```
CREATE OR REPLACE TYPE T_SCHEMA.PARAMETER_ARRAY_TYPE IS TABLE OF
NUMBER(19);

/

CREATE OR REPLACE PACKAGE T_SCHEMA."PCK_TEST" AS

PROCEDURE sp_with_array_parameter(p_array IN PARAMETER_ARRAY_TYPE);

END PCK_TEST;

/

CREATE OR REPLACE PACKAGE BODY T_SCHEMA."PCK_TEST" AS

PROCEDURE sp_with_array_parameter (p_array IN PARAMETER_ARRAY_TYPE) IS

CURSOR c_records (p_array PARAMETER_ARRAY_TYPE) IS SELECT * FROM
MY_TABLE WHERE id IN (SELECT * FROM TABLE (CAST (p_array AS
PARAMETER_ARRAY_TYPE)));

BEGIN

FOR r_record IN c_records (p_array)

LOOP

--some biz logic here

END LOOP;

END;

END PCK_TEST;
/
```

```
import java.sql.Connection;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.jdbc.core.SqlParameter;

import org.springframework.jdbc.core.simple.SimpleJdbcCall;
import org.springframework.jdbc.core.support.AbstractSqlTypeValue;

public class PassArray {

protected JdbcTemplate jdbcTemplate;

private String schemaName="T_SCHEMA";

public class MyArray extends AbstractSqlTypeValue {

private List values;

public MyArray(List values) {

this.values = values;

}

public Object createTypeValue(Connection con, int sqlType,

String typeName) throws SQLException {

oracle.sql.ArrayDescriptor desc = new Oracle.sql.ArrayDescriptor(typeName, con);

return new oracle.sql.ARRAY(desc, con,(Long[])values.toArray(new Long[values.size()]));

}

}

public void callProcedureWithArrayParameter() {

List values = new ArrayList();

values.add(1L);

values.add(2L);

SimpleJdbcCall jdbcCall = new SimpleJdbcCall(jdbcTemplate)

.withSchemaName(schemaName)

.withProcedureName("PCK_TEST.SP_WITH_ARRAY_PARAMETER")

.withoutProcedureColumnMetaDataAccess()

.declareParameters(new SqlParameter("P_ARRAY",java.sql.Types.ARRAY,schemaName +
".PARAMETER_ARRAY_TYPE"));

Map map = new HashMap();

map.put("P_ARRAY", new MyArray(values));
jdbcCall.execute(map);
}
}
```