CSC343: Introduction to Databases

Winter 2019

JDBC

DriverManager class

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

- Attempts to establish a connection to the DB using the given url
- Parameters:
  - url: a database url of the form jdbc:postgresql://localhost:5432/csc343h-<user_name>
  - User: your user name
  - Password: empty string
- Returns a connection to the URL

JDBC

Connection class

```
conn.prepareStatement(string sql);
```

- Create a *PreparedStatement* object for sending parametrized SQL statements to the db.
- Parameters:
  - Sql: the query which may contain '?' for parameter placeholders
- Returns a prepared statement object containing a pre-compiled SQL statement

```
conn.close()
```

JDBC

Why do I need to use prepared statements?

1. If you will run the query multiple times
2. Protects against SQL injections

JDBC

PreparedStatement class

- `ps.setInt(int idx, int value);`
- `Ps.setFloat(int idx, double value);`
- `Ps.setString(int idx, String value);`

JDBC

PreparedStatement class

```
ps.executeQuery();
```

- Executes the SQL query in this *PreparedStatement* object and returns the *ResultSet* object generated by the query
- Parameters:
  - Sql: the query which may contain '?' for parameter placeholders
- Returns a *ResultSet* object that contains the data produced by the query

JDBC

PreparedStatement class

```
ps.executeUpdate();
```

- Executes the SQL DML statement in this *PreparedStatement*
- Returns the number of affected rows

JDBC

ResultSet class

```
rs.next();
```

- Iterate over the rows

```
rs.getInt(String name); rs.getString(String name);
rs.getInt(int idx); rs.getString(int idx);
```

- Iterate over the columns