```java
//Recipes.java

package recipes;

import recipes.entity.Recipe;
import recipes.exception.*;
import recipes.service.RecipeService;

import java.time.LocalTime;
import java.util.*;

public class Recipes {

    private Scanner scanner = new Scanner(System.in);
    private RecipeService recipeService = new RecipeService();

    private List<String> operations = List.of(
            "1) Create and populate tables",
            "2) Add a recipe"
            );

    public static void main(String[] args) {
        new Recipes().displayMenu();


    }

    private void displayMenu() {
        boolean done = false;

        while(!done) {
            try {
            int operation = getOperation();
            switch(operation) {
            case -1:
                done = exitMenu();
                break;

            case 1:
                createTables();
                break;

            case 2:
                addRecipe();
                break;

            default:
                System.out.println("\n"+ operation + " is not valid. Please try again.");
                break;

            }
            }catch(Exception e) {
                System.out.println("\nError: "+ e.toString() + " Try again.");
            }
        }

    }

    private void addRecipe() {
        String name = getStringInput("Enter the recipe name: ");
        String notes = getStringInput("Enter the recipe notes: ");
        Integer NumServings = getIntInput("Enter the number of servings: ");
        Integer prepMinutes = getIntInput("Enter prep time in minutes: ");
        Integer cookMinutes = getIntInput(" Enter cook time in minutes: ");

        LocalTime preptime = minutesToLocalTime(prepMinutes);
        LocalTime cooktime = minutesToLocalTime(cookMinutes);

        Recipe recipe = new Recipe();

        recipe.setRecipeName(name);
        recipe.setNotes(notes);
        recipe.setNumServings(NumServings);
        recipe.setPrepTime(preptime);
        recipe.setCookTime(cooktime);
        Recipe dbRecipe = recipeService.addRecipe(recipe);
        System.out.println("You added this recipe: \n"+ dbRecipe);


    }

    private LocalTime minutesToLocalTime(Integer numMinutes) {
        int min = Objects.isNull(numMinutes) ? 0 : numMinutes;
        int hours = min / 60;
        int minutes = min % 60;

        return LocalTime.of(hours, minutes);

    }

    private void createTables() {
        recipeService.createAndPopulateTables();
        System.out.println("\nTables created and populated!");

    }

    private boolean exitMenu() {
        System.out.println("\nYou have now EXITED the menu.");
        return true;
    }

    private int getOperation() {
        printOperations();
        Integer op = getIntInput("\nEnter a operation number (Press ENTER to quit)");

        return Objects.isNull(op) ? -1 : op;
    }


    private void printOperations() {
        System.out.println();
        System.out.println("Here's what you can do:");

        operations.forEach(op -> System.out.println("   "+ op));

    }

    private Integer getIntInput(String prompt) {
        String input = getStringInput(prompt);

        if(Objects.isNull(input)) {
            return null;
        }
        try {
            return Integer.parseInt(input);
        } catch(NumberFormatException e) {
            throw new DbException(input + " is not a valid number.");
        }
    }

    @SuppressWarnings("unused")
    private Double getDoubleInput(String prompt) {
        String input = getStringInput(prompt);

        if(Objects.isNull(input)) {
            return null;
        }
        try {
            return Double.parseDouble(input);
        } catch(NumberFormatException e) {
            throw new DbException(input + "is not a valid number.");
        }
    }

    private String getStringInput(String prompt) {
        System.out.print(prompt + ": ");
        String line = scanner.nextLine();

        return line.isBlank() ? null : line.trim();
    }
```

```java
}



//DbConnection.java

package recipes.dao;

import java.sql.*;
import recipes.exception.DbException;

public class DbConnection {
    private static String HOST = "localhost";
    private static String PASSWORD = "recipes";
    private static int PORT = 3306;
    private static String SCHEMA = "recipes";
    private static String USER = "recipes";

    public static Connection getConnection() {
        String url = String.format("jdbc:mysql://%s:%d/%s?user=%s&password=%s&useSSL=false",
                HOST, PORT, SCHEMA, USER, PASSWORD);
        try {
        Connection connection = DriverManager.getConnection(url);
        System.out.println("The connection succeeded!");
        return connection;
        }catch(SQLException e) {
            System.out.println("The connection failed.");
            throw new DbException(e);
        }

    }
}




//RecipeDao.java
package recipes.dao;

import java.sql.*;
import java.util.List;
import provided.util.DaoBase;
import recipes.entity.Recipe;
import recipes.exception.DbException;
import java.time.*;

public class RecipeDao extends DaoBase {

    private static final String CATEGORY_TABLE = "category";
    private static final String INGREDIENT_TABLE = "ingredient";
    private static final String RECIPE_TABLE = "recipe";
    private static final String RECIPE_CATEGORY = "recipe_category";
    private static final String STEP_TABLE = "step";
    private static final String UNIT_TABLE = "unit";

    public Recipe insertRecipe(Recipe recipe) {
        String sql = " " + "INSERT INTO " + RECIPE_TABLE + " "
                + "(recipe_name , notes , num_servings, prep_time, cook_time) " + "VALUES " + " (?, ?, ?, ?, ?)";

        try (Connection conn = DbConnection.getConnection()) {
            startTransaction(conn);
            try (PreparedStatement stmt = conn.prepareStatement(sql)) {
                setParameter(stmt, 1, recipe.getRecipeName(), String.class);
                setParameter(stmt, 2, recipe.getNotes(), String.class);
                setParameter(stmt, 3, recipe.getNumServings(), Integer.class);
                setParameter(stmt, 4, recipe.getPrepTime(), LocalTime.class);
                setParameter(stmt, 5, recipe.getCookTime(), LocalTime.class);

                stmt.executeUpdate();
                Integer recipeId = getLastInsertId(conn, RECIPE_TABLE);

                commitTransaction(conn);

                recipe.setRecipeId(recipeId);
                return recipe;
            } catch (Exception e) {
                rollbackTransaction(conn);
                throw new DbException(e);
            }
        } catch (SQLException e) {
            throw new DbException(e);
        }
    }

    public void executeBatch(List<String> sqlBatch) {
        try (Connection conn = DbConnection.getConnection()) {
            startTransaction(conn);

            try (Statement stmt = conn.createStatement()) {
                for (String sql : sqlBatch) {
                    stmt.addBatch(sql);
                }

                stmt.executeBatch();
                commitTransaction(conn);

            } catch (Exception e) {
                rollbackTransaction(conn);
                throw new DbException(e);
            }
        } catch (SQLException e) {
            throw new DbException(e);
        }
    }

}




//Category.java
package recipes.entity;

public class Category {
    private Integer categoryId;
    private String categoryName;

    public String getCategoryName() {
        return categoryName;
    }
    public void setCategoryName(String categoryName) {
        this.categoryName = categoryName;
    }
    public Integer getCategoryId() {
        return categoryId;
    }
    public void setCategoryId(Integer categoryId) {
        this.categoryId = categoryId;
    }
    @Override
    public String toString() {
        return "ID=" + categoryId + ", categoryName=" + categoryName;
    }

}



//Ingredient.java

package recipes.entity;

import java.math.BigDecimal;
import java.util.Objects;

import provided.entity.EntityBase;

public class Ingredient extends EntityBase{
    private Integer ingredientId;
    private Integer recipe_id;
    private unit unit;
    private String ingredientName;
    private String instruction;
    private Integer ingredientOrder;
    private BigDecimal amount;

    @Override
    public String toString() {
        StringBuilder b = new StringBuilder();

        b.append("ID =").append(ingredientId).append(": ");
        b.append(toFraction(amount));

        if(Objects.nonNull(unit) && Objects.nonNull(unit.getUnitId())) {
            String singular = unit.getUnitNameSingular();
            String plural = unit.getUnitNamePlural();
            String word = amount.compareTo(BigDecimal.ONE) > 0 ? plural : singular;

            b.append(word).append(" ");
        }
        b.append(ingredientName);
        if(Objects.nonNull(instruction)) {
            b.append(", ").append(instruction);
        }
        return b.toString();
    }
    public Integer getIngregredient_id() {
        return ingredientId;
    }
    public void setIngregredient_id(Integer ingregredient_id) {
        this.ingredientId = ingregredient_id;
    }
    public Integer getRecipe_id() {
        return recipe_id;
    }
    public void setRecipe_id(Integer recipe_id) {
        this.recipe_id = recipe_id;
    }
    public unit getUnit() {
        return unit;
    }
    public void setUnit(unit unit) {
        this.unit = unit;
    }
    public String getIngredientName() {
        return ingredientName;
    }
    public void setIngredientName(String ingredientName) {
        this.ingredientName = ingredientName;
    }
    public String getInstruction() {
        return instruction;
    }
    public void setInstruction(String instruction) {
        this.instruction = instruction;
    }
    public Integer getIngredientOrder() {
        return ingredientOrder;
    }
    public void setIngredientOrder(Integer ingredientOrder) {
        this.ingredientOrder = ingredientOrder;
    }
    public BigDecimal getAmount() {
        return amount;
    }
    public void setAmount(BigDecimal amount) {
        this.amount = amount;
    }

}



//Step.java

package recipes.entity;
```

```java
public class Step {
    private Integer stepId;
    private Integer recipeId;
    private Integer stepOrder;
    private String stepText;

    @Override
    public String toString() {
        return "ID=" + stepId + ", stepText=" + stepText;
    }

    public Integer getStepId() {
        return stepId;
    }

    public void setStepId(Integer stepId) {
        this.stepId = stepId;
    }

    public Integer getRecipeId() {
        return recipeId;
    }

    public void setRecipeId(Integer recipeId) {
        this.recipeId = recipeId;
    }

    public Integer getStepOrder() {
        return stepOrder;
    }

    public void setStepOrder(Integer stepOrder) {
        this.stepOrder = stepOrder;
    }

    public String getStepText() {
        return stepText;
    }

    public void setStepText(String stepText) {
        this.stepText = stepText;
    }
}




//Unit.java

package recipes.entity;

public class unit {
    public Integer getUnitId() {
        return unitId;
    }
    public void setUnitId(Integer unitId) {
        this.unitId = unitId;
    }
    public String getUnitNameSingular() {
        return unitNameSingular;
    }
    public void setUnitNameSingular(String unitNameSingular) {
        this.unitNameSingular = unitNameSingular;
    }
    public String getUnitNamePlural() {
        return unitNamePlural;
    }
    public void setUnitNamePlural(String unitNamePlural) {
        this.unitNamePlural = unitNamePlural;
    }
    private Integer unitId;
    private String unitNameSingular;
    private String unitNamePlural;
    @Override
    public String toString() {
        return "unit [unitId=" + unitId + ", unitNameSingular=" + unitNameSingular + ", unitNamePlural="
                + unitNamePlural + "]";
    }

}





//DebException.java

package recipes.exception;

@SuppressWarnings("serial")
public class DbException extends RuntimeException {

    public DbException() {
    }

    public DbException(String message) {
        super(message);
    }

    public DbException(Throwable cause) {
        super(cause);
    }

    public DbException(String message, Throwable cause) {
        super(message, cause);
    }

}





//RecipeService.java
package recipes.service;

import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.util.*;
import recipes.exception.DbException;
import recipes.dao.*;
import recipes.entity.Recipe;

public class RecipeService {
    private static final String SCHEMA_FILE = "recipe_schema.sql";
    private static final String DATA_FILE = "recipe_data.sql";

    private RecipeDao recipeDao = new RecipeDao();

    public void createAndPopulateTables() {
        loadFromFile(SCHEMA_FILE);
        loadFromFile(DATA_FILE);
    }

    private void loadFromFile(String fileName) {
        String content = readFileContent(fileName);
        List<String> sqlStatements = convertContentToSqlStatements(content);

        recipeDao.executeBatch(sqlStatements);
    }

    private List<String> convertContentToSqlStatements(String content) {
        content = removeComments(content);
        content = replaceWhiteSpaceSequencesWithSingleSpace(content);

        return extractLinesFromContent(content);
    }

    private List<String> extractLinesFromContent(String content) {
        List<String> lines = new LinkedList<>();

        while (!content.isEmpty()) {
            int semicolon = content.indexOf(";");

            if (semicolon == -1) {
                if (!content.isBlank()) {
                    lines.add(content);
                }
                content = "";
            } else {
                lines.add(content.substring(0, semicolon).trim());
                content = content.substring(semicolon + 1);
            }
        }

        return lines;
    }

    private String replaceWhiteSpaceSequencesWithSingleSpace(String content) {
        return content.replaceAll("\\s+", " ");
    }

    private String removeComments(String content) {
        StringBuilder builder = new StringBuilder(content);
        int commentPos = 0;

        while ((commentPos = builder.indexOf("-- ", commentPos)) != -1) {
            int eolPos = builder.indexOf("\n", commentPos + 1);

            if (eolPos == -1) {
                builder.replace(commentPos, builder.length(), "");
            } else {
                builder.replace(commentPos, eolPos + 1, "");
            }
        }
        return builder.toString();
    }

    private String readFileContent(String fileName) {
        try {
            Path path = Paths.get(getClass().getClassLoader().getResource(fileName).toURI());
            return Files.readString(path);
        } catch (Exception e) {
            throw new DbException(e);
        }
    }

    public Recipe addRecipe(Recipe recipe) {
        return recipeDao.insertRecipe(recipe);

    }
    public static void main(String[] args) {
        new RecipeService().createAndPopulateTables();
    }

}
```

```
// REFERENCES:
// https://youtu.be/EeqkC39WO7o
// https://github.com/fmd5045/Week07-11SQLRecipe
```