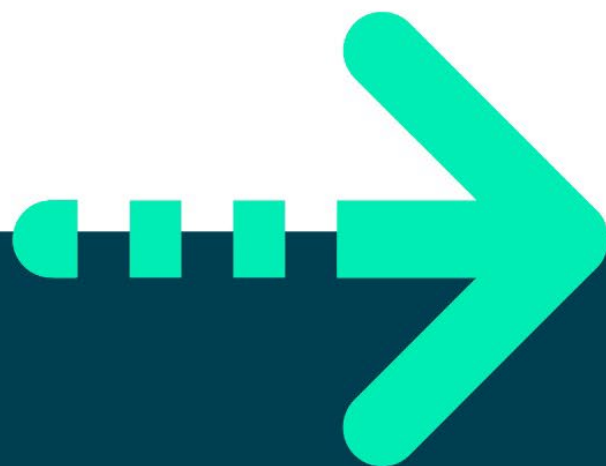




ACCESSING DATABASES USING JAVA





Set up SQL server for JDBC

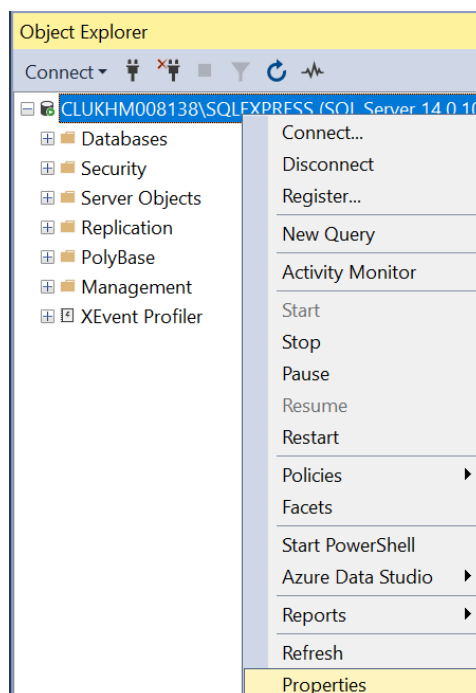
In the following lab you'll practise creating a new user for MS-SQL using SSMS. You will then set the right this new should have and then use the username and password in code to connect and extract data from a database using Java.

Create the database needed for this lab

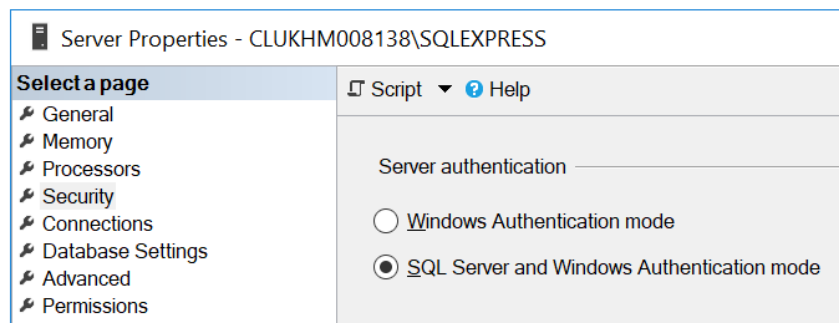
- In this lab you will require a database called QAStore.
- To install this database, double-click on the file **QAStore.sql** in the Resources folder.
- **Microsoft Server Management Studio** (SSMS) app will open.
- Log in as the default Windows user (you!)
- Select **Execute** (or press F5) to install the QAStore database.

Prepare and configure MS-SQL

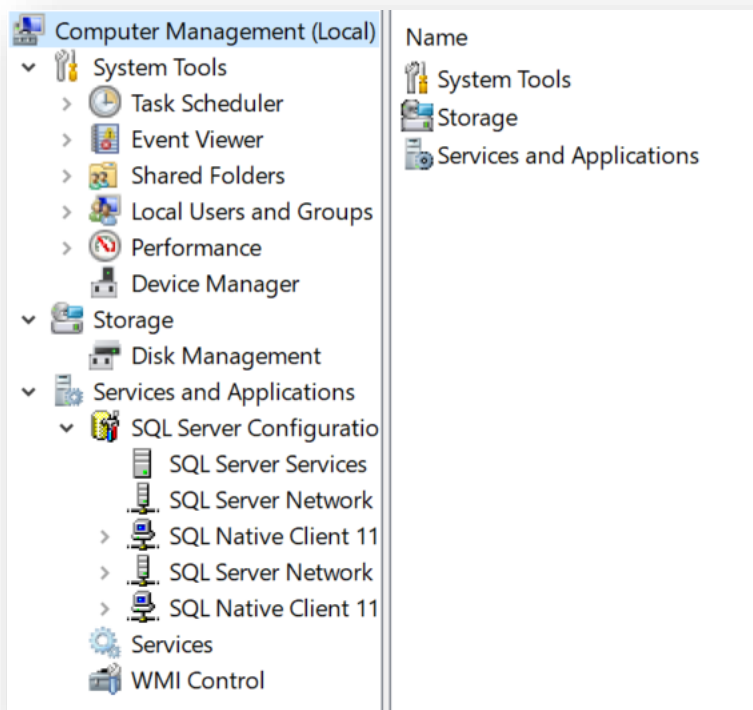
- Start SSMS (if not started already) and then right-click on the server.
- Select **Properties**.



- Choose **Security**.
- Select **SQL Server and Windows Authentication mode**.
- Select **OK**.
- Close SSMS because you're going to make more changes.

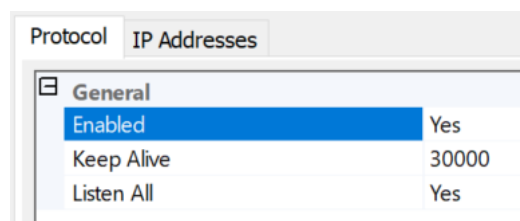
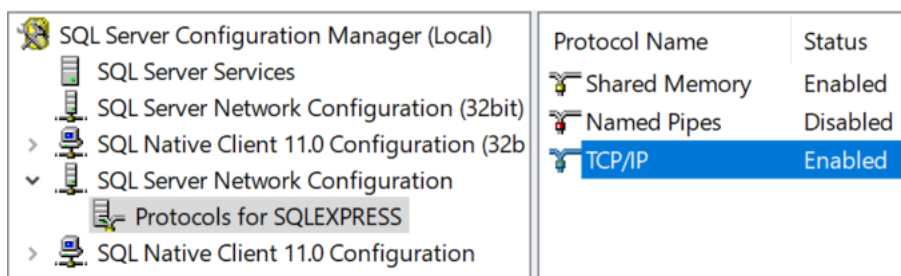


- In Windows, type the word **compmgmt.msc** in the search box. This will run the Computer management application.
- Select **Services and Applications > SQL Sever Network Configuration**.

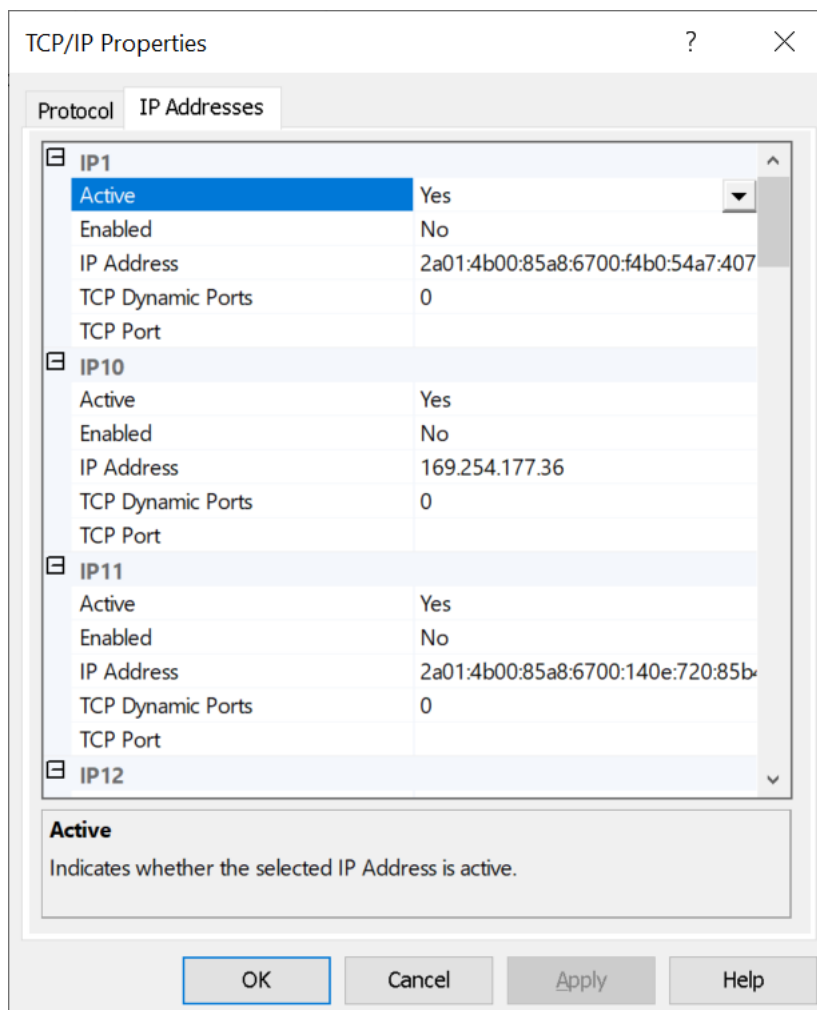


- Double-click the **SQL Server Network** menu from the list on the left.
- Double click the Protocols for SQLEXPRESS (on the right hand side).
- Double-click the **TCP/IP** menu from the right-hand side.

- Double-click on the Enabled attribute to change its state to Yes (Enabled).



- Select the **IP Addresses** tab.

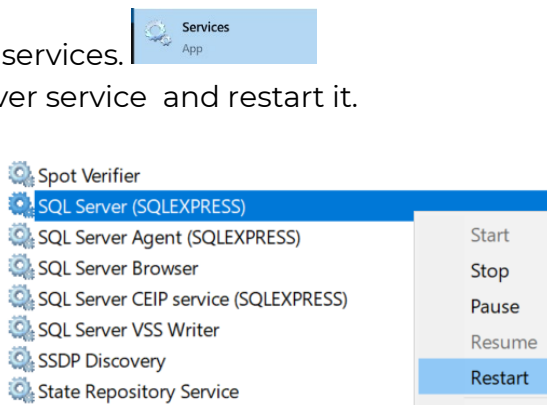




- Scroll to the bottom of the list and take a note of the **IPAll** TCP port **1433 (could be different)** which you'll need when you construct a connection string in your Java code later on.

Restart the SQL service

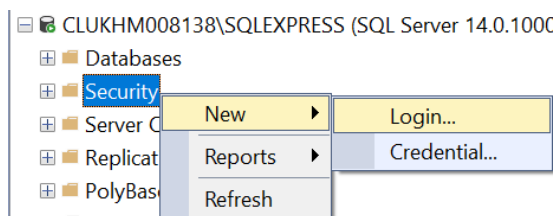
- In windows, type services.
- Find the SQL Server service and restart it.



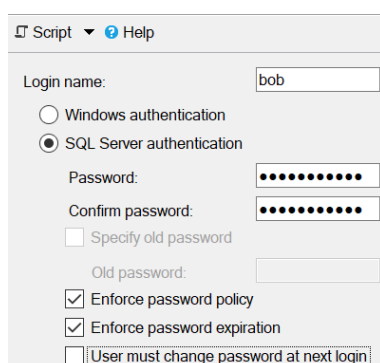
Create a user in SSMS

Next we'll create a user for your application with enough rights to access a database of your choice.

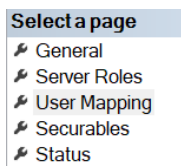
- Create a new login by right-clicking on the Security tab.



- Type the user details and uncheck **User must change password at next login**.



- Grant rights by selecting the **User Mapping** item on the left.



- Select the **qastore** database and assign as many rights as needed. Here we've selected **db_owner** rights but in real life you may select fewer rights.

Users mapped to this login:

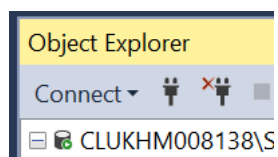
Map	Database	User
<input type="checkbox"/>	AdventureWorks2008R2	
<input checked="" type="checkbox"/>	qastore	bob
<input type="checkbox"/>	QATSQLPLUS	
<input type="checkbox"/>	temp	
<input type="checkbox"/>	tempdb	

☐ Guest account enabled for: qastore

Database role membership for: qastore

- ☐ db_accessadmin
- ☐ db_backupoperator
- ☐ db_datareader
- ☐ db_datawriter
- ☐ db_ddladmin
- ☐ db_denydatareader
- ☐ db_denydatawriter
- ☒ db_owner
- ☐ db_securityadmin
- ☒ public

- Choose **OK**.
- Now you need to disconnect and reconnect the database.



- Try the new user when you connect back again to test the new login.
- You're now ready to use this new user's credentials in code.

Write the code

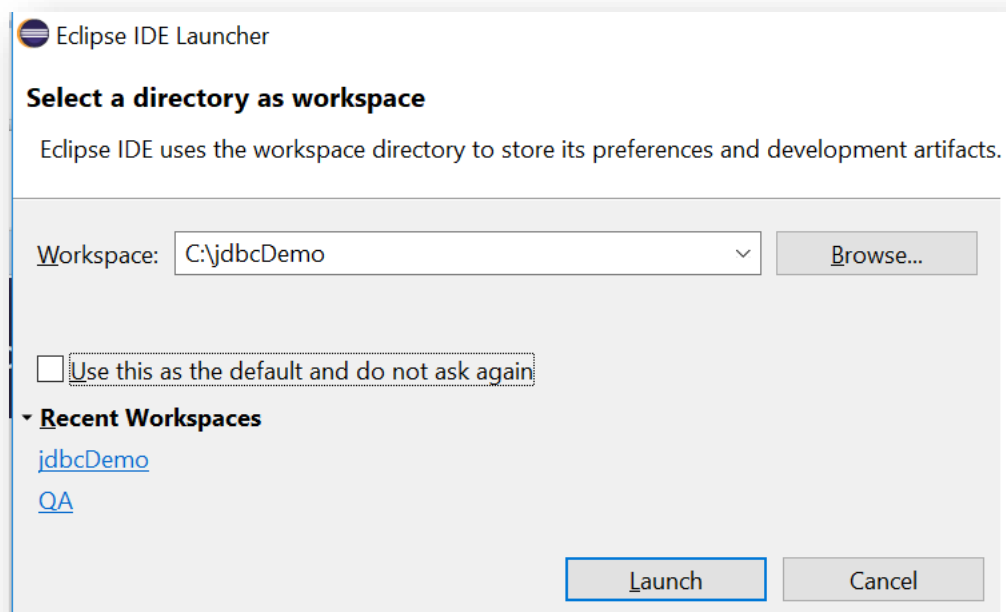
In this section you'll extract rows from the **qastore** database:

- Before you start coding you'll need the JDBC library which you can download by opening this link:
<https://repo1.maven.org/maven2/com/microsoft/sqlserver/mssql-jdbc/8.3.1.jre14-preview/>

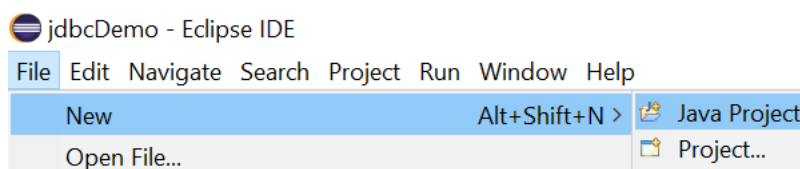
You'll see all the supported database types on this page.

To save you downloading a driver we've included the **mssql-jdbc-8.3.1.jre14-preview.jar** file in your course material folder.

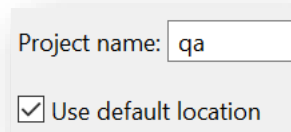
- You'll now need to create an application and use this jar file:
 - Start Eclipse.
 - Select a folder where you want to save your project.



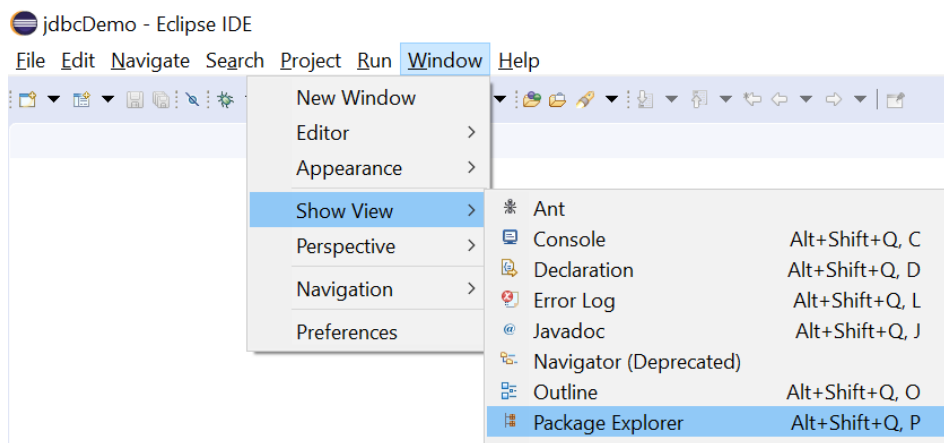
- Select **File > New > Java Project**.



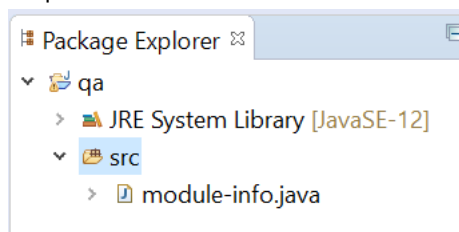
- Type a project name and choose **Finished**.



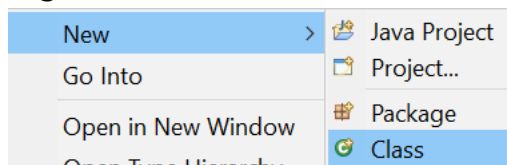
- Select the **Create** button in the next dialog.
- You can close the Welcome page.
- Select the package explorer button (small icon on top left). Alternatively navigate to the following menus:



- Expand the nodes.



- Right-click on the src node and then select the **New > Class** menus.



- Select name Program (or any name you like, first letter in caps) and make sure you select the **public static void main(String[] args)** check box.

Java Class

Create a new Java class.



Source folder:

Package:

☐ Enclosing type:

Name:

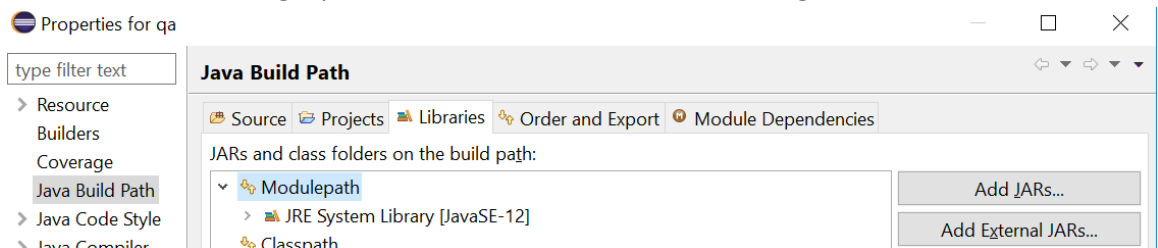
Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?
☒ public static void main(String[] args)

- Choose **Finished**.
- Right-click on your project (here it's **qa**) and then select **Properties**.
- Select the following options and tabs in the next dialog:



- Choose **Add External JARs...**
- Find the jar file provided (**mssql-jdbc-7.1.2.jre8-preview.jar**) and then select **Apply and Close**.
- Back in the **src** node, select the Program class to display its code.



- Replace the main() method's code with

```
public static void main(String[] args) throws Exception{

    String connectionUrl = "jdbc:sqlserver://localhost\\SQLEXPRESS:1433"
        + ";databaseName=qastore;user=bob;password=password123";

    Connection connection = DriverManager.getConnection(connectionUrl);

    String sql = "SELECT company_name FROM company";

    Statement statement = connection.createStatement() ;
    ResultSet resultSet = statement.executeQuery(sql);

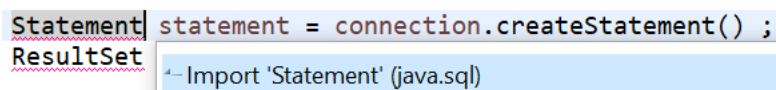
    while ( resultSet.next() ){
        System.out.println( resultSet.getString(1));
    }

    connection.close();
}
```

- The code sample seems to have failed with errors! To get rid of these select the class name **Connection** in code and then press Ctrl-I (control I).

You can now double-click the **Import 'Connection' (java.sql)...** option to resolve the Connection error.

Press Ctrl-I on other errors and then select the **Import** option.



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

Import 'Statement' (java.sql)

- After all the errors are resolved, save your code by pressing Ctrl-s.
- Run your code to see it working by pressing Ctrl-F11.



Create parameterised queries (optional)

- Create another class called Program2.
Right-click on the **src** node and then select **New > Class**.
- Copy the following code to the class.
This is a complex piece of code. Please study the code and see if it makes sense.

```
private static void example2(Connection connection) throws Exception
{
    String sql = "SELECT * FROM company WHERE county = 'London' " ;

    Statement statement = connection.createStatement() ;
    ResultSet resultSet = statement.executeQuery(sql);
    resultSet.setFetchSize(2); // optional

    String id, name, county;
    StringBuilder out = new StringBuilder() ;
    while ( resultSet.next() ){
        id = resultSet.getString(1);
        name = resultSet.getString(2) ;
        county = resultSet.getString("county") ;
        out.append(String.format("Id: %-7s Name:%-25s County:%s\n",
            id,name,county) );
    }
    connection.close();
    System.out.println(out.toString());
}
```

Create Parameterised queries

To protect your system against SQL Injection attacks, and for better constructed SQL statements, always use parameterised queries. In the above example the word 'London' was inserted into a SQL string. Please copy and run the following code. You'll find this method a lot easier when you've multiple parameters.



```
private static void example3(Connection connection) throws Exception {
    String sql = "select company_no, company_name, post_code, county "
        + "from company WHERE county=? AND post_code like ?" ;
    PreparedStatement statement = connection.prepareStatement(sql) ;

    // 'London or 1=1; --' will not return any result!
    statement.setString(1,"London"); // the first ?      (city)
    statement.setString(2,"N%");// the second ? (contactName)

    ResultSet resultSet = statement.executeQuery();

    String company_no, company_name, post_code, county;
    StringBuilder out = new StringBuilder() ;

    while ( resultSet.next() ){
        company_no = resultSet.getString(1);
        company_name = resultSet.getString(2) ;
        post_code = resultSet.getString(3) ;
        county = resultSet.getString("county") ;

        out.append(String.format("No: %-7s Name:%-25s County:%s Post_Code:%s\n", company_no, company_name, county, post_code) );
    }
    System.out.println(out.toString());
}
```

Now try inserting a new row into the **customers** table.

Hint: use the statement's `executeUpdate()` method

Try to update the row which you just created.

See if you can delete a row.

