

HW6: Appointment Reservation System



Administrivia

- > HW6 released on Wednesday:
 - Part 1: suggested date of completion Nov 24;
 - Part 1 & Part 2: due Dec 3;
 - Setup due Nov 22 (5 points);
- > HW4, HW5, and Midterm grades should be out next week;



Agenda

- > **Assignment Introduction**
- > **Exceptions**
- > **Running SQL Queries**
- > **Handling Passwords**
- > **Demo: create and login for caregivers**



Assignment Introduction

- > Appointment scheduler for vaccinations via the command-line interface, connected to Azure.
- > Objective: Gain experience with database application development.
- > Two versions available: Java (JDBC) and Python (pymssql).
- > Two Parts:
 - Part 1: Computer setup, database design (E/R diagram), and implementation of two operations.
 - Part 2: Implement the rest of the application, and optional extra credits.
- > Note: our solution is about 600 lines of code.



Exceptions

(some) Slides generously
provided by Kevin Zatloukal
and CSE 311 staff.



Not all “errors” should be failures

> Some “error” cases:

1. Misuse of code.

- > E.g., precondition violation.
- > Should be a failure.

2. Typos in code.

- > E.g., referencing a variable that does not exist in the scope.
- > Should be a failure.

3. Unexpected resource problems.

- > E.g., missing files, server offline.
- > Should not be a failure.



Errors vs. Exceptions

- > **Error: an illegal operation performed by the user which results in the abnormal working of the program.**
 - Compile-time error;
 - Runtime error;
 - Logical error;
- > **Exceptions: an unexpected event, which occurs during the execution of a program (runtime) that disrupts the normal flow of the program's instructions.**
 - **Checked exceptions: exceptions that are checked at compile time.**
 - > E.g., IOException, SQLException.
 - > Either handle the exception or specify the exception using the throws keyword.
 - **Unchecked exceptions: those are “basically” runtime errors.**



Running SQL Queries



Java: JDBC

- > Standard API that allows Java programs to access database management systems.
- > PreparedStatement: prevent SQL Injection attacks.

```
Statement withoutPlaceholder = con.createStatement();
withoutPlaceholder.execute( sql: "INSERT INTO students VALUES('\\" + userInput + "\\')");

PreparedStatement withPlaceholder = con.prepareStatement( sql: "INSERT INTO student VALUES(?)");
withPlaceholder.setString( parameterIndex: 1, userInput);
withPlaceholder.execute();
```

- > What if the user input was
"Robert'); DROP TABLE students; --"?



Java: JDBC

```
String selectUsername = "SELECT * FROM Caregivers WHERE Username = ?";
try {
    PreparedStatement statement = con.prepareStatement(selectUsername);
    statement.setString(1, username);
    ResultSet resultSet = statement.executeQuery();
    // returns false if the cursor is not before the first record or if there are no rows in the ResultSet.
    return resultSet.isBeforeFirst();
} catch (SQLException e) {
    System.out.println("Error occurred when checking username");
    e.printStackTrace();
} finally {
    cm.closeConnection();
}
```

- > Try-Catch block: Catch and handle exception.
- > Finally: code inside the finally clause will always be executed.

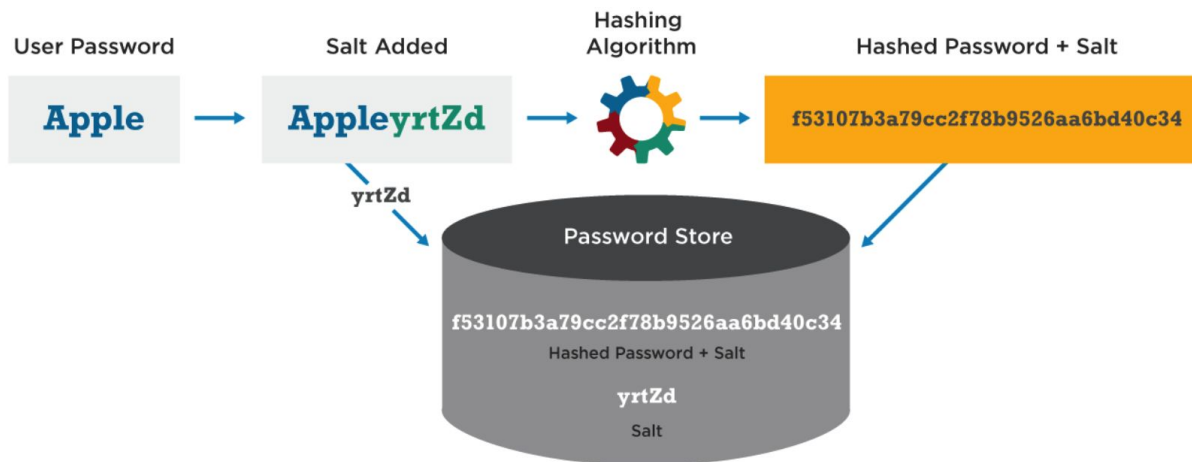


Authentication



Password Hashing and Salt

- > Instead of user password, store Hash(password).
 - System does not store actual passwords.
 - When user enters password, compute its hash and compare with entry in password file.
- > Not entirely safe: Dictionary Attack.
 - Many passwords come from a small dictionary.
 - Attacker pre-compute Hash(password) for all words in the dictionary.
- > Password salting



W