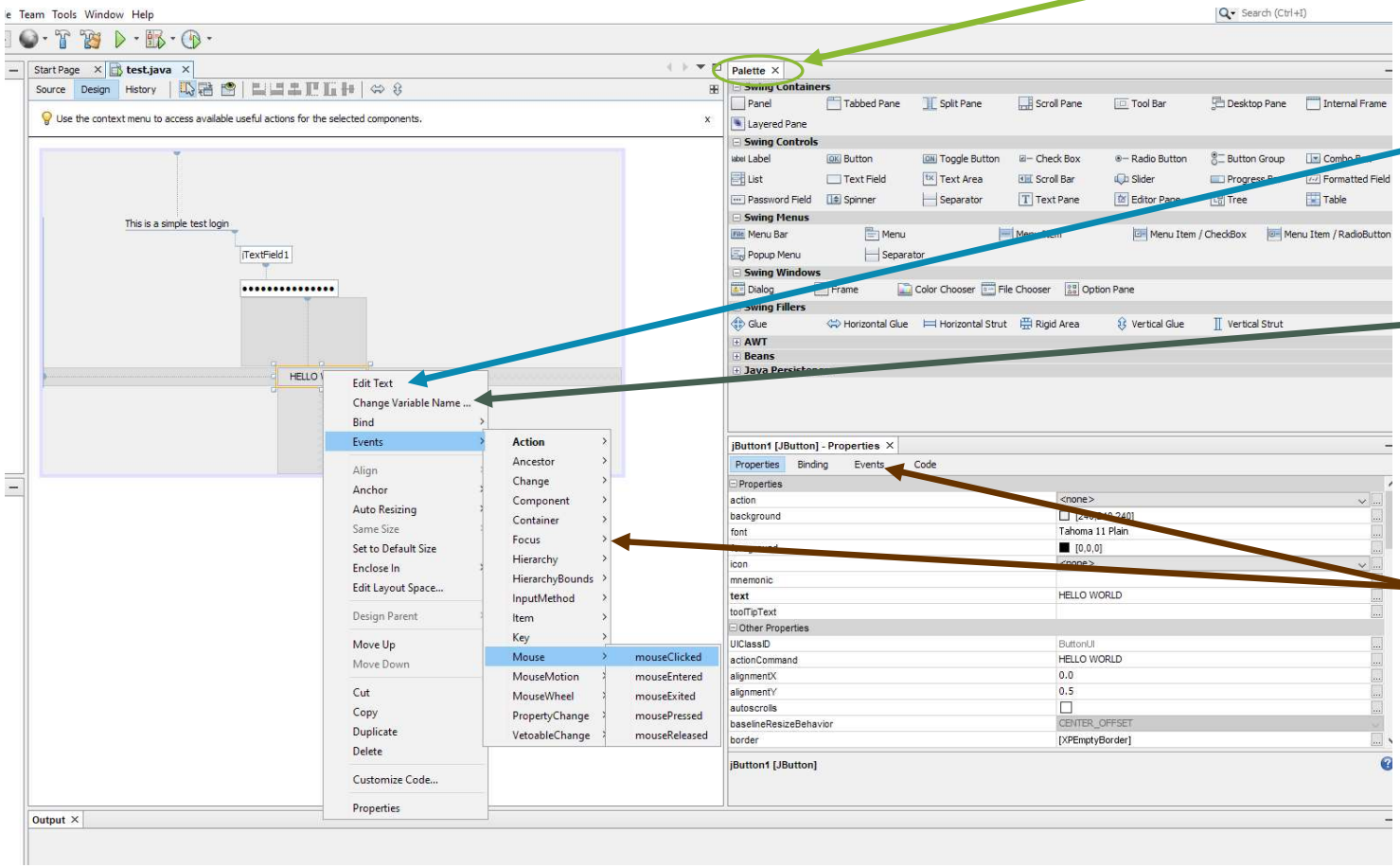


Το παράθυρο της εφαρμογής είναι έτοιμο! Δεν χρειάζεται να κάνουμε τίποτα παραπάνω, καθώς το Netbeans έχει κατασκευάσει το source code αυτόματα.

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:

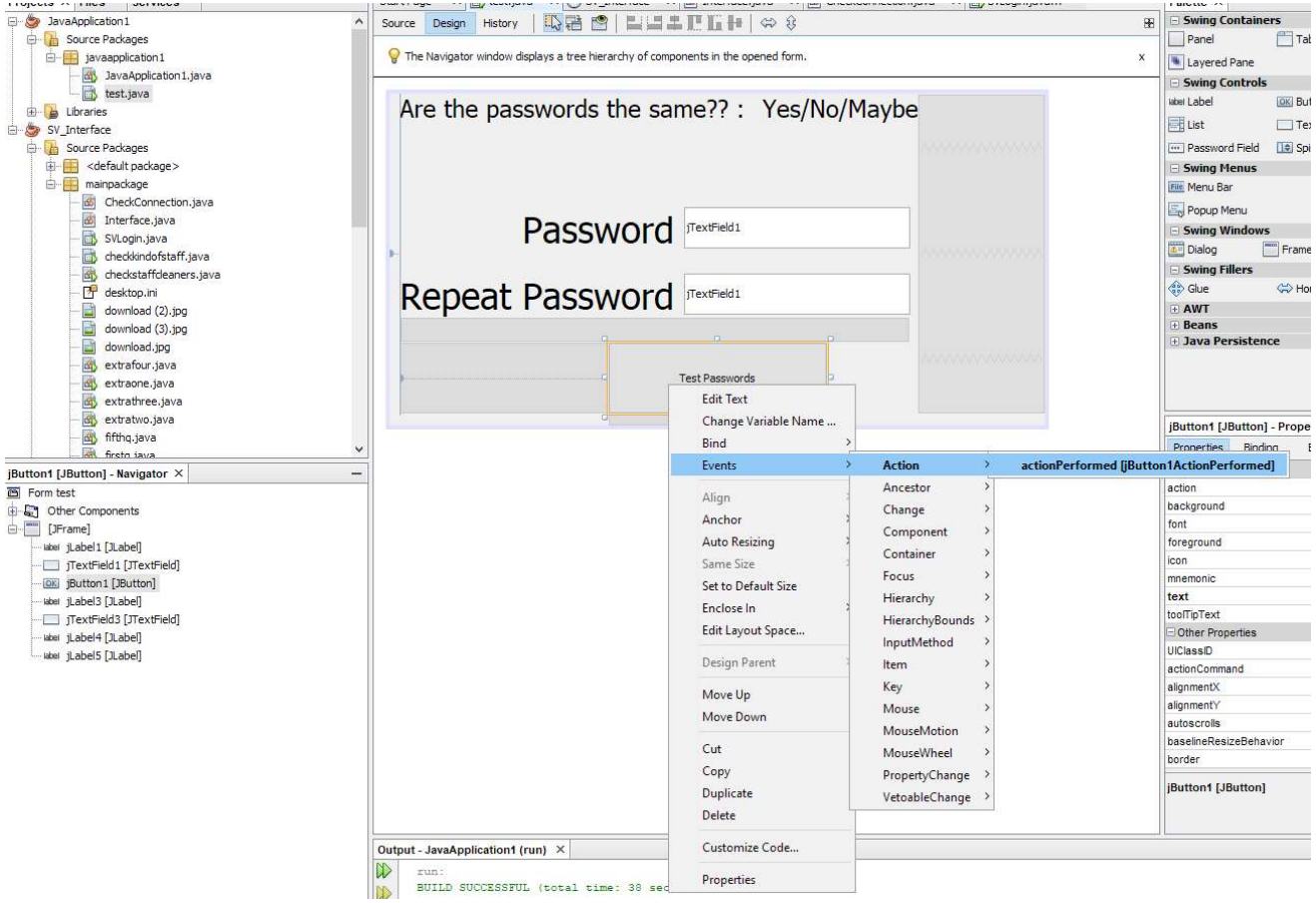


Text
Η ετικέτα του στοιχείου στην γραφική διεπαφή

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 τους κωδικούς



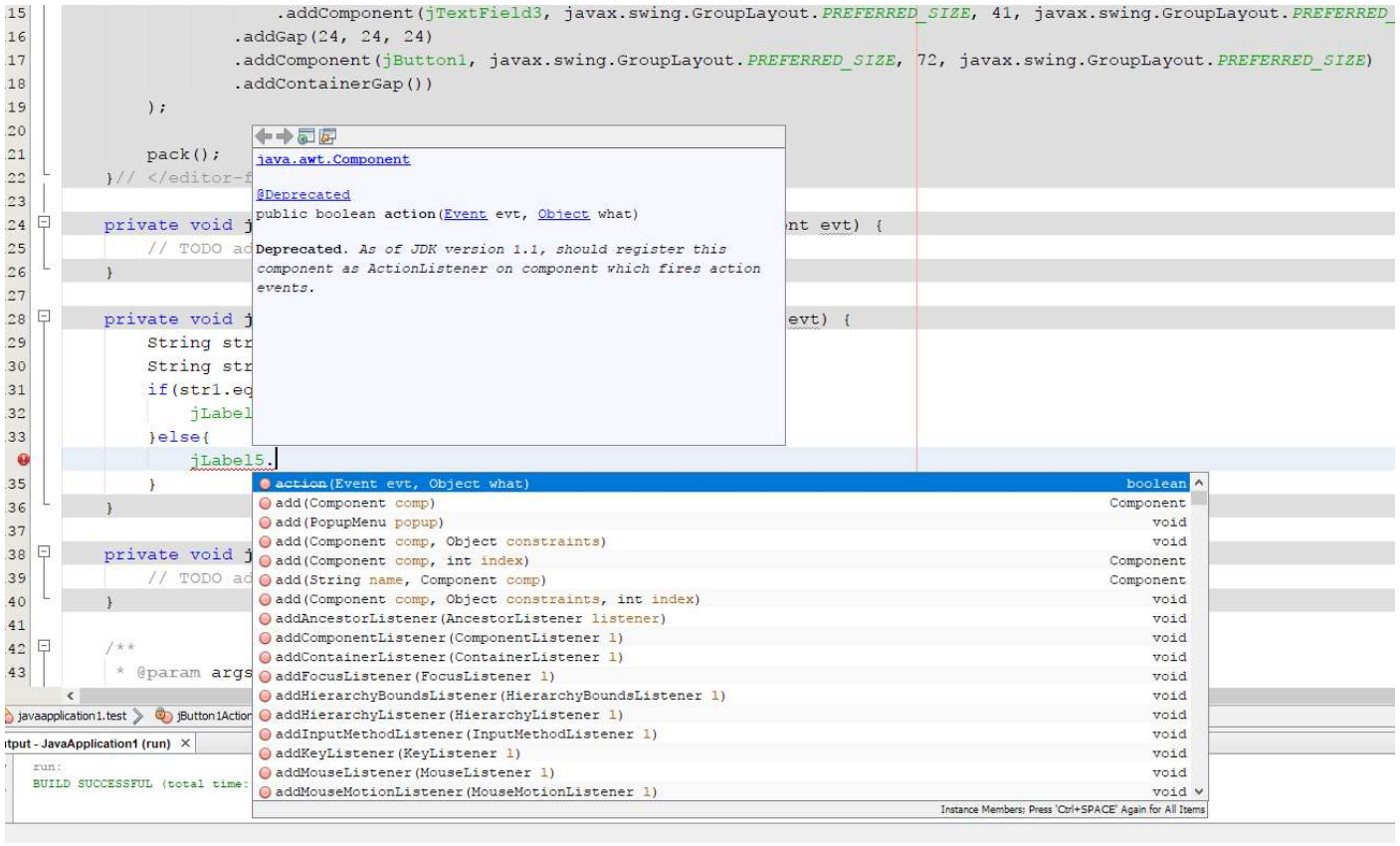
Εάν θέλουμε το Yes/No/Maybe που είναι ένα απλό label στοιχείο της παλέτας, να μας λέει εάν οι κωδικοί είναι ίδιοι στο πάτημα του κουμπιού, αυτό που θέλουμε είναι να εκτελεστεί κάποιος κώδικα όταν γίνει κάποιο action στο button με ετικέτα ***Test Passwords***

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.

Why an engineer needs an IDE…



Οι γλώσσες προγραμματισμού πλέον αποτελούνται από χιλιάδες ενσωματωμένες μεθόδους τις οποίες ο Μηχανικός Λογισμικού πρέπει να χρησιμοποιήσει για την κατασκευή του έργου του. Ένα IDE παρέχει πρόσβαση σε όλους αυτούς τους μηχανισμούς, ακόμα και όταν ο Μηχανικός δεν τους γνωρίζει.

Σημ.: Εάν το Netbeans δεν εμφανίσει αυτόματα κάποιο **Auto Complete**, αρκεί να πατήσετε **Ctrl+Space**. Το Auto Complete αναφέρεται σε οτιδήποτε εκείνη τη στιγμή μπορείτε να χρησιμοποιήσετε.

Java Database Connectivity (JDBC)

Java Database Connectivity (JDBC)

- Διεπαφή της Java που επιτρέπει την **επικοινωνία** μίας Java εφαρμογής με μία Βάση Δεδομένων (ΒΔ).
- Για κάθε Σύστημα Διαχείρισης Βάσεων Δεδομένων (πχ. MySQL, Microsoft SQL Server, MariaDB, PostgreSQL κ.α.) υπάρχει ένα πρόγραμμα οδήγησης (**driver**) το οποίο επιτρέπει τον χειρισμό της ΒΔ με το **JDBC API**.
- Μία τυπική χρήση του JDBC API είναι:
  - **Εγκαθίδρυση σύνδεσης στην ΒΔ**
  - **Ορισμός και εκτέλεση SQL ερωτημάτων**
  - **Επεξεργασία αποτελεσμάτων**
  - **Τερματισμός σύνδεσης με την ΒΔ**

Σύνδεση στην Βάση Δεδομένων (1)

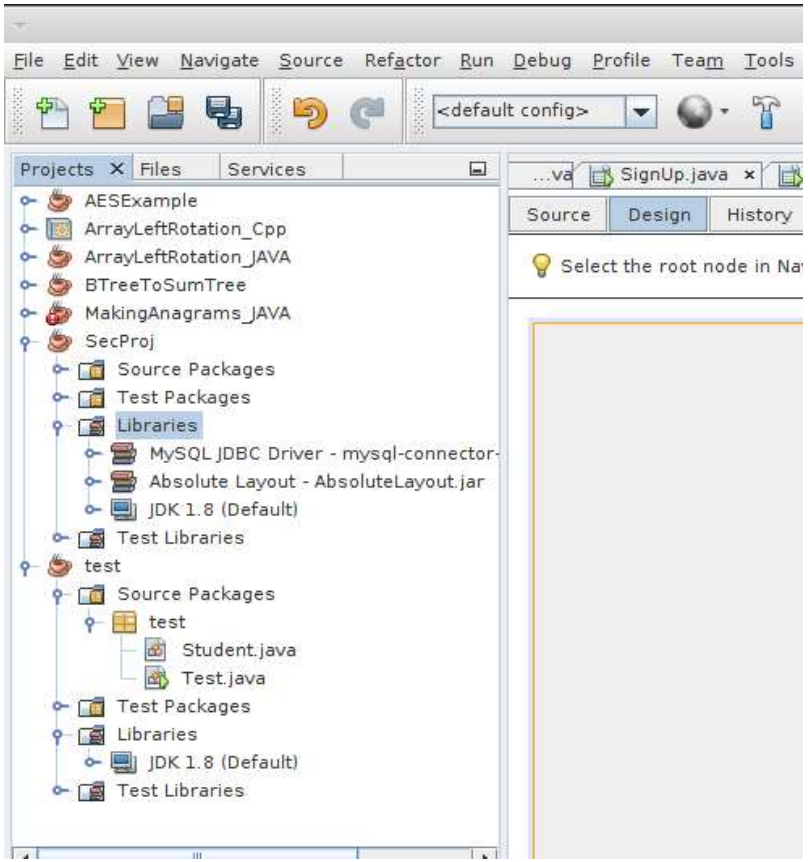
- Ορίζουμε ένα αντικείμενο τύπου **Connection**
  - Διαχειρίζεται την σύνδεση ανάμεσα στην Java εφαρμογή και την ΒΔ
  - Επιτρέπει την δημιουργία SQL προτάσεων (statements) από την εφαρμογή Java στην ΒΔ

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

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

MySQL: “jdbc:mysql://hostname:portNumber/databaseName”
MariaDB: “jdbc:mariadb://hostname:portNumber/databaseName”

Σύνδεση στην Βάση Δεδομένων (2)



- **Προσοχή**: Θα πρέπει να κατεβάσετε τον JDBC driver για την αντίστοιχη τεχνολογία ΒΔ (π.χ. MySQL) που θα χρησιμοποιήσετε και να τον προσθέσετε στα libraries 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

```
ResultSet resultSet = statement.executeQuery("SHOW TABLES");
```

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();
```

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();
}
```

Οι αριθμοί των στηλών ξεκινούν από το 1!

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

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://www.ntu.edu.sg/home/ehchua/programming/howto/NetBeans_HowTo.html

https://docs.oracle.com/javase/tutorial/uiswing/learn/index.html

https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

"Java Προγραμματισμός", Deitel P. και Deitel H., 10η έκδοση, 2015.