Week 12

*JDBC*

* Java Database Connector (but all good languages have ways of connecting to databases)
* Import java.sql.\*
* Connects with any SQL database (where supported by the vendor)
* Able to configure, connect and disconnect databases, create queries and produce results.
* Requires a vendor-specific driver to bridge the gap between Java and RDBMS
  + Java application ⬄ JDBC framework ⬄ JDBC driver ⬄ DBMS
  + Need to download specific driver
* We need to manage every step of the process
  + Load driver at runtime ⬄ Open/authenticate connection ⬄ Create/execute queries ⬄ Parse results ⬄ Close connection
  + All code must be wrapped in a try/catch
  + Web applications automate this process – efficiency
* Using JDBC:
  + Load driver at runtime
  + Create a connection (need to know connection string)
  + Connect with the DriverManager class (exception possible)
  + Create Statement – object that executes a query (exception possible)
  + Execute query
    - For SELECT: executeQuery()
    - else: executeUpdate()
  + Parse results
    - Results returned as “table”
      * Rows are tuples/records, columns are attributes
      * Iterate through the rows and collect columns through attribute name
      * Specify data – throws exception if incorrect type
  + Close connection
    - Multiple connections can be open but this is resource-expensive

*SQL Injection*

* In any web-based database with user input with poorly constructed queries, malicious SQL can be injected.
* Malicious SQL can drop tables, steal data, insert malicious data, override permissions, etc.
* Use PreparedStatement to prevent (can stop injection completely when used correctly)