# Applied Static Analysis

Software Technology Group
Department of Computer Science
Technische Universität Darmstadt
Dr. Michael Eichberg

# Simple Data-flow Analysis

You should use `MyOPALProject` as a template. That project is preconfigured to use the latest snapshot version of OPAL. You can clone the project using:
`git clone --depth 1 https://bitbucket.org/OPAL-Project/myopalproject Project`

An integrated JavaDoc of the latest snapshot version of OPAL that spans all subprojects can be found at: www.opal-project.de

For further details regarding the development of static analysis using OPAL see the OPAL tutorial.

You should develop both of the following analyses on top of the 3-address code representation (TACAI) offered by OPAL. Use the `l1.DefaultDomainWithCFGAndDefUse` domain and the `ProjectInformationKey` `ComputeTACAIKey` as the foundation for your analysis.

## Use Arrays.equals

Develop an analysis which finds violations of the following rule taken from The CERT Oracle Secure Coding Standard for Java:

> EXP02-J: Use the two-argument Arrays.equals() method to compare the contents of arrays.

Non-compliant example:

```
int[] a1 = new int[]{0};
int[] a2 = new int[]{0};
a1.equals(a2); // <= FALSE (performs a reference comparison)
```

Compliant example:

```
int[] a1 = new int[]{0};
int[] a2 = new int[]{0};
Arrays.equals(a1,a2); // <= TRUE (compares the content)
```

Recall that arrays are objects and that it is therefore possible to call those methods (e.g., wait, notify and equals) on arrays which are defined by `java.lang.Object`. Furthermore, the declared receiver of the call will be the class type `java.lang.Object`.

***Tasks***

1. Test your analysis using the class `ArraysEquals`.
2. Run your analysis against the JDK

## BigDecimal and Floating Point Literals

Develop an analysis which finds violations of the following rule taken from The CERT Oracle Secure Coding Standard for Java:

> NUM10-J: Do not construct BigDecimal objects from floating-point literals.

Non-compliant example:

```
new BigDecimal(1.0f);
```

Compliant example:

```
new BigDecimal("1.0");
```

***Tasks***

1. How does the bytecode change, when you exchange the floating-point literal `1.0f` (a float literal) against the floating-point literal `1.0d` $(a double literal).
2. Test your analysis using the class `BigDecimalAndStringLiteral`.
3. Run your analysis against the JDK.

## Closeables

Develop an **interprocedural** analysis which finds violations of the following rule taken from The CERT Oracle Secure Coding Standard for Java:

> FIO04-J: Close resources when they are not longer needed.

Here, we consider as a resource every instance of the class `java.lang.AutoCloseable`.

Non-compliant example:

```
public int processFile(String fileName) throws Exception {
  FileInputStream stream = new FileInputStream(fileName);
  BufferedReader bufRead = new BufferedReader(new InputStreamReader(stream));
  String line;
  while ((line = bufRead.readLine()) != null) {
    sendLine(line);
  }
  return 1;
}
```

Compliant example:

```
try (FileInputStream stream = new FileInputStream(fileName);
     InputStreamReader reader = new InputStreamReader(stream);
     BufferedReader bufRead = new BufferedReader(reader)) {
  String line;
  while ((line = bufRead.readLine()) != null) {
    sendLine(line);
```

```
    }
  } catch (IOException e) {
      log(e);
  }
```

To reduce the number of false positives only apply this check if the resource object is created inside the same method and ...:
- not passed to some other method and not stored in a field (i.e., the resource object may be garbage collected at the end of the method), or
- the method is closed at least on one path.

For example, ignore the following cases where the stream is passed in as a parameter:

```
public int processFile(FileInputStream stream) throws Exception {
  BufferedReader bufRead = new BufferedReader(new InputStreamReader(stream));
  String line;
  while ((line = bufRead.readLine()) != null) {
    sendLine(line);
  }
  return 1;
}
```

**Tasks**

1. Test your analysis using the class `Closeables`. It should not produce any false positives.
2. Run your analysis against the JDK.