```
import com.intellij.database.model.DasTable
import com.intellij.database.model.ObjectKind
import com.intellij.database.util.Case
import com.intellij.database.util.DasUtil
import com.intellij.psi.codeStyle.NameUtil

import javax.swing.*

/**
 * @author shirohoo
 * @link https://github.com/shirohoo/create-automation-jpa-entity
 * @param pakageName , primaryKey
 *
 * <pre>
 *
 *     this script's default primary key strategy is
@GeneratedValue(strategy = GenerationType.IDENTITY)
 *     and specialized in Microsoft SQL Server
 *     and finally implemented Serializable so recommend that create serial
version UID
 *
 *     first. enter your project package name. for example:
 *     >  com.intelliJ.psi
 *
 *     second. enter primary key column name of target database table.
 *     this script is convert input to camel case. for example 1:
 *     >  table primary key column name = MEMBER_ID
 *     >  enter primary key = memberId
 *
 *     example 2:
 *     >  table primary key column name = ID
 *     >  enter primary key = id
 *
 * </pre>
 */

columnType = [
        (~/(?i)bigint/)             : "Long",
        (~/(?i)int/)                : "Integer",
        (~/(?i)bit/)                : "Boolean",
        (~/(?i)decimal/)            : "BigDecimal",
        (~/(?i)float|double|real/) : "Double",
        (~/(?i)datetime|timestamp/): "LocalDateTime",
        (~/(?i)time/)               : "LocalTime",
        (~/(?i)date/)               : "LocalDate",
        (~/(?i)nvarchar/)           : "nvarchar",
        (~/(?i)varchar/)            : "varchar",
        (~/(?i)char/)               : "String",
        (~/(?i)text/)               : "String"
]
def input = {
    JFrame jframe = new JFrame()
    String answer = JOptionPane.showInputDialog(jframe, it)
    jframe.dispose()
    answer
}
packageName = input("Enter your package name")
primaryKey = input("Enter column name of primary key")
FILES.chooseDirectoryAndSave("Choose directory", "Choose where to store
generated files") { dir ->
    SELECTION.filter {
        it instanceof DasTable && it.getKind() == ObjectKind.TABLE
    }.each {
        generate(it, dir)
    }
}

def generate(table, dir) {
    def tableName = table.getName()
    def className = convertFieldName(tableName + "Entity", true)
    def fields = categorizeFields(table)
    new File(dir, className + ".java").withPrintWriter {
        out -> generate(out, tableName, className, fields)
    }
}

def generate(out, tableName, className, fields) {
    out.println "package $packageName;"
    out.println ""
    out.println "import javax.persistence.*;"
    out.println "import java.time.LocalDateTime;"
    out.println "import lombok.Data;"
    out.println ""
    out.println "@Entity"
    out.println "@Data"
    out.println "@Table(name = \"$tableName\")"
    out.println "public class $className"
    out.println "{"
    fields.each() {
        if (it.annos != "") {
            out.println "   ${it.annos}"
        }
        if (it.name == primaryKey) {
            out.println "   @Id"
            out.println "   @GeneratedValue(strategy =
GenerationType.IDENTITY)"
        }
        if (it.type == 'nvarchar') {
            out.println "   @Nationalized"
            out.println "   @Column(name = \"${it. colName}\")"
            out.println "   private String ${it.name};"
        } else if (it.type == 'varchar') {
          out.println "   @Column(name = \"${it.colName}\")"
          out.println "   private String ${it.name};"
      } else {
          out.println "   @Column(name = \"${it.colName}\")"
          out.println "   private ${it.type} ${it.name};"
      }
      out.println ""
    }
    out.println "}"
}

def categorizeFields(table) {
    DasUtil.getColumns(table).reduce([]) { fields, col ->
        def spec = Case.LOWER.apply(col.getDataType().getSpecification())
        def typeStr = columnType.find {
            p, t -> p.matcher(spec).find()
        }.value
        fields += [[
                           colName: col.getName(),
                           name   : convertFieldName(col.getName(), false),
                           type   : typeStr,
                           annos  : ""]]
    }
}

def convertFieldName(str, capitalize) {
    def s = NameUtil.splitNameIntoWords(str)
            .collect {
                Case.LOWER.apply(it).capitalize()
            }
            .join("")
            .replaceAll(/[^\p{javaJavaIdentifierPart}[_]]/, "_")
    capitalize || s.length() == 1 ? s : Case.LOWER.apply(s[0]) + s[1..-1]
}
```