// Description of how field declared in database and in cache.
public class JdbcTypeField implements Serializable {
    /** Field JDBC type in database. */
    private int dbFieldType;

    /** Field name in database. */
    private String dbFieldName;

    /** Field java type. */
    private Class<?> javaFieldType;

    /** Field name in java object. */
    private String javaFieldName;

    // True if this field should be used for hash-code calculation.
    private boolean isHashCode();
...
}

// Description for type that could be stored into database by store.
public class JdbcType implements Serializable {
    /** Cache name. */
    private String cacheName;

    /** Schema name in database. */
    private String dbSchema;

    /** Table name in database. */
    private String dbTbl;

    /** Key class used to store key in cache. */
    private String keyType;

    /** List of fields descriptors for key object. */
    @GridToStringInclude
    private JdbcTypeField[] keyFields;

    /** Value class used to store value in cache. */
    private String valType;

    /** List of fields descriptors for value object. */
    @GridToStringInclude
    private JdbcTypeField[] valFields;
...
}

// JDBC POJO store factory.
public class CacheJdbcPojoStoreFactory<K, V> implements Factory<CacheJdbcPojoStore<K, V>> {
    /** Maximum batch size for writeAll and deleteAll operations. */
    private int batchSz = DFLT_BATCH_SIZE;

    /** Name of data source bean. */
    private String dataSrcBean;

    /** Database dialect. */
    private JdbcDialect dialect;

    /** Max workers thread count. These threads are responsible for load cache. */
    private int maxPoolSz = Runtime.getRuntime().availableProcessors();

    /** Maximum write attempts in case of database error. */
    private int maxWrtAttempts = DFLT_WRITE_ATTEMPTS;

    /** Parallel load cache minimum threshold. If 0 code 0 then load sequentially. */
    private int parallelLoadCacheMinThreshold = DFLT_PARALLEL_LOAD_CACHE_MINIMUM_THRESHOLD;

    /** Types that store could process. */
private JdbcType[] types;

/** Data source. */
private transient DataSource dataSrc;

/** Application context. */
@SpringApplicationContextResource
private transient Object appCtx;
...
}