**利用Idea 重构功能及Java8语法特性 优化深层嵌套代码**

　　当遇到深层嵌套代码，如for,if,lambda表达式或内部类及这些代码的组合，这时我们可以通过Java 8的语法特性来进行优化。

　　下面的代码是一个嵌套循环的示例。

　　public MappedField getMappedField(final String storedName) {

　　for (final MappedField mf : persistenceFields) {

　　for (final String n : mf.getLoadNames()) {

　　if (storedName.equals(n)) {

　　return mf;

　　}

　　}

　　}

　　return null;

　　}

**重构1:**

　　嵌套的for/if语句通畅可以通过Java 8中的stream来替代。

　　Optional<String> found = persistenceFields.stream()

　　.flatMap(mappedField -> mappedField.getLoadNames().stream())

　　.filter(storedName::equals)

　　.findFirst();

上述重构代码会返回Optional，但笔者希望返回mappedField对象，再次改造后的代码如下。

　　persistenceFields.stream()

　　.filter(mappedField -> {

　　for (String name : mappedField.getLoadNames()) {

　　if (storedName.equals(name)) {

　　return true;

　　}

　　}

　　return false;

　　})

　　.findFirst()

**重构2： 进行更好的封装**

重构1还存在一些问题，我们需要了解mappedField的结构，并通过循环遍历其所有name来找到匹配的name。根据迪米特法则（Law of Demeter ），及命令-不要去询问法则(Tell, Don’t Ask), 下面代码应该由MappedField对象来提供对应的方法来判断，而不是由调用者去了解MappedField结构后去写逻辑进行判断。

　　for (final MappedField mf : persistenceFields) {

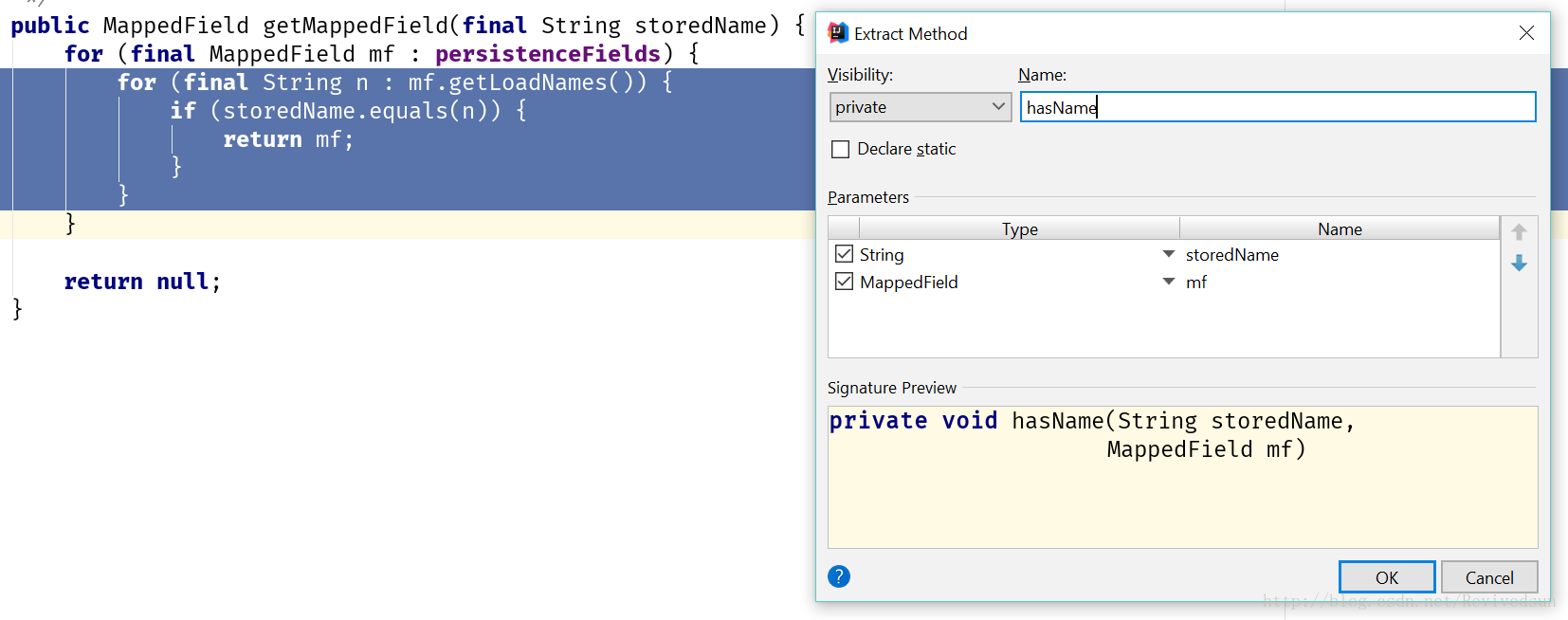
　　if (mf.hasName(storedName)) {

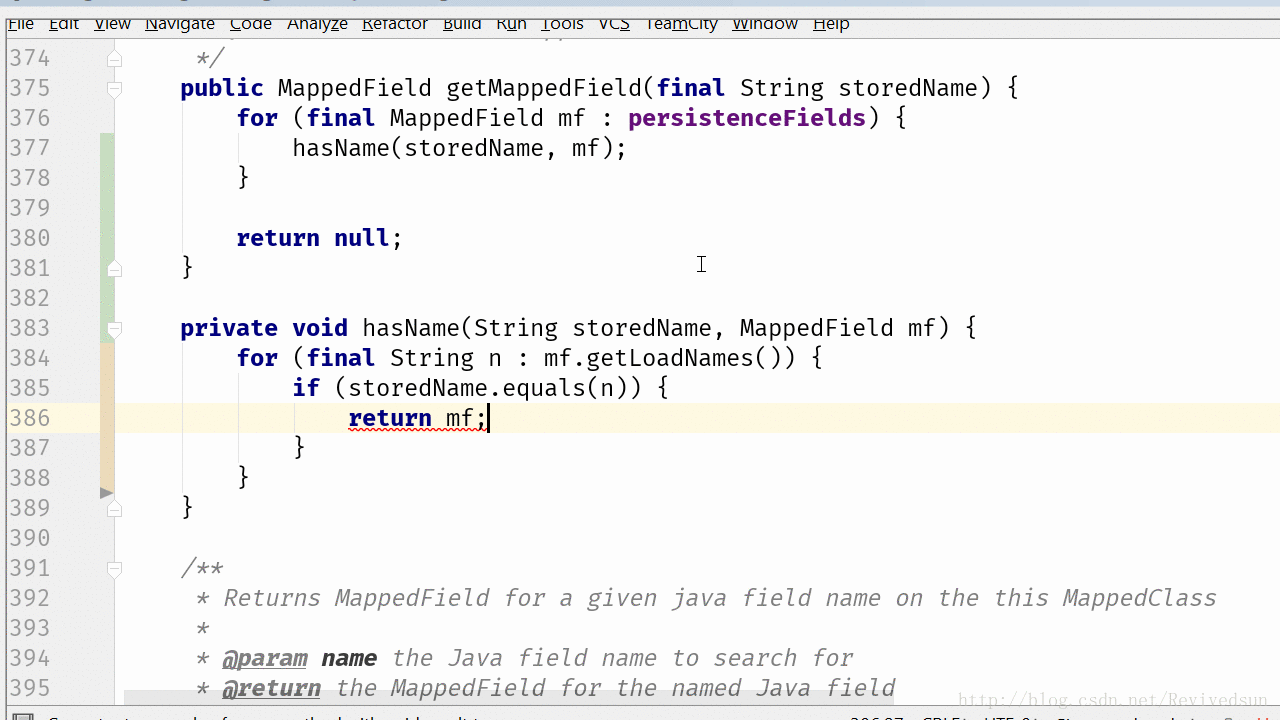
　　return mf;

　　}

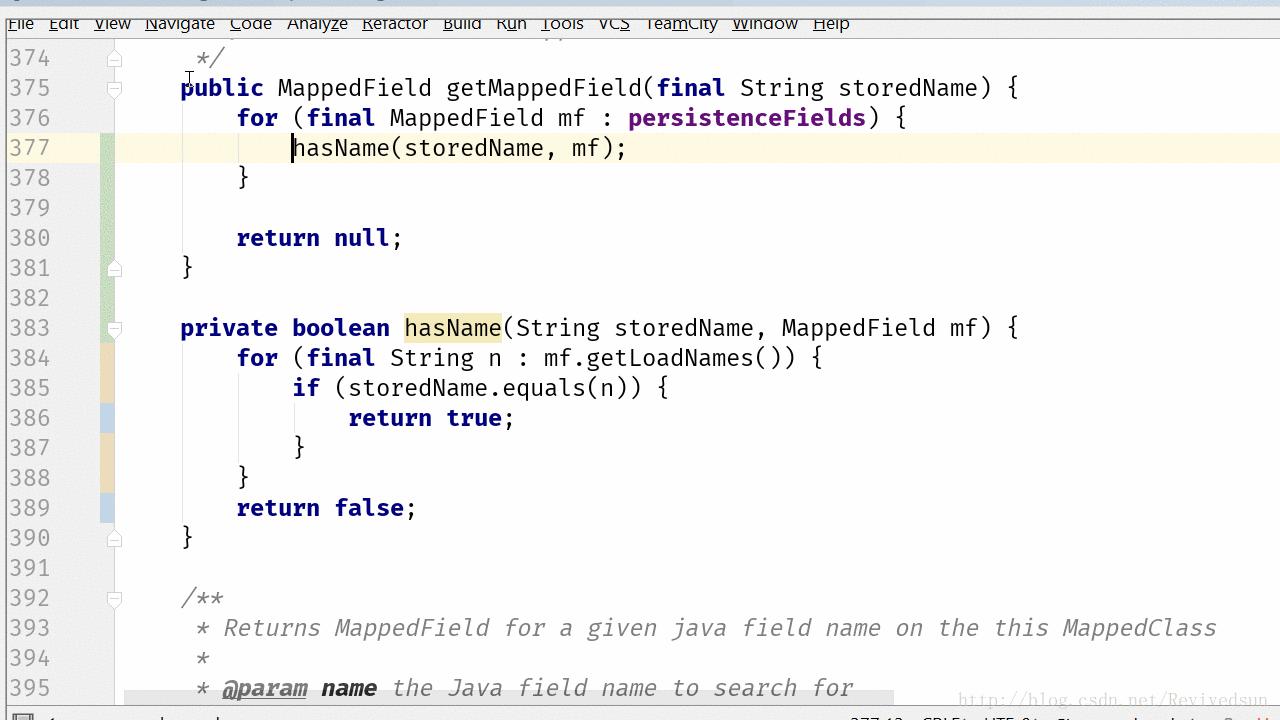
　　}

　因此将上述代码提取为MappedField类中独立的方法，并命名为hasName。如果使用的IDE 是IDEA则可以通过refractor中的extract功能完成提取。

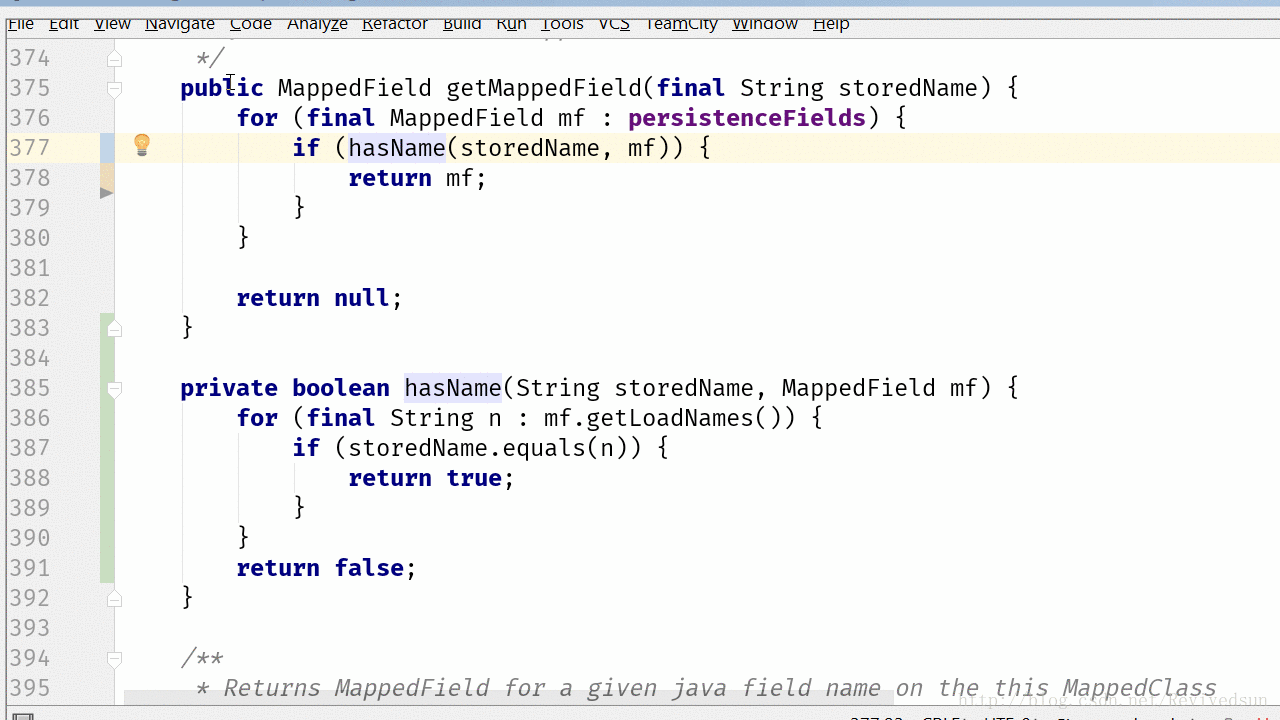




最后调用hasName方法来替代循环判断逻辑。



接着通过Idea的refractor 中的move功能将代码移动到目标类位置。



　　接着通过stream来重构hasName方法，hasName方法变更为下面的形式。

　　public boolean hasName(String storedName) {

　　return getLoadNames().stream()

　　.anyMatch(storedName::equals);

　　}

　　经过上述步骤最终重构后的代码为。

　　public MappedField getMappedField(final String storedName) {

　　return persistenceFields.stream()

　　.filter(mf -> mf.hasName(storedName))

　　.findFirst()

　　.orElse(null);

　　}

　　如需要返回Optional包装的对象则需要去掉orElse。

　　public Optional<MappedField> getMappedField(final String storedName) {

　　return persistenceFields.stream()

　　.filter(mf -> mf.hasName(storedName))

　　.findFirst();

　　}

**总结**

　　这类代码特征通常为：

　　存在深层的循环或条件判断嵌套。

　　需要通过多个getter方法来访问对象内部数据。

　　重构方法：

　　考虑tell don’t ask原则，提供专用的方法供外部调用访问数据，而不是通过使用者经过多次访问去获取对象数据。并通过stream提供的操作来完成重构。