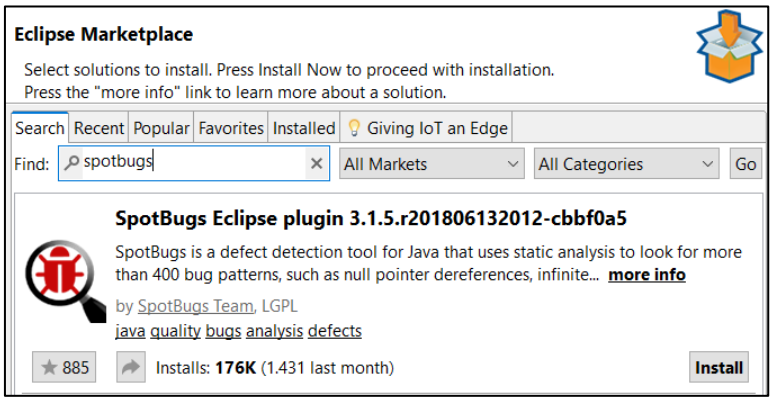
TESTING E VERIFICA DEL SOFTWARE

SPOTBUGS TUTORIAL

Ing. Piazzi Simona

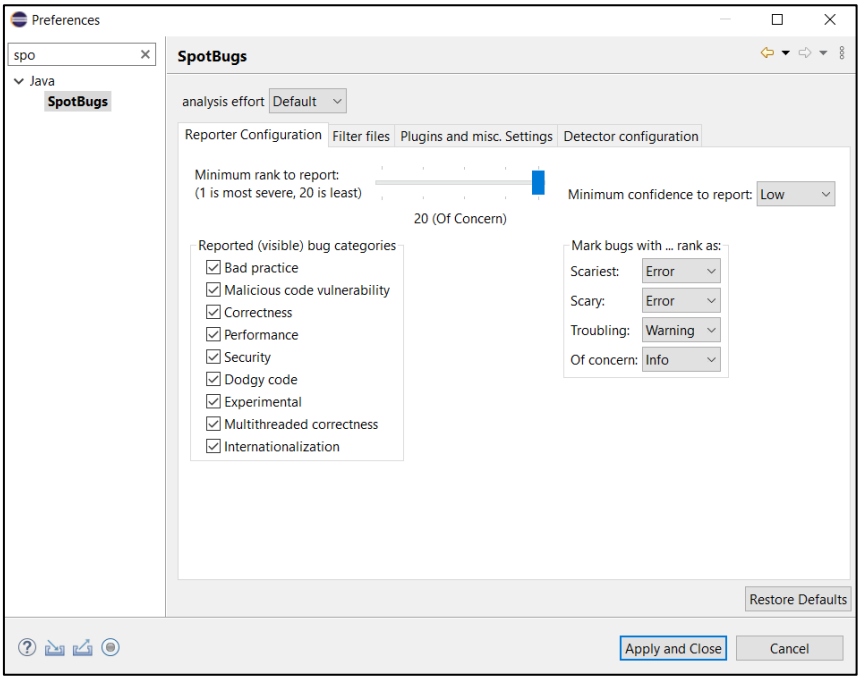
Setup

Per Eclipse è disponibile un plugin per SpotBugs che si può installare tramite Eclipse Marketplace.



Cliccare “Install”, accettare i termini della licenza e cliccare “Finish”.

Una volta terminata l’installazione, riavviare Eclipse. Infine, impostare i seguenti parametri per SpotBugs, seguendo il percorso Window – Preferences - SpotBugs e poi cliccare “Apply and Close”



Esempi e spiegazione

SpotBugs è un tool che permette di eseguire un'analisi statica sul codice Java per individuare eventuali bug. Ciascun bug ha un proprio livello di priorità e appartiene ad una specifica categoria. La documentazione (https://spotbugs.readthedocs.io/en/latest/bugDescriptions.html) fornisce una spiegazione per ciascuno di questi bug e suggerisce un'eventuale soluzione.

Per attivare SpotBugs bisogna cliccare il tasto destro sul nome del progetto o sulla classe che si vuole analizzare e selezionare SpotBugs – SpotBugs dal menù a tendina. Se viene rilevato qualche bug apparirà una piccola icona a lato della riga di codice interessato.

Tuttavia, per avere un'analisi più dettagliata del problema, è possibile generare un file XML con un'ampia descrizione per ciascun bug rilevato. Per farlo bisogna cliccare con il tasto destro sul progetto o sulla classe considerata e selezionare SpotBugs – Open Analysis Results in Editor.

```
28  <BugInstance type="ES_COMPARING_STRINGS_WITH_EQ" priority="3" rank="14" abbrev="ES" category="BAD_PRACTICE" first="1">
29    <Class classname="Main">
30      <SourceLine classname="Main" sourcefile="Main.java" sourcepath="Main.java"/>
31    </Class>
32    <Method classname="Main" name="main" signature="([Ljava/lang/String;)V" isStatic="true">
33      <SourceLine classname="Main" start="5" end="21" startBytecode="0" endBytecode="39" sourcefile="Main.java" sourcepath="Main.java"/>
34    </Method>
35    <Type descriptor="Ljava/lang/String;" role="TYPE_FOUND">
36      <SourceLine classname="java.lang.String" start="140" end="4653" sourcefile="String.java" sourcepath="java/lang/String.java"/>
37    </Type>
38    <LocalVariable name="winSequence" register="2" pc="82" role="LOCAL_VARIABLE_VALUE_OF"/>
39    <LocalVariable name="temp" register="3" pc="82" role="LOCAL_VARIABLE_VALUE_OF"/>
40    <SourceLine classname="Main" start="10" end="10" startBytecode="83" endBytecode="83" sourcefile="Main.java" sourcepath="Main.java"/>
41    <SourceLine classname="Main" start="10" end="10" startBytecode="83" endBytecode="83" sourcefile="Main.java" sourcepath="Main.java"/>
42    <Property name="edu.umd.cs.findbugs.detect.RefComparisonWarningProperty.STATIC_AND_UNKNOWN" value="true"/>
```

Ciascun bug sarà segnalato dal tag *BugInstance* e sarà identificato da un tipo che può essere utilizzato come chiave di ricerca nella documentazione. Nel caso specifico illustrato nella figura, la segnalazione è dovuta alla presenza di un'errata comparazione di stringhe. Il problema è situato nella classe Main.java alla riga 10 e le due variabili che vengono confrontate erroneamente sono *winSequence* e *temp*, come si può leggere nei tag *LocalVariable* e *SourceLine*.

```
10          while(temp != winSequence) {
```

Come si nota dal codice le due stringhe vengono confrontate con l'operatore != anziché con il metodo appropriato, ovvero equals(Object).

```
44  <BugInstance type="DM_NEXTINT_VIA_NEXTDOUBLE" priority="2" rank="18" abbrev="Dm" category="PERFORMANCE" first="1">
45    <Class classname="Match">
46      <SourceLine classname="Match" sourcefile="Match.java" sourcepath="Match.java"/>
47    </Class>
48    <Method classname="Match" name="generator" signature="()I" isStatic="false">
49      <SourceLine classname="Match" start="20" end="20" startBytecode="0" endBytecode="50" sourcefile="Match.java" sourcepath="Match.java"/>
50    </Method>
51    <SourceLine classname="Match" start="20" end="20" startBytecode="7" endBytecode="7" sourcefile="Match.java" sourcepath="Match.java"/>
52    <SourceLine classname="Match" start="20" end="20" startBytecode="7" endBytecode="7" sourcefile="Match.java" sourcepath="Match.java"/>
53  </BugInstance>
```

```
20          return (int) (Math.random() * 9);
```

In questo esempio, invece, viene segnalato un problema di ottimizzazione delle performance. La linea in questione prevede la generazione di un numero casuale tramite un cast di un valore double. Il tool suggerisce invece un'istruzione che genera direttamente il valore intero desiderato.