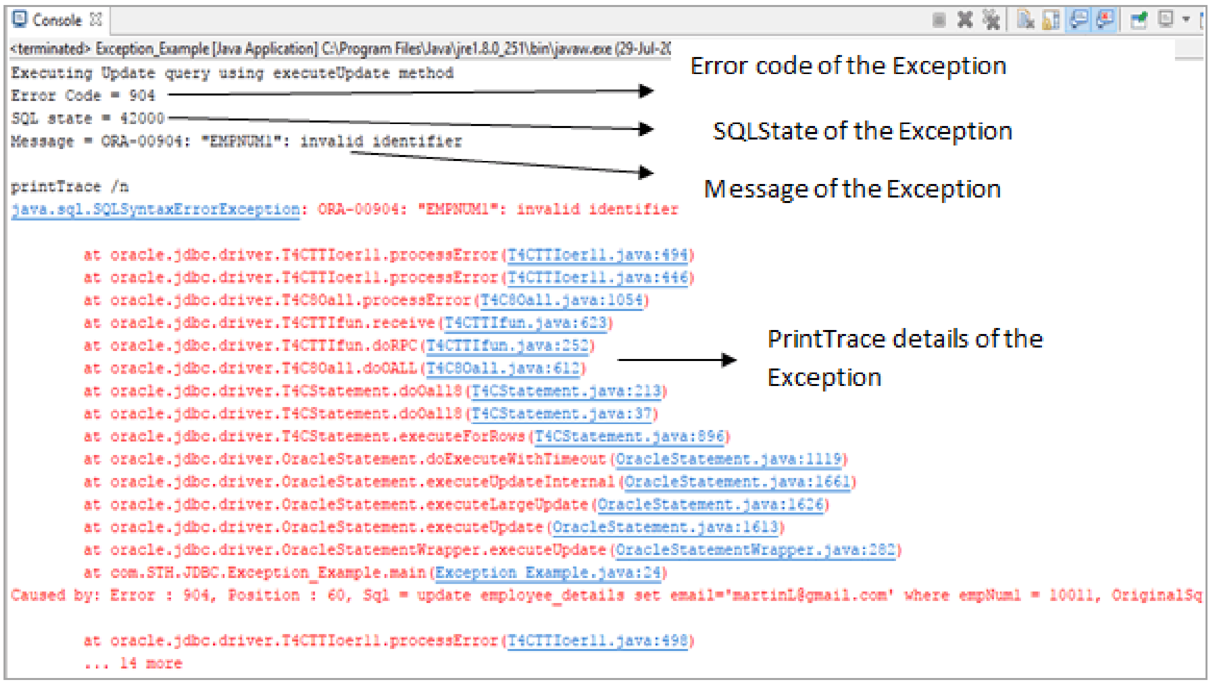
```
package com.STH.JDBC;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class Exception_Example {

public static void main(String[] args) throws ClassNotFoundException {
// TODO Auto-generated method stub

String update_query = "update employee_details set
email='martinL@gmail.com' where empNum1 = 10011";
//Update query to set the email id for the employee whose empNUM is 10011
Class.forName("oracle.jdbc.driver.OracleDriver");
try(Connection conn =
DriverManager.getConnection("jdbc:oracle:thin:system/pass123@localhost:152
1:XE"))
{
Statement statemnt1 = conn.createStatement();
ResultSet rs1 =null;
statemnt1 = conn.createStatement();
System.out.println("Executing Update query using executeUpdate method");
int return_rows = statemnt1.executeUpdate(update_query);
System.out.println("No. of Affected Rows = "+ return_rows);
}
catch(SQLException sqe)
{
System.out.println("Error Code = " + sqe.getErrorCode());
System.out.println("SQL state = " + sqe.getSQLState());
System.out.println("Message = " + sqe.getMessage());
System.out.println("printTrace /n");
sqe.printStackTrace();
}
}
}
```


Console
Error code of the Exception
SQLState of the Exception
Message of the Exception
PrintTrace details of the Exception

```
<terminated> Exception_Example [Java Application] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe (29-Jul-2(
Executing Update query using executeUpdate method
Error Code = 904
SQL state = 42000
Message = ORA-00904: "EMPNUM1": invalid identifier

printTrace /n
java.sql.SQLSyntaxErrorException: ORA-00904: "EMPNUM1": invalid identifier

        at oracle.jdbc.driver.T4CTTIoer11.processError(T4CTTIoer11.java:494)
        at oracle.jdbc.driver.T4CTTIoer11.processError(T4CTTIoer11.java:446)
        at oracle.jdbc.driver.T4C8Oall.processError(T4C8Oall.java:1054)
        at oracle.jdbc.driver.T4CTTIfun.receive(T4CTTIfun.java:623)
        at oracle.jdbc.driver.T4CTTIfun.doRPC(T4CTTIfun.java:252)
        at oracle.jdbc.driver.T4C8Oall.doOALL(T4C8Oall.java:612)
        at oracle.jdbc.driver.T4CStatement.doOall8(T4CStatement.java:213)
        at oracle.jdbc.driver.T4CStatement.doOall8(T4CStatement.java:37)
        at oracle.jdbc.driver.T4CStatement.executeForRows(T4CStatement.java:896)
        at oracle.jdbc.driver.OracleStatement.doExecuteWithTimeout(OracleStatement.java:1119)
        at oracle.jdbc.driver.OracleStatement.executeUpdateInternal(OracleStatement.java:1661)
        at oracle.jdbc.driver.OracleStatement.executeLargeUpdate(OracleStatement.java:1626)
        at oracle.jdbc.driver.OracleStatement.executeUpdate(OracleStatement.java:1613)
        at oracle.jdbc.driver.OracleStatementWrapper.executeUpdate(OracleStatementWrapper.java:282)
        at com.STH.JDBC.Exception_Example.main(Exception_Example.java:24)
Caused by: Error : 904, Position : 60, Sql = update employee_details set email='martinL@gmail.com' where empNum1 = 10011, OriginalSq

        at oracle.jdbc.driver.T4CTTIoer11.processError(T4CTTIoer11.java:498)
        ... 14 more
```

```
import java.sql.*;

public class JdbcExample {
    public static void main(String[] args) {
        // JDBC connection parameters
        String url = "jdbc:mysql://localhost:3306/mydatabase";
        String username = "myuser";
        String password = "mypassword";

        // Connection, CallableStatement, and ResultSet variables
        Connection connection = null;
        CallableStatement callableStatement = null;
        ResultSet resultSet = null;

        try {
            // Step 1: Load and register the JDBC driver
            Class.forName("com.mysql.cj.jdbc.Driver");

            // Step 2: Establish the connection
            connection = DriverManager.getConnection(url, username, password);

            // Step 3: Prepare the stored procedure call
            String procedureCall = "{CALL get_customer_info(?, ?)}";
            callableStatement = connection.prepareCall(procedureCall);

            // Step 4: Set input parameters
            int customerId = 123;
            callableStatement.setInt(1, customerId);

            // Step 5: Register the output parameter
            callableStatement.registerOutParameter(2, Types.VARCHAR);

            // Step 6: Execute the stored procedure
            callableStatement.execute();

            // Step 7: Retrieve the output parameter value
            String customerInfo = callableStatement.getString(2);
            System.out.println("Customer Info: " + customerInfo);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        } catch (SQLException e) {
            // Handle any SQL exceptions
            e.printStackTrace();
        } finally {
            // Step 8: Close the resources
            try {
                if (resultSet != null) {
                    resultSet.close();
                }
                if (callableStatement != null) {
                    callableStatement.close();
                }
                if (connection != null) {
                    connection.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

In this example, we connect to a MySQL database using the JDBC driver `com.mysql.cj.jdbc.Driver`. We establish a connection by providing the database URL, username, and password.

We create a `CallableStatement` object to call a stored procedure named `get_customer_info`. The stored procedure takes an input parameter `customerId` and returns customer information as an output parameter.

We set the input parameter using `setInt()` and register the output parameter using `registerOutParameter()`. Then, we execute the stored procedure using `execute()`.

After executing the stored procedure, we retrieve the output parameter value using `getString()` and print it.
We also handle potential exceptions that may occur during the JDBC operations using try-catch blocks. The `SQLException` is a checked exception and can be caught and handled appropriately.

Please make sure to replace the JDBC driver, database URL, username, and password with appropriate values for your database. Additionally, modify the stored procedure call (`procedureCall`) and input/output parameter handling according to your specific stored procedure's requirements.