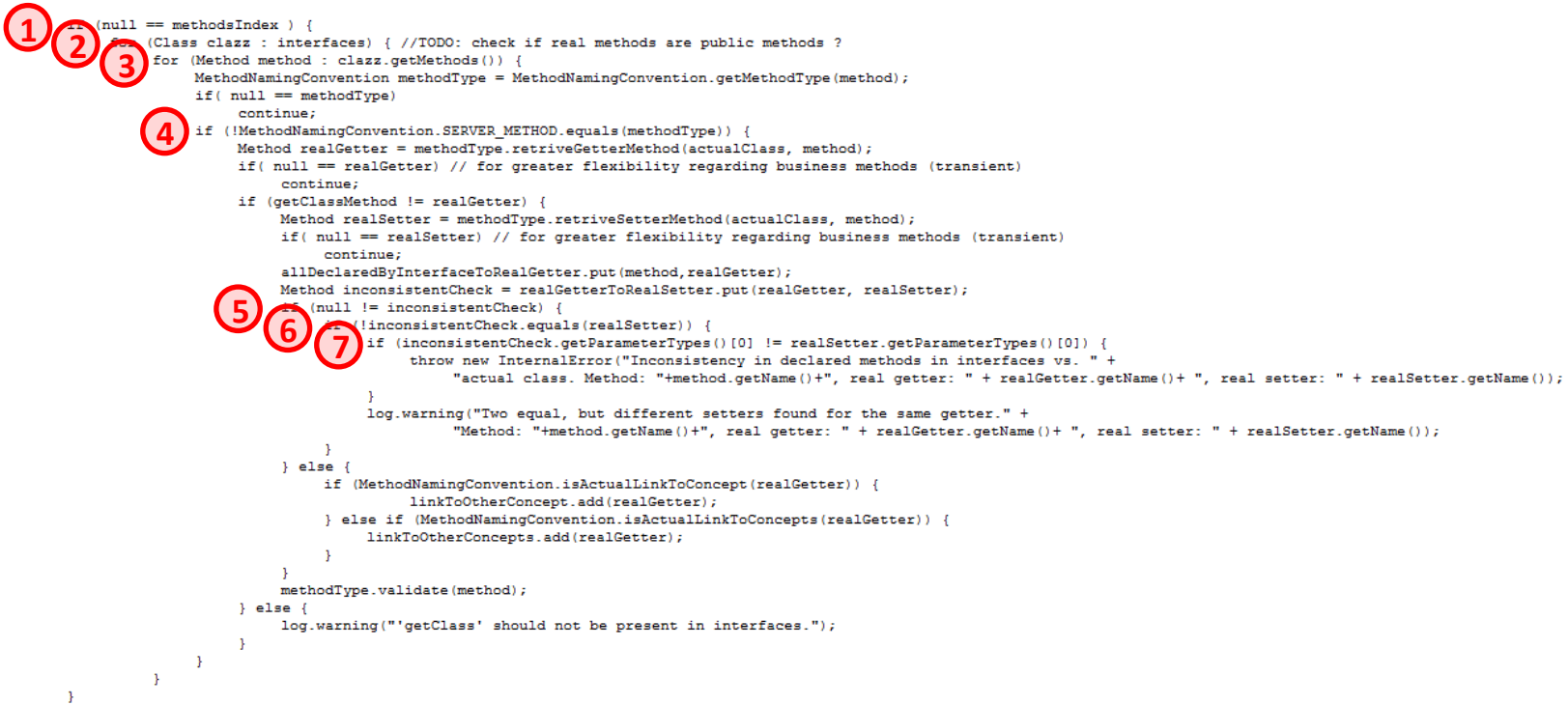


```
if (null == methodsIndex ) {
    for (Class clazz : interfaces) { //TODO: check if real methods are public methods ?
        for (Method method : clazz.getMethods()) {
            MethodNamingConvention methodType = MethodNamingConvention.getMethodType(method);
            if( null == methodType)
                continue;
            if (!MethodNamingConvention.SERVER_METHOD.equals(methodType)) {
                Method realGetter = methodType.retriveGetterMethod(actualClass, method);
                if( null == realGetter) // for greater flexibility regarding business methods (transient)
                    continue;
                if (getClassMethod != realGetter) {
                    Method realSetter = methodType.retriveSetterMethod(actualClass, method);
                    if( null == realSetter) // for greater flexibility regarding business methods (transient)
                        continue;
                    allDeclaredByInterfaceToRealGetter.put(method,realGetter);
                    Method inconsistentCheck = realGetterToRealSetter.put(realGetter, realSetter);
                    if (null != inconsistentCheck) {
                        if (!inconsistentCheck.equals(realSetter)) {
                            if (inconsistentCheck.getParameterTypes()[0] != realSetter.getParameterTypes()[0]) {
                                throw new InternalError("Inconsistency in declared methods in interfaces vs. " +
                                        "actual class. Method: "+method.getName()+", real getter: " + realGetter.getName()+ ", real setter: " + realSetter.getName());
                            }
                            log.warning("Two equal, but different setters found for the same getter." +
                                    "Method: "+method.getName()+", real getter: " + realGetter.getName()+ ", real setter: " + realSetter.getName());
                        }
                    } else {
                        if (MethodNamingConvention.isActualLinkToConcept(realGetter)) {
                                linkToOtherConcept.add(realGetter);
                        } else if (MethodNamingConvention.isActualLinkToConcepts(realGetter)) {
                            linkToOtherConcepts.add(realGetter);
                        }
                    }
                    methodType.validate(method);
                } else {
                    log.warning("'getClass' should not be present in interfaces.");
                }
            }
        }
    }
}
```