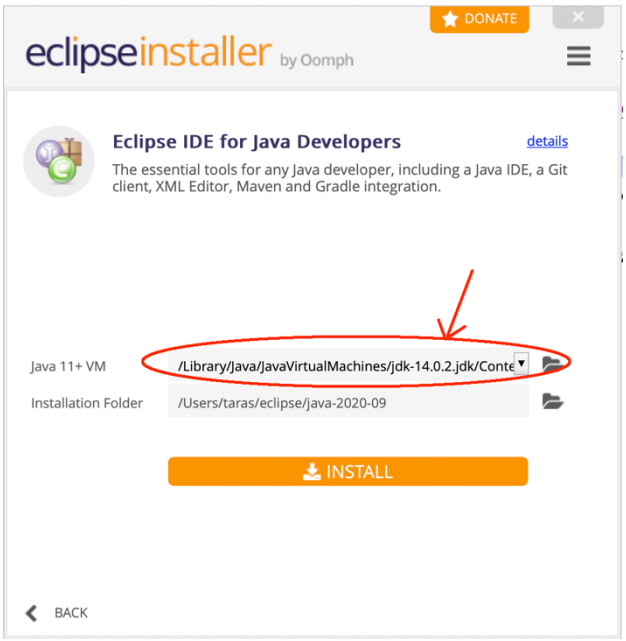
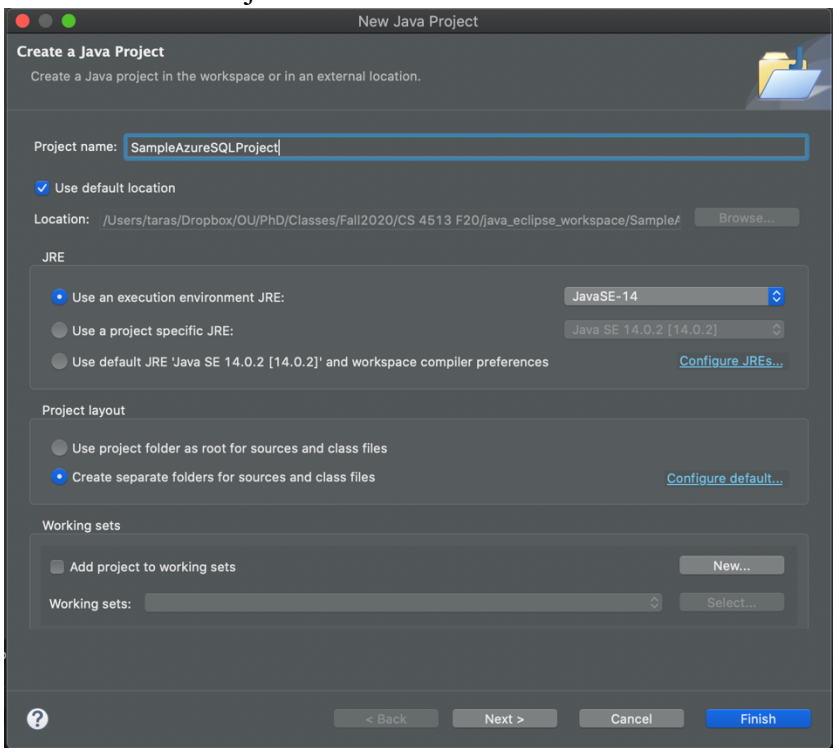
CS/DSA 4513 Fall 2020
Using Java and JDBC to Connect to the Azure SQL Database

1. **Download & Install Java SDK v14**
    - Download JDK v14 from the below link
        - https://www.oracle.com/java/technologies/javase-jdk14-downloads.html
    - Follow the installation instructions at the below link
        - https://docs.oracle.com/en/java/javase/14/install/overview-jdk-installation.html
2. **Download & Install Eclipse IDE (2020-09 R) for Java Developers**
    - Please follow the instructions at the below link (also see the note below)
        - http://www.eclipse.org/downloads/packages/installer
    - Note, if you asked about Java VM, make sure to select the version 14 which you just installed (see the screenshot below)

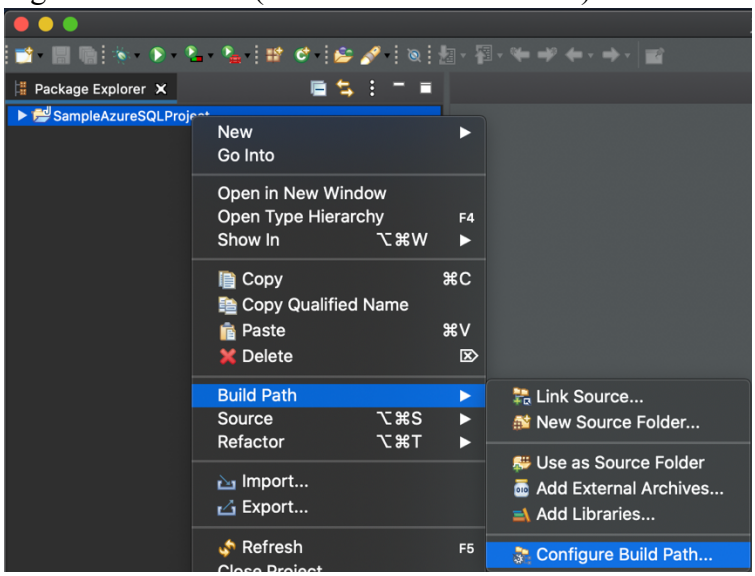


3. **Download & Install Microsoft JDBC Driver 8.4**
    - Download Microsoft JDBC Driver 8.4 for SQL Server from the below link
        - https://docs.microsoft.com/en-us/sql/connect/jdbc/download-microsoft-jdbc-driver-for-sql-server?view=sql-server-ver15
    - Follow the Install Instructions on the same page. Make sure to note your installation directory.
4. **Connect to Azure SQL DB using Java**
    - Launch Eclipse IDE you just installed
        - We recommend creating a new workspace directory for this class when prompted
    - Close the "Welcome" window if you see it.
    - Click on "File" > "New" > "Java Project"
    - When presented with the "New Java Project" Window
        - Give the project name ("*SampleAzureSQLProject*" for example)
        - Select "Use an execution environment JRE" and "JavaSE-14"

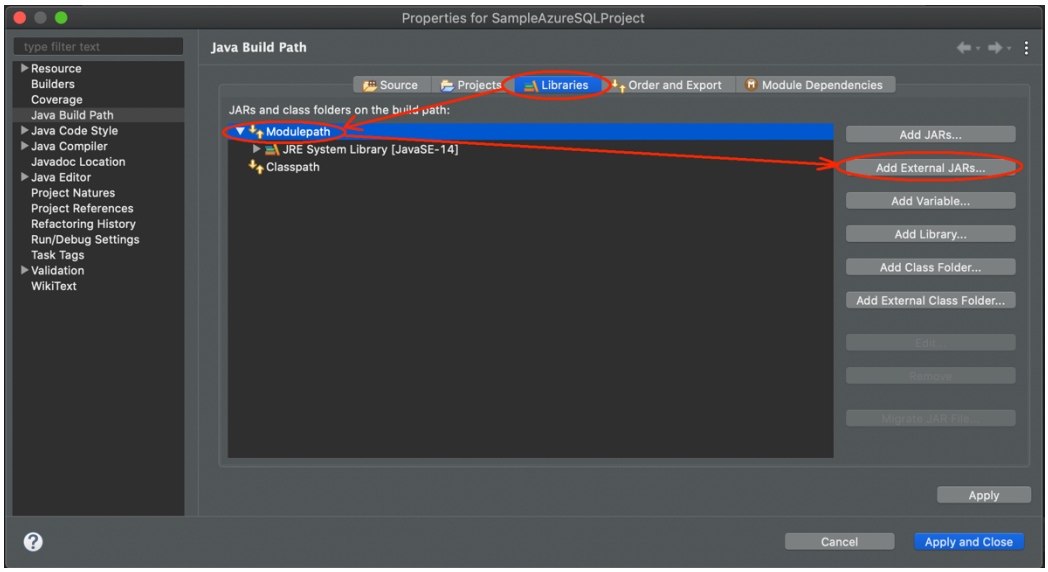
- Your window should look like the screenshot below
- Click on “Finish”. Choose “Don’t create” when asked about creating module-info.java



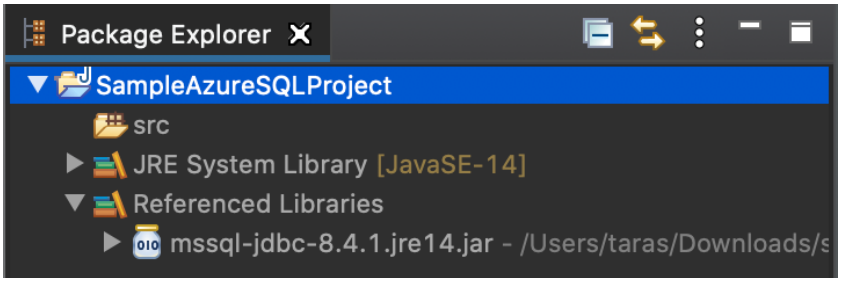
- Now, right-click on the project “SampleAzureSQLProject” > “Build Path” > “Configure Build Path” (see the screenshot below)



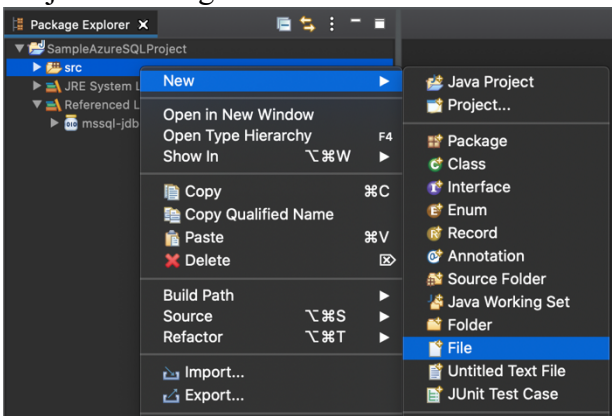
- Click on “Libraries tab”, click on “Modulepath” and “Add External JARs…” and select the “mssql-jdbc-8.4.1.jre14.jar” file available in the installation directory of the JDBC driver (we asked you to note it earlier).



- Click on “Apply and Close”.
- You should now see that JAR file added into referenced libraries of your project.

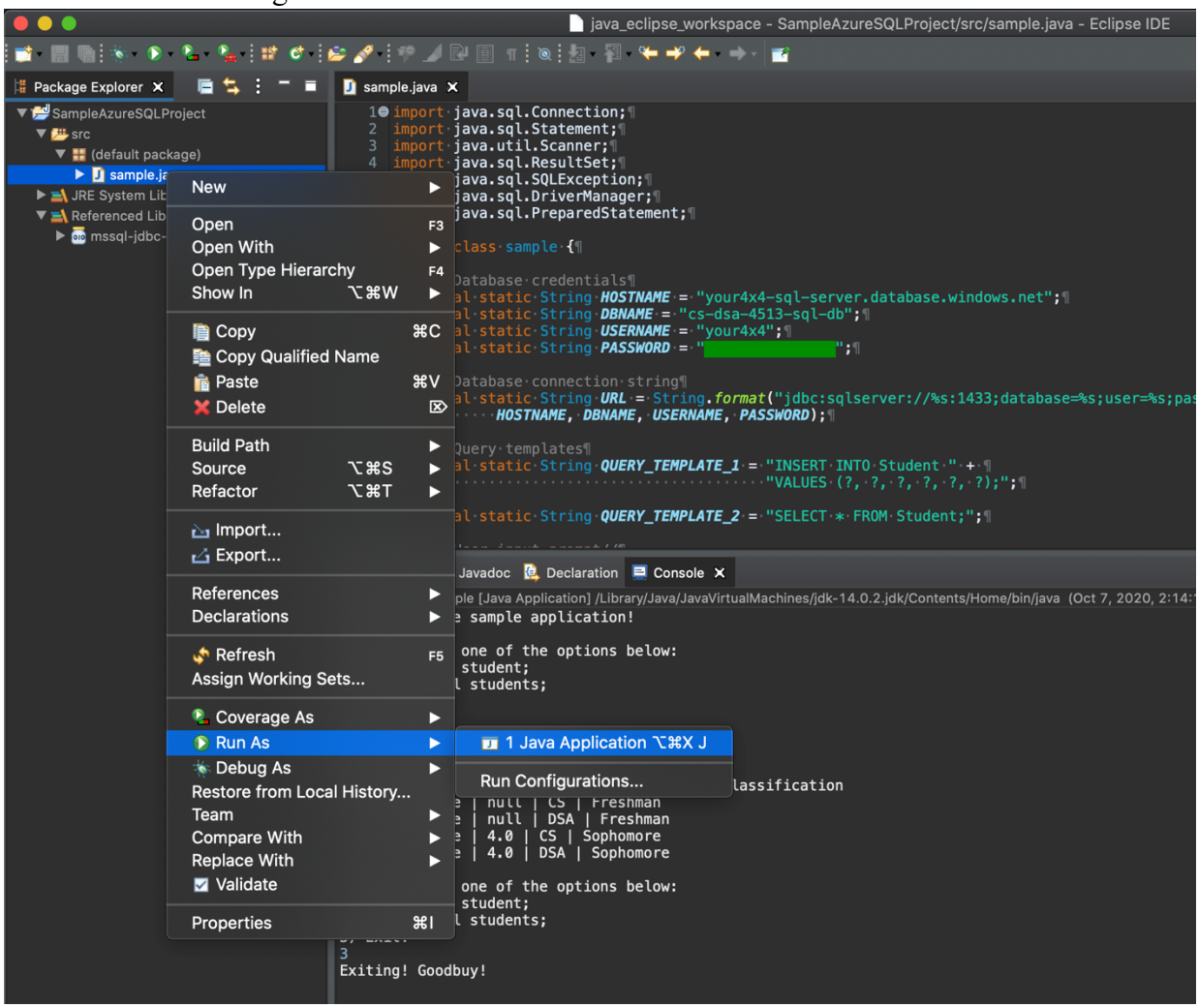


- Create a new java file. Right click on “src” folder > “New” > “File”



- Name it as “*sample.java*” and click “Finish” button.
- Once the new file is opened in a tab copy the code from Appendix A (also available on Canvas) below into it.
- Replace the values “<your4x4>” and “<your password>” in the java program with your SQL server admin username and password. Make sure the hostName and dbName values match your Azure SQL server and database names.
    - FYI: The connection strings (url variable in the sample code) can also be taken from your Azure Dashboard > “cs-dsa-4513-sql-db” > “connection strings” > “JDBC”

- Note, the following step assumes the SQL statements from the Appendix B (also available on Canvas) below were previously executed in your SQL database. If it's not the case, either execute them now, or modify the queries you want your java program to execute.
- Run your java program by right-clicking on sample.java > "Run As" > "Java Application". This application is capable of printing out the Student table and inserting new students into it.



- You were able to successful connect to your Azure SQL database and should now be ready to work on your Graded Homework #3, good luck!

Helpful links:

1. Sample JDBC driver applications
   a. https://docs.microsoft.com/en-us/sql/connect/jdbc/sample-jdbc-driver-applications?view=azuresqldb-current

Appendix A. sample.java program (also available on Canvas)

```
import java.sql.Connection;
import java.sql.Statement;
import java.util.Scanner;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.DriverManager;
import java.sql.PreparedStatement;

public class sample {

    // Database credentials
    final static String HOSTNAME = "<your4x4>-sql-server.database.windows.net";
    final static String DBNAME = "cs-dsa-4513-sql-db";
    final static String USERNAME = "<your4x4>";
    final static String PASSWORD = "<your_password>";

    // Database connection string
    final static String URL =
String.format("jdbc:sqlserver://%s:1433;database=%s;user=%s;password=%s;encrypt=true;trustServerCertificate=false;hostNameIn
Certificate=*.database.windows.net;loginTimeout=30;",
            HOSTNAME, DBNAME, USERNAME, PASSWORD);

    // Query templates
    final static String QUERY_TEMPLATE_1 = "INSERT INTO Student " +
                                    "VALUES (?, ?, ?, ?, ?, ?);";

    final static String QUERY_TEMPLATE_2 = "SELECT * FROM Student;";

    // User input prompt//
    final static String PROMPT =
            "\nPlease select one of the options below: \n" +
            "1) Insert new student; \n" +
            "2) Display all students; \n" +
            "3) Exit!";

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

        System.out.println("Welcome to the sample application!");

        final Scanner sc = new Scanner(System.in); // Scanner is used to collect the user input
        String option = ""; // Initialize user option selection as nothing
        while (!option.equals("3")) { // As user for options until option 3 is selected
            System.out.println(PROMPT); // Print the available options
            option = sc.next(); // Read in the user option selection

            switch (option) { // Switch between different options
                case "1": // Insert a new student option
                    // Collect the new student data from the user
                    System.out.println("Please enter integer student ID:");
                    final int id = sc.nextInt(); // Read in the user input of student ID

                    System.out.println("Please enter student first name:");
                    // Preceding nextInt, nextFloar, etc. do not consume new line characters from the user input.
                    // We call nextLine to consume that newline character, so that subsequent nextLine doesn't return nothing.
                    sc.nextLine();
                    final String fname = sc.nextLine(); // Read in user input of student First Name (white-spaces allowed).

                    System.out.println("Please enter student last name:");
                    // No need to call nextLine extra time here, because the preceding nextLine consumed the newline character.
                    final String lname = sc.nextLine(); // Read in user input of student Last Name (white-spaces allowed).

                    System.out.println("Please enter float student GPA:");
                    final float gpa = sc.nextFloat(); // Read in user input of student GPA

                    System.out.println("Please enter student major:");
                    sc.nextLine(); // Consuming the trailing new line character left after nextFloat
                    final String major = sc.nextLine(); // Read in user input of student Major
```

```
            System.out.println("Please enter student classification (Freshman, Sophomore, Junior, or Senior):");
            final String classification = sc.nextLine(); // Read in user input of student Classification

            System.out.println("Connecting to the database...");
            // Get a database connection and prepare a query statement
            try (final Connection connection = DriverManager.getConnection(URL)) {
               try (
                  final PreparedStatement statement = connection.prepareStatement(QUERY_TEMPLATE_1)) {
                  // Populate the query template with the data collected from the user
                  statement.setInt(1, id);
                  statement.setString(2, fname);
                  statement.setString(3, lname);
                  statement.setFloat(4, gpa);
                  statement.setString(5, major);
                  statement.setString(6, classification);

                  System.out.println("Dispatching the query...");
                  // Actually execute the populated query
                  final int rows_inserted = statement.executeUpdate();
                  System.out.println(String.format("Done. %d rows inserted.", rows_inserted));
               }
            }

            break;
         case "2":
            System.out.println("Connecting to the database...");
            // Get the database connection, create statement and execute it right away, as no user input need be collected
            try (final Connection connection = DriverManager.getConnection(URL)) {
               System.out.println("Dispatching the query...");
               try (
                  final Statement statement = connection.createStatement();
                  final ResultSet resultSet = statement.executeQuery(QUERY_TEMPLATE_2)) {

                     System.out.println("Contents of the Student table:");
                     System.out.println("ID | first name | last name | GPA | major | classification ");

                     // Unpack the tuples returned by the database and print them out to the user
                     while (resultSet.next()) {
                        System.out.println(String.format("%s | %s | %s | %s | %s | %s ",
                           resultSet.getString(1),
                           resultSet.getString(2),
                           resultSet.getString(3),
                           resultSet.getString(4),
                           resultSet.getString(5),
                           resultSet.getString(6)));
                     }
               }
            }

            break;
         case "3": // Do nothing, the while loop will terminate upon the next iteration
            System.out.println("Exiting! Goodbuy!");
            break;
         default: // Unrecognized option, re-prompt the user for the correct one
            System.out.println(String.format(
               "Unrecognized option: %s\n" +
               "Please try again!",
               option));
            break;
      }
   }

   sc.close(); // Close the scanner before exiting the application
  }
}
```

Appendix B. sample.sql statements (also available on Canvas)

```
-- While working on the database design, it's usefull to start from scratch every time
-- Hence, we drop tables in reverse order they are created (so the foreign keys not
violated)
DROP TABLE IF EXISTS Enrollment;
DROP TABLE IF EXISTS Class;
DROP TABLE IF EXISTS Student;

-- Create tables
CREATE TABLE Student (
    id INT PRIMARY KEY,
    first_name VARCHAR(64) NOT NULL,
    last_name VARCHAR(64) NOT NULL,
    gpa real,
    major VARCHAR(16) NOT NULL,
    classification VARCHAR(10) NOT NULL,
    CONSTRAINT CHK_gpa CHECK (gpa BETWEEN 0.0 AND 4.0),
    CONSTRAINT CHK_classification CHECK (classification IN ('Freshman', 'Sophomore',
'Junior', 'Senior'))
);

CREATE TABLE Class (
    code VARCHAR(10) PRIMARY KEY,
    name VARCHAR(64),
    description VARCHAR(1024)
);

CREATE TABLE Enrollment (
    student_id INT,
    class_code VARCHAR(10),
    semester VARCHAR(6) NOT NULL,
    year INT NOT NULL,
    grade CHAR(1),
    CONSTRAINT PK_enrollment PRIMARY KEY (student_id, class_code, semester, year),
    CONSTRAINT FK_student_id FOREIGN KEY (student_id) REFERENCES Student,
    CONSTRAINT FK_class_code FOREIGN KEY (class_code) REFERENCES Class,
    CONSTRAINT CHK_semester CHECK (semester IN ('Spring', 'Fall')),
    CONSTRAINT CHK_grade CHECK (grade IN ('A', 'B', 'C', 'D', 'F'))
);

-- Insert some valid records

-- Students without GPA
INSERT INTO Student
    (id, first_name, last_name, major, classification)
VALUES
```

```
    (1, 'John', 'Doe', 'CS', 'Freshman'),
    (2, 'Jane', 'Doe', 'DSA', 'Freshman');

-- Students with GPA
INSERT INTO Student
VALUES
    (3, 'Jack', 'Doe', 4.0, 'CS', 'Sophomore'),
    (4, 'Jill', 'Doe', 4.0, 'DSA', 'Sophomore');

-- Classes
INSERT INTO Class
VALUES
    ('CS4513', 'Database Management Systems', 'Just read the sylabus.'),
    ('CS5513', 'Advanced Database Management Systems', NULL);

-- Enrollments without grades
INSERT INTO Enrollment
    (student_id, class_code, semester, year)
VALUES
    (1, 'CS4513', 'Fall', 2020),
    (2, 'CS4513', 'Fall', 2020);

-- Enrollments with grades
INSERT INTO Enrollment
VALUES
    (3, 'CS5513', 'Spring', 2020, 'A'),
    (4, 'CS5513', 'Spring', 2020, 'A');
```