```
import java.sql.*;
import java.util.*;

public class GuessingNumberGame {

    private static String computerNumber;
    private static int moves;
    private static long startTime;

    public static void main(String[] args) throws SQLException {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter your name: ");
        String playerName = scanner.nextLine();



        startNewGame(playerName);

        boolean isGuessed = false;

        while (!isGuessed) {
            System.out.print("Enter your 4-digit guess: ");
            String userGuess = scanner.nextLine();

            if (userGuess.length() != 4 || !userGuess.matches("\\d+")) {
                System.out.println("Invalid input. Please enter exactly 4 digits.");
                continue;
            }

            String feedback = getFeedback(userGuess);
```

```
            System.out.println("Feedback: " + feedback);

            moves++;
            if (feedback.equals("++++")) {
                long endTime = System.currentTimeMillis();
                double timeTaken = (endTime - startTime) / 1000.0;

                System.out.println("Congratulations! You guessed the number in " + moves + "
moves and " + timeTaken + " seconds.");

                saveScore(playerName, moves, timeTaken);
                displayBestScore();
                isGuessed = true;
            }
        }

        scanner.close();
    }

    private static void startNewGame(String playerName) throws SQLException {
        moves = 0;
        computerNumber = generateRandomNumber();
        startTime = System.currentTimeMillis();

        try (Connection connection = DatabaseUtil.getConnection()) {
            String insertUser = "INSERT INTO Users (name) VALUES (?)";
            PreparedStatement stmt = connection.prepareStatement(insertUser,
Statement.RETURN_GENERATED_KEYS);
            stmt.setString(1, playerName);
            stmt.executeUpdate();
        }
```

```
    System.out.println("Game started! Try to guess the 4-digit number.");
}

private static String generateRandomNumber() {
    Random random = new Random();
    Set<Integer> digits = new LinkedHashSet<>();

    while (digits.size() < 4) {
        digits.add(random.nextInt(10));
    }

    StringBuilder number = new StringBuilder();
    for (int digit : digits) {
        number.append(digit);
    }

    return number.toString();
}

private static String getFeedback(String userGuess) {
    StringBuilder feedback = new StringBuilder();

    for (int i = 0; i < 4; i++) {
        if (userGuess.charAt(i) == computerNumber.charAt(i)) {
            feedback.append("+");
        } else if (computerNumber.contains(String.valueOf(userGuess.charAt(i)))) {
            feedback.append("-");
        }
    }
```

```
        return feedback.toString();
    }

    private static void saveScore(String playerName, int moves, double timeTaken) throws
SQLException {
        double score = (1000.0 / (moves * timeTaken)) * 100;

        try (Connection connection = DatabaseUtil.getConnection()) {
            String getUser = "SELECT id FROM Users WHERE name = ?";
            PreparedStatement stmtUser = connection.prepareStatement(getUser);
            stmtUser.setString(1, playerName);
            ResultSet rs = stmtUser.executeQuery();

            if (rs.next()) {
                int userId = rs.getInt("id");

                String insertScore = "INSERT INTO Scores (user_id, moves, time_taken, score)
VALUES (?, ?, ?, ?)";
                PreparedStatement stmtScore = connection.prepareStatement(insertScore);
                stmtScore.setInt(1, userId);
                stmtScore.setInt(2, moves);
                stmtScore.setDouble(3, timeTaken);
                stmtScore.setDouble(4, score);
                stmtScore.executeUpdate();
            }
        }
    }

    private static void displayBestScore() throws SQLException {
        try (Connection connection = DatabaseUtil.getConnection()) {
```

```
        String bestScoreQuery = """
            SELECT u.name, s.moves, s.time_taken, s.score
            FROM Scores s
            JOIN Users u ON s.user_id = u.id
            ORDER BY s.score DESC
            LIMIT 1
        """;

        PreparedStatement stmt = connection.prepareStatement(bestScoreQuery);
        ResultSet rs = stmt.executeQuery();

        if (rs.next()) {
            System.out.println("Best Score: " + rs.getString("name") +
                    " | Moves: " + rs.getInt("moves") +
                    " | Time Taken: " + rs.getDouble("time_taken") +
                    " | Score: " + rs.getDouble("score"));
        }
    }
  }
}
```

```
CREATE DATABASE GuessingGame;

USE GuessingGame;

CREATE TABLE Users (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL
);

CREATE TABLE Scores (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_id INT,
    moves INT NOT NULL,
    time_taken DOUBLE NOT NULL,
    score DOUBLE NOT NULL,
    FOREIGN KEY (user_id) REFERENCES Users(id)
)
```