Design Patterns

Below I describe the three design patterns I found, as proposed.

The first “design pattern”: Singleton Class

```
private JabRefPreferences() {
    try {
        if (new File("jabref.xml").exists()) {
            importPreferences(Path.of("jabref.xml"));
        }
    } catch (JabRefException e) {
        LOGGER.warn("Could not import preferences from jabref.xml: " +
e.getMessage(), e);
    }

    // load user preferences
    prefs = PREFS_NODE;


        ...
```

This block of code is located in **jabref > preferences > JabRefPreferences.java**

A private constructor, such as the above, ensures there is only one instance of this class to be produced and exist throughout an application's lifetime and guarantee that exists a global point of access to it.

The second “design pattern”: Adapter Class

```
/**
 * Provides an adapter class to CSL. It holds a CSL instance under the
*hood that is only recreated when
 * the style changes.
*/

public class CSLAdapter {

        ...
```

This block of code is located in **jabref > logic > citationstyle > CSLAdapter.**

As described in the comment this class acts as an adapter class, converting incompatible objects to make them compatible with other classes. In this exemple, converting the bibEntry from our own representation to a representation that the CLS library uses to create the preview string, for example.

The third “design pattern”: Template Method

```
DBMSProcessor.java

/**
 * Creates and sets up the needed tables and columns according to the
database type.
 *
 * @throws SQLException
 */
protected abstract void setUp() throws SQLException;

/**
 * Escapes parts of SQL expressions such as a table name or a field
name to match the conventions of the database
 * system using the current dbmsType.
 * <p>
 * This method is package private, because of DBMSProcessorTest
 *
 * @param expression Table or field name
 * @return Correctly escaped expression
 */
abstract String escape(String expression);


--------------------------------------------------------------------------
MySQLProcessor.java

@Override
public void setUp() throws SQLException {
    connection.createStatement().executeUpdate(
            "CREATE TABLE IF NOT EXISTS `ENTRY` (" +
                    "`SHARED_ID` INT(11) NOT NULL PRIMARY KEY
AUTO_INCREMENT, " +
                    "`TYPE` VARCHAR(255) NOT NULL, " +
                    "`VERSION` INT(11) DEFAULT 1)");

    connection.createStatement().executeUpdate(
            "CREATE TABLE IF NOT EXISTS `FIELD` (" +
                    "`ENTRY_SHARED_ID` INT(11) NOT NULL, " +
                    "`NAME` VARCHAR(255) NOT NULL, " +
                    "`VALUE` TEXT DEFAULT NULL, " +
                    "FOREIGN KEY (`ENTRY_SHARED_ID`) REFERENCES
`ENTRY`(`SHARED_ID`) ON DELETE CASCADE)");

    connection.createStatement().executeUpdate(
            "CREATE TABLE IF NOT EXISTS `METADATA` (" +
                    "`KEY` varchar(255) NOT NULL," +
                    "`VALUE` text NOT NULL)");
}

@Override
String escape(String expression) {
    return "`" + expression + "`";
}

--------------------------------------------------------------------------
```

```
class DBMSProcessorTest {

    private DBMSConnection dbmsConnection;
    private DBMSProcessor dbmsProcessor;
    private DBMSType dbmsType;

    @BeforeEach
    public void setup() throws Exception {
        this.dbmsType = TestManager.getDBMSTypeTestParameter();
        this.dbmsConnection =
TestConnector.getTestDBMSConnection(dbmsType);
        this.dbmsProcessor =
DBMSProcessor.getProcessorInstance(TestConnector.getTestDBMSConnection
(dbmsType));
```

This block of code is located in **jabref > logic > shared > DBMSProcessorTest.java.**

The abstract class DBMSProcessor defines the skeleton of the methods and describes them in a general way, leaving the implementation and details to its subclasses. These methods are abstract in the DBMSProcessor class and consequently implemented in the following classes: OracleProcessor, MySQLProcessor, PostgreSQLProcessor, to be used in the class DBMSProcessorTest.