KIV/OPSWI

Propojení CoCA-Ex a JaCC

Jméno a příjmení: Luboš Müller
Osobní číslo: A13N0025K
Email: mullelub@students.zcu.cz

Obsah

1. Zadání 3
2. Analýza problému a návrh řešení 3
2.1 Volání JaCC API 3
2.2 Vytvořit propojení JaCC a CoCA-Ex 4
2.3 Přidat informaci o nekompatibilitě do grafu 4
2.3 Rozšířit CoCA-Ex o možnost uložení diagramu 5
3. Programové zpracování 5
4. Instalace a použití 9
5. Závěr 10

1. Zadání

Úkolem je vytvořit nástroj, který bude umět nahrát jar soubory na server, zavolat nad nimi JaCC a získat tak informace o nekompatibilitě. Tyto informace pak vhodně reprezentovat pro následný přenos na web klienta a zobrazit nekompatibilitu v nástroji CoCA-Ex, tj. upravit vzhled CoCA-Ex nástroje. Dále bude program rozšířen o možnost uložení diagramu.

2. Analýza problému a návrh řešení

CoCA-Ex (Complex Component Applications Explorer) pracuje s platformou ComAV (Component Application Visualizer), která tvoří model aplikace a nástroj CoCA-Ex vizualizuje data ve webovém prohlížeči.

Cílem JaCC je vytvořit kompletní reprezentaci Java tříd a run-time kontrolu této reprezentace. Pomocí JaCC API lze prozkoumat kompatibilitu jar souborů a v nich obsažených třídách až na úroveň jednotlivých funkcí a jejich parametrů.

Vizualizace kompatibility byla rozdělena do jednotlivých úkolů.

2.1 Volání JaCC API

Hlavní části JaCC API pro zjištění kompatibility jsou následující:

```
ApiInterCompatibilityChecker<File> checker =
ApiCheckersFactory.getApiInterCompatibilityChecker();


ApiInterCompatibilityResult comparisonResult =
checker.checkInternalCompatibility(app, lib);
```

- Parametr *app* – je množina aplikačních souborů (JAR soubor), může hodnotu null
- Parametr *lib* – je množina knihovních souborů (JAR soubory)

Získaný výsledek je traverzován následujícím způsobem:

Načtení nekompatibilních JAR souborů:

```
Set<String> origins = comparisonResult.getIncompatible();
```

Načtení detailu nekompatibility:

```
JClass incompatibleClass = comparisonResult.getIncompatible(origin);


ApiInterCmpResult apiCmpResult =
comparisonResult.getIncompatible(incompatibleClass, origin);
```

Získání stromu s detailem nekompatibility:

```
CmpResult<JClass> result = apiCmpResult.getResult();
```

Proměnná *result* obsahuje strom s informací o nekompatibilitě strukturované jako prvky Javy – výsledek tedy obsahuje třídu s jejími metodami, poli, konstruktory apod.

2.2 Vytvořit propojení JaCC a CoCA-Ex

Po prozkoumání zdrojových souborů CoCA-Ex se jako nejlepší místo pro propojení s JaCC jeví package *cz.zcu.kiv.offscreen.graph.creator* s třídou *GraphMaker*.

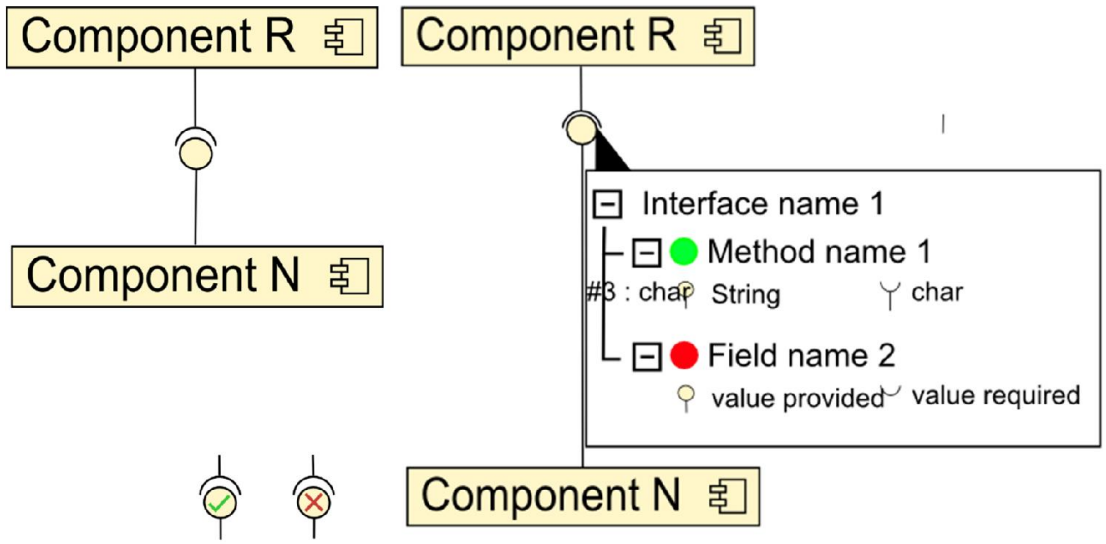
Protože informace o nekompatibilitě se ve výsledku týká hran v grafu, tak možné využít funkce *createEdges()*, kde se bude kontrolovat kompatibilita mezi jednotlivými uzly, které jsou hranou spojeny.

Bude nutné rozšířit třídu *EdgeImpl* v balíku *cz.zcu.kiv.offscreen.graph* o informaci o nekompatibilitě.

2.3 Přidat informaci o nekompatibilitě do grafu

Nekompatibilit musí být v grafu jednoduše rozpoznatelná. Nekompatibilní uzly proto budou mít na hraně zobrazený „červený křížek“ a kompatibilní „zelenou fajfku“. Po kliknutí na hranu bude možné zobrazit detailní informace.

Grafický návrh:



2.3 Rozšířit CoCA-Ex o možnost uložení diagramu

Webová aplikace prozatím nedisponuje možností uložení diagramu v jakékoliv grafické formě. Protože diagram je implementován jako SVG, tak bude možné využít funkce *toDataURL()* HTML elementu *canvas*. Funkce *toDataURL()* vrací URI obsahující reprezentaci obrazu ve formátu určeném svými parametry. Lze tedy vytvořit například obraz ve formátu PNG, který bude vhodný jako výstup z CoCA-Ex.

3. Programové zpracování

Kontrola kompatibility je provedena ve třídě *GraphMaker* balíku *cz.zcu.kiv.offscreen.graph.creator*.

Nejprve se vytvoří uzly grafu z názvů importovaných JAR souborů metoda *generateVertices()*. Hrany grafu jsou rozděleny na ty s informací o nekompatibilitě a na hranu, která vede do uzlu NOT_FOUND a obsahuje informace o nenalezených třídách. Hrany grafu generuje funkce *createEdges()*.

Pro traverzování stromu s informacemi o nekompatibilitě a vytvoření JSON stringu byla vytvořena funkce *findCompatibilityCause*, kterou lze volat rekurzivně.

```
/**
     * Recursive function for traversing tree with incompatibility information
     * Creates JSON string
     *
     * @param children
     * @param className
     * @param jarName
     * @param corrStrategy
     */
    public void findCompatibilityCause(List<CmpResultNode> children, String className,
String jarName, String corrStrategy) {
…
}
```

Další úpravy CoCA-Ex se týkají webové aplikace.

V *graphManager.js* byla upravena funkce *buildGraph()* přidáním fajfky nebo křížku na hrany v grafu podle toho, jestli jsou komponenty kompatibilní:

```
if(edge.isCompatible) {
                        svgBuff += '<line class="lollipop-tick" x1="6" y1="-4" x2="-4"
y2="6" transform="rotate(' + (-lollipop.angle) + ',0,0) translate(0,0)" id="lollipop-
tick_a_' + edge.id + '"></line>';
                        svgBuff += '<line class="lollipop-tick" x1="-5" y1="-3" x2="-4"
y2="5"  transform="rotate(' + (-lollipop.angle) + ',0,0) translate(0,0)" id="lollipop-
tick_b_' + edge.id + '"></line>';
                    } else {
                        svgBuff += '<line class="lollipop-cross" x1="-5" y1="-5" x2="5"
y2="5" transform="rotate(' + (-lollipop.angle) + ',0,0) translate(0,0)" id="lollipop-
cross_a_' + edge.id + '"></line>';
                        svgBuff += '<line class="lollipop-cross" x1="-5" y1="5" x2="5"
y2="-5" transform="rotate(' + (-lollipop.angle) + ',0,0) translate(0,0)" id="lollipop-
cross_b_' + edge.id + '"></line>';
                    }
```

Do *tooltips.js* byly přidány funkce pro zpracování JSON objektu s informací o nekompatibilitě. Pro tooltip, který se zobrazí po kliknutí na hranu grafu je vytvořen vnořený seznam.

```
/**
 * Return HTML for incompatibility tooltip
 */
function getCompatibilityInfo(data) {

    compatibilityTooltip = "";
    for (var i = 0; i < data.length; i++) {
        var Class = data[i];
        compatibilityTooltip += "<li><strong>" + Class.theClass + "</strong><ul>";
        for (var j = 0; j < Class.incomps.length; j++) {
            if (Class.incomps[j] &&  Class.incomps[j].subtree.length) {
                parseCompatibilityInfo(Class.incomps[j]);
            }
        }
        compatibilityTooltip += "</ul></li>";
    }
    compatibilityTooltip += "";
    return compatibilityTooltip;

}

/**
 * Traverses incompatibility JSON object and creates HTML nested list
 */
function parseCompatibilityInfo(data) {
    if(data.desc.isIncompCause == "true") {
        compatibilityTooltip += "<li><strong class=\"incomp\">" + data.desc.incompName
+ "</strong>";
        compatibilityTooltip += "<ul class=\"compatibility-list\">";

        if (data.desc.difference != "DEL") {
            compatibilityTooltip += "<li><span><img
src=\"images/efp_qtip/provided.png\"> <span class=\"second\">" +
data.desc.objectNameSecond + "</span></span></li>";
            compatibilityTooltip += "<li><span><img
src=\"images/efp qtip/required.png\"> <span class=\"first\">" +
data.desc.objectNameFirst + "</span></span></li>";
        }
        compatibilityTooltip += "</ul>";
    } else {
        if (data.desc.level > 0) {
            compatibilityTooltip += "<li><strong>" + data.desc.name + "</strong>";
        }
    }

    if (data.subtree.length) {
        if (data.desc.level > 0) {
            compatibilityTooltip += "<ul class=\"compatibility-list\">";
        }
        for (var i = 0; i < data.subtree.length; i++) {
            if (data.subtree[i].subtree.length || data.subtree[i].desc.isIncompCause ==
"true") {
                parseCompatibilityInfo(data.subtree[i]);
            }
        }
        if (data.desc.level > 0) {
            compatibilityTooltip += "</ul>";
        }
    }
    if (data.desc.level > 0) {
        compatibilityTooltip += "</li>";
    }
}
```

Menší změny týkající se zobrazení tooltipu byly provedeny také v CSS (*basic.css*).

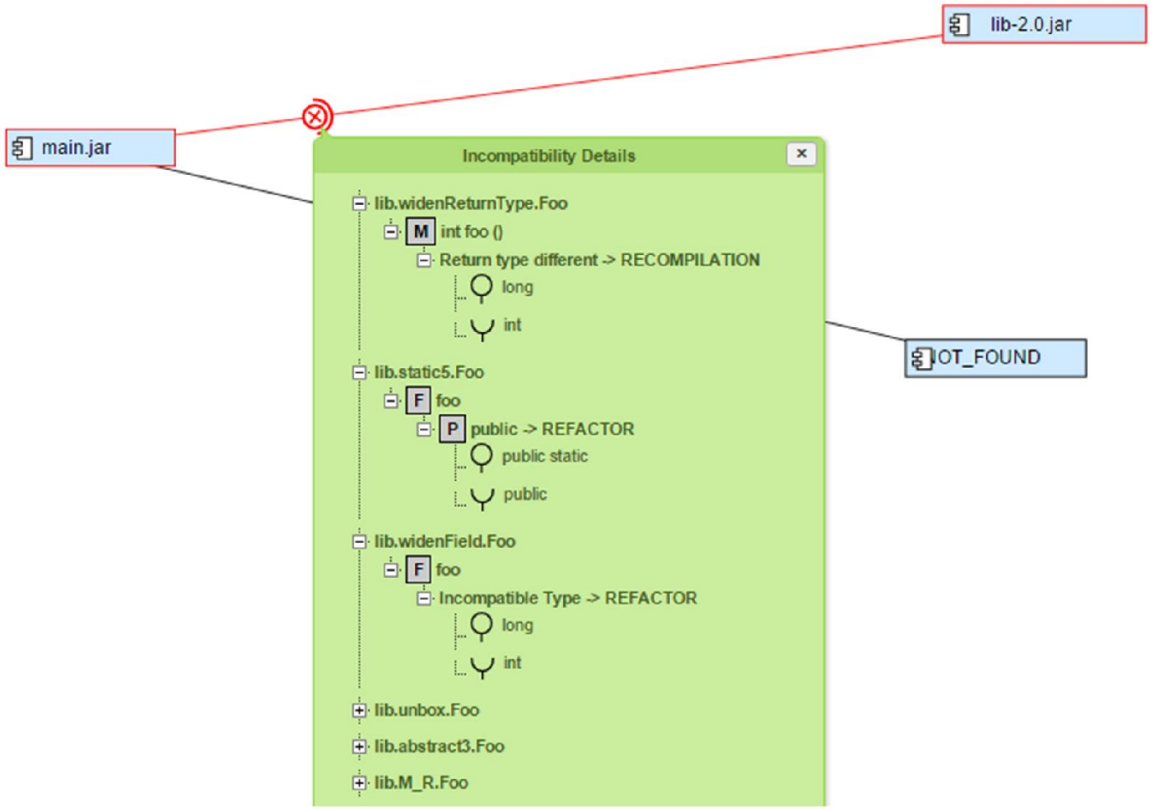
Jako poslední byla implementována možnost uložení diagramu do PNG. Pro tento účel byl využit open-source script saveSvgAsPng dostupný na https://github.com/exupero/saveSvgAsPng

Dialog pro uložení PNG se vyvolá po kliknutí na ikonu diskety:

Uložení diagramu je optimalizováno pro prohlížeče Chrome a Firefox. V prohlížeči Internet Explorer nemusí uložení fungovat kvůli SecurityError.

V případě vzniku chyby při generování grafu je chyba vypsána do konzole prohlížeče. To usnadňuje následné dubuggování.

Ukázka výstupu (tooltip se ve výsledném PNG nezobrazuje):



4. Instalace a použití

Kromě zdrojových kódů je součástí práce také WAR archiv, který lze nasadit standardním způsobem např. na server Apache Tomcat. Podporovaná verze je veze JDK 7.

Rozdíl oproti původnímu Co-CA-Ex je pouze v tom, že se po nahrání JAR souborů na server nemusí vybírat typ frameworku. Další práce s Co-CA-Ex se jinak neliší od původní verze.

5. Závěr

Podle zadání byl nástroj CoCA-Ex upraven tak, aby bylo možné zobrazit informace o nekompatibilitě komponent. Pro získání informací o nekompatibilitě byl použit JaCC. Součástí zadání bylo také rozšíření CoCA-Ex o možnost uložení diagramu. Tento úkol byl také splněn.

Nejsložitější byla analýza zdrojových kódů a určení míst v programu, která bylo nutné upravit pro splnění zadání. Menší problémy byly s částí volání JaCC API, které nemá dokumentaci. Proto bylo třeba projít celé API a zkoušet jednotlivé funkce a jejich návratové hodnoty. Implementace úprav webového klienta vycházela z JSON struktur zavedených při implementaci úprav back-endu, nebyla proto obzvlášť obtížná.

Celkově hodnotím práci na propojení CoCA-Ex a JaCC jako přínos. Pochopil jsem, jak CoCA-Ex funguje, a obnovil jsem si znalosti z programování v Javě. Jako příležitost do budoucna vidím úpravu GUI web klienta.