checkstyle-code-arguable

VariableDeclarationUsageDistance

```
<module name="Checker">
  <module name="TreeWalker">
    <module name="VariableDeclarationUsageDistance"/>
  </module>
</module>
```

```
public void foo1() {
    // violation below, 'variable 'num' declaration and its first usage is 4.'
    int num;
    final double PI;   // ok, final variables not checked
    System.out.println("Statement 1");
    System.out.println("Statement 2");
    System.out.println("Statement 3");
    num = 1;
    PI = 3.14;
  }
```

https://checkstyle.sourceforge.io/checks/coding/variabledeclarationusagedistance.html

OverloadMethodsDeclarationOrder

```
<module name="Checker">
  <module name="TreeWalker">
    <module name="OverloadMethodsDeclarationOrder"/>
  </module>
</module>
```

```
class Example1 {
  void same(int i) {}
  // comments between overloaded methods are allowed.
  void same(String s) {}
  void same(String s, int i) {}
  void same(int i, String s) {}
  void notSame() {}
  interface notSame{}

  void otherSame(String s) {}
  void foo() {}
  // violation below, 'All overloaded methods should be placed next to each other'
```

```
  void otherSame(String s, int i) {}
}
```

https://checkstyle.sourceforge.io/checks/coding/overloadmethodsdeclarationorder.html#OverloadMethodsDeclarationOrder

BooleanExpressionComplexity

```
<module name="Checker">
  <module name="TreeWalker">
    <module name="BooleanExpressionComplexity"/>
  </module>
</module>
```

```
public class Example1
{
  public static void main(String ... args)
  {
    boolean a = true;
    boolean b = false;

    boolean c = (a & b) | (b ^ a); // ok, 1(&) + 1(|) + 1(^) = 3 (max allowed 3)

    boolean d = (a & b) | (b ^ a) | (a ^ b);
    // violation above, 'Boolean expression complexity is 5 (max allowed is 3)'
    // 1(&) + 1(|) + 1(^) + 1(|) + 1(^) = 5

    boolean e = a ^ (a || b) ^ (b || a) & (a | b);
    // violation above, 'Boolean expression complexity is 6 (max allowed is 3)'
    // 1(^) + 1(||) + 1(^) + 1(||) + 1(&) + 1(|) = 6
  }
}
```

https://checkstyle.sourceforge.io/checks/metrics/booleanexpressioncomplexity.html#BooleanExpressionComplexity

BooleanExpressionComplexity

```
<module name="BooleanExpressionComplexity">
  <property name="max" value="3"/>
</module>
```

```
public class Example1
{
  public static void main(String ... args)
  {
    boolean a = true;
    boolean b = false;
```

```
    boolean c = (a & b) | (b ^ a); // ok, 1(&) + 1(|) + 1(^) = 3 (max allowed 3)

    boolean d = (a & b) | (b ^ a) | (a ^ b);
    // violation above, 'Boolean expression complexity is 5 (max allowed is 3)'
    // 1(&) + 1(|) + 1(^) + 1(|) + 1(^) = 5

    boolean e = a ^ (a || b) ^ (b || a) & (a | b);
    // violation above, 'Boolean expression complexity is 6 (max allowed is 3)'
    // 1(^) + 1(||) + 1(^) + 1(||) + 1(&) + 1(|) = 6
  }
}
```

https://checkstyle.sourceforge.io/checks/metrics/booleanexpressioncomplexity.html#BooleanExpressionComplexity

MagicNumber

```
<module name="MagicNumber">
  <property name="ignoreNumbers" value="0,1,2,3,4,5,6,7,8,9,10,100,1000"/>
  <property name="ignoreHashCodeMethod" value="true"/>
  <property name="ignoreAnnotation" value="true"/>
</module>
```

```
@Example1.Annotation(6) // violation, ''6' is a magic number.'
public class Example1 {
  private int field = 7; // violation, ''7' is a magic number.'

  void method1() {
    int i = 1;
    int j = 8; // violation, ''8' is a magic number.'
  }
}
```

https://checkstyle.sourceforge.io/checks/coding/magicnumber.html#MagicNumber