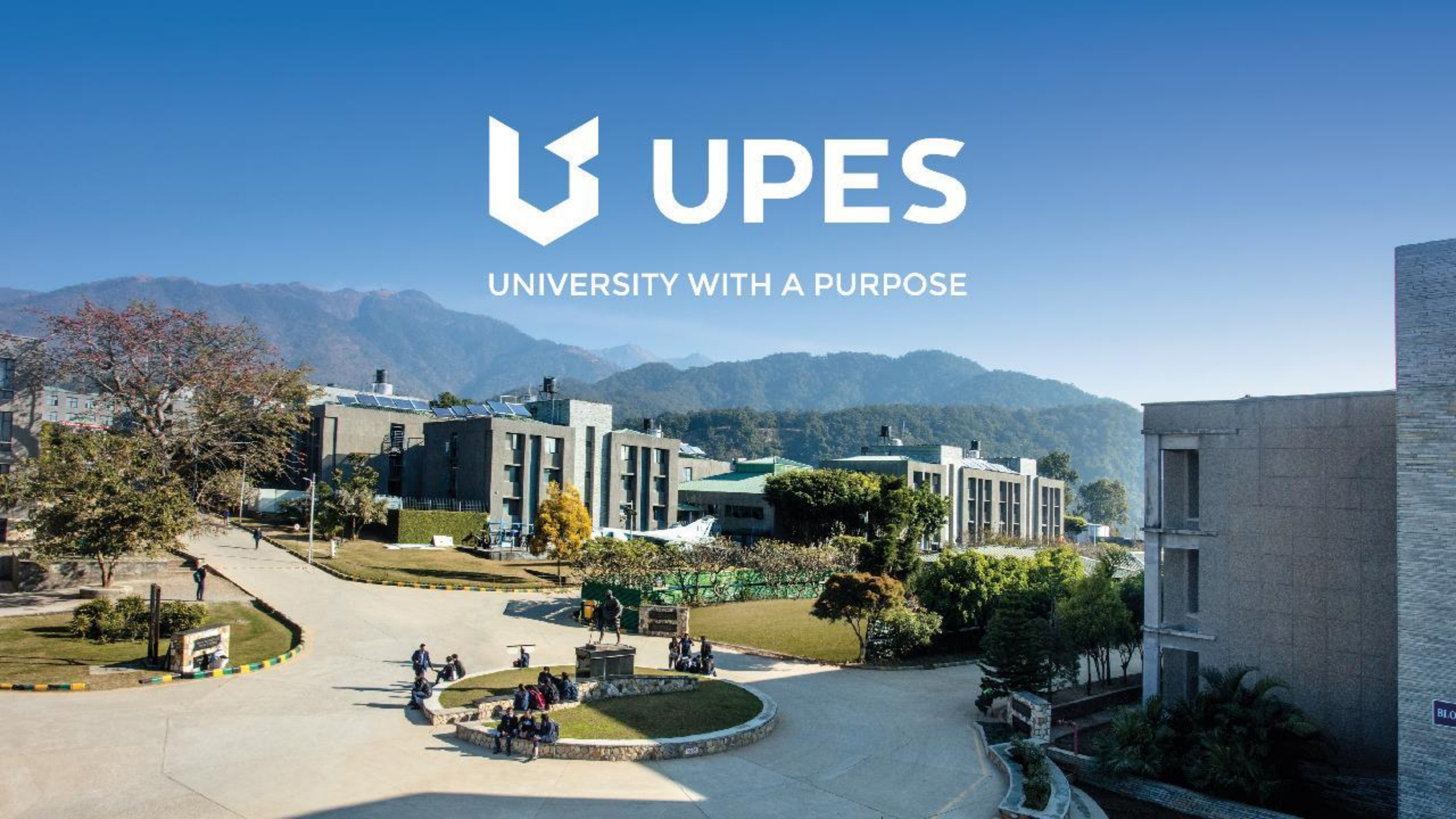

UPES
UNIVERSITY WITH A PURPOSE

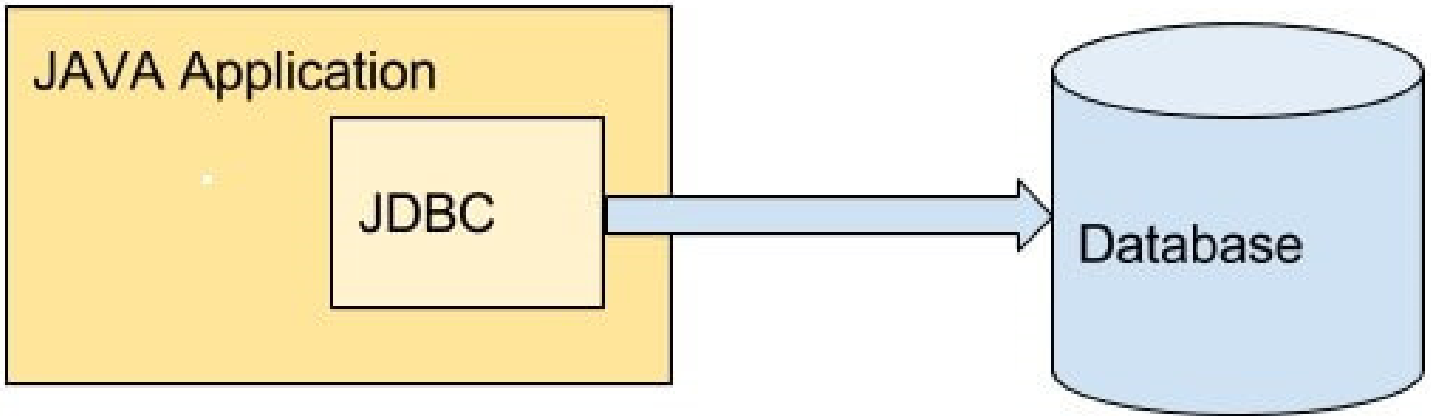
Introduction to JDBC

Introduction to JDBC

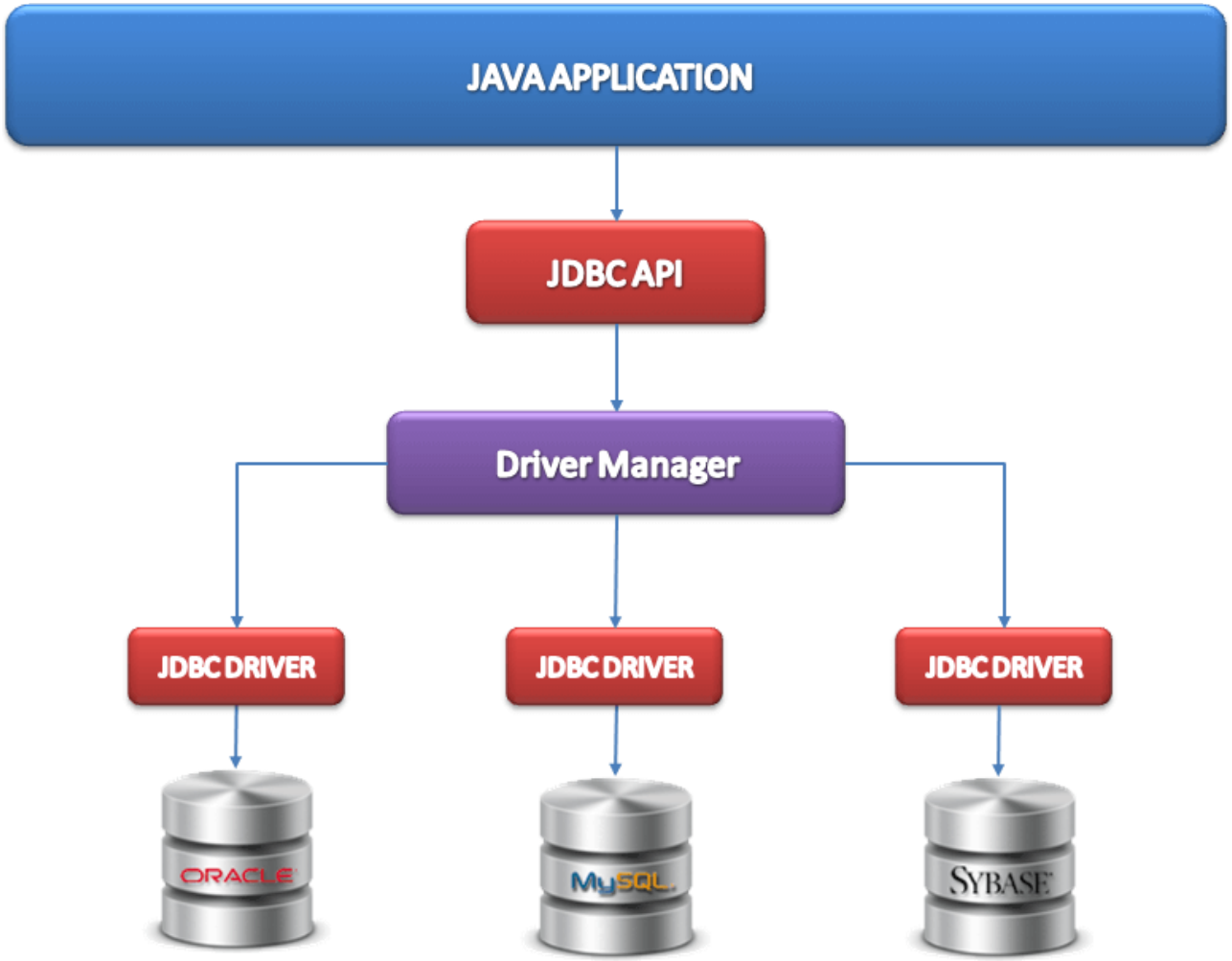
- JDBC: Java DataBase Connectivity
- JDBC API defines interfaces and classes for writing database applications in Java by making database connections.
- JDBC API is a Java API that can access any kind of tabular data, especially data stored in a Relational Database.
- The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage:

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

Introduction to JDBC



Introduction to JDBC



JDBC Components: Database Driver

- A JDBC driver is a software component enabling a Java application to interact with a database.
- Driver handles the communications with the database server.
- However, we interact directly with Driver objects very rarely. Instead, we use DriverManager objects, which manages objects of this type.
- It also abstracts the details associated with working with Driver objects

JDBC Components: DriverManager

- DriverManager manages a list of database drivers.
- It matches connection requests from Java applications with the proper database driver using communication protocols.
- The first driver that recognizes a certain protocol under JDBC will be used to establish a database connection.

JDBC Components: Connection

- Connection interface has all the methods for making connection with database.
- The connection object represents communication context, i.e. all communication with database is through connection object only.

JDBC Components: CreateStatement

- createStatement is an interface for representing SQL statements.
- A SQL statement is precompiled and stored in a createStatement object. This object can then be used to efficiently execute this statement multiple times.
- **Exp:**

```
className ps = con.createStatement ();
ps. methodName();
```

JDBC Components: ResultSet

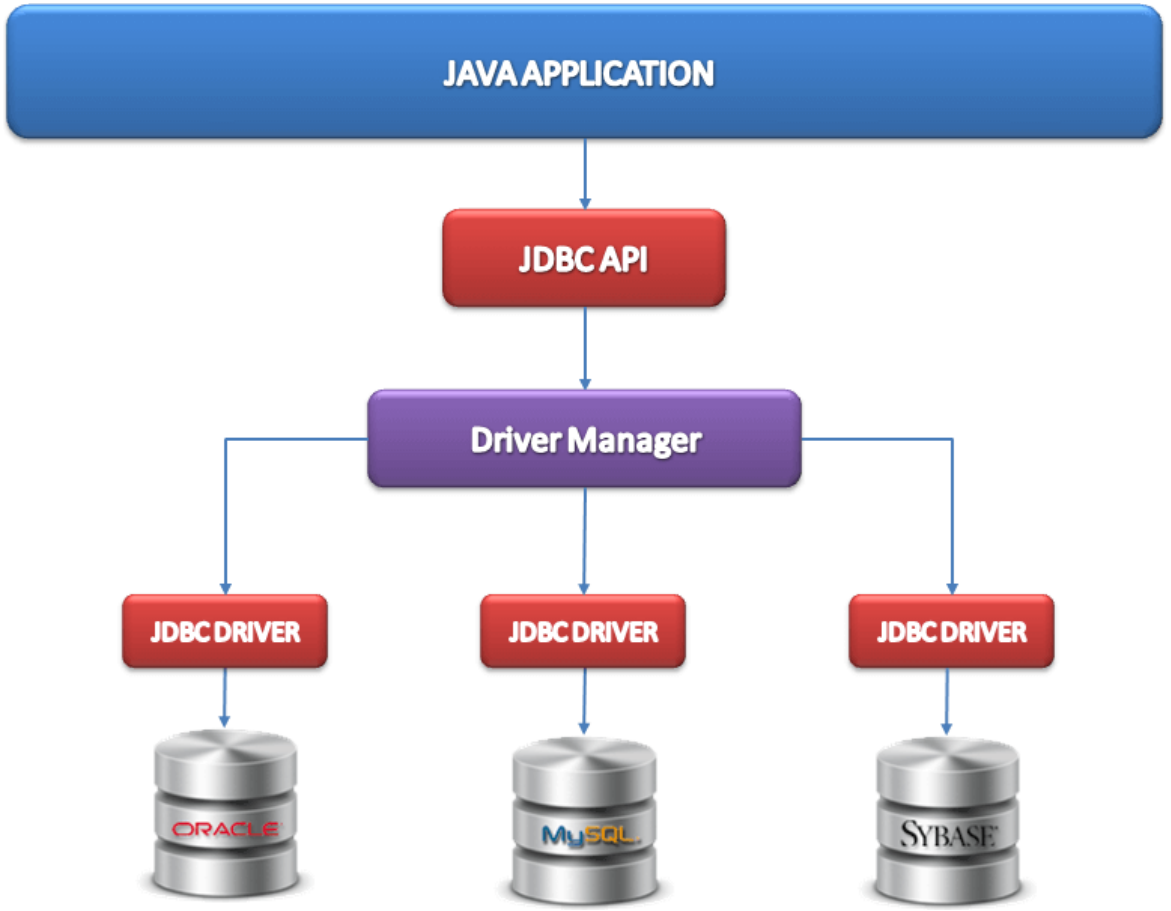
- ResultSet is an interface to represent a database table.
- These objects hold data retrieved from a database after you execute an SQL query using Statement objects.
- It acts as an iterator to allow you to move through its data.

JDBC Components: SQLException

- SQLException is a subclass of Exception class in Java.
- It is an exception that provides information on a database access error or other errors.

Steps for Connections

- Loading the Drivers
- Establishing Connection
- Creating Statements
- Executing Statements
- Getting Results
- Closing database connection



Downloads & Setup

1) Download and install MYSQL from https://dev.mysql.com/downloads/installer/

2) Download MySQL Connector from https://dev.mysql.com/downloads/connector/j/ (Platform Independent)

Unzip the file. Executable JAR file is the connector file.

3) RightClick on the Project in Eclipse -> BuildPath -> Add External Archives

Select the Executable JAR file

Steps for Connections contd (MySQL)..

Loading the Drivers:

```
Class.forName("com.mysql.cj.jdbc.Driver");
```

Establishing Connection:

When getConnection is called the DriverManager will attempt to locate a suitable driver from the loaded.

```
DriverManager.getConnection(url, user, password)
```

```
url= "jdbc:mysql://localhost:3306/db1",
```

getConnection: method returns Connection Object on success otherwise Null

Create Statement: Connection object has a method to create an SQL statement.

```
Statement stmt=con.createStatement();
```

Steps for Connections contd..

Executing Statement:

```
Execute()
executeQuery(): Returns object of resultSet
executeUpdate()
```

Getting ResultSet: ResultSet object can be obtained as a returned object by executeQuery() method of createStatement.

- ResultSet object when not null or not empty, can iterate over that and get results.

Closing Database Connection: Connection object has a method close() which is used to close the connection.

```
con.close()
```

THANK YOU