

```
@Override
public int visualizationLegend(int arg1, String arg2, SomeClass arg3)
throws MyException
{
    int local1;
    local1 = 42;
    String local2 = arg2;

    if (arg1 >= 5) {
        System.out.println("Arg1 is greater or equal 4");
        return 1;
    } else if (arg1 < 0) {
        System.out.println("Arg1 is negative");

        while (arg1 < 0) {
            ++arg1;
            arg3.foo(arg1, local2);
        }
    } else {
        assert(arg1 >= 0 && arg1 < 5);
        System.out.print("Arg1 is within the proper range.");
    }

    int local3 = arg3.bar(this, local1);
    for(int i = 0; i < local3; i++) {
        if (i - 10 % 5 == 0) continue;
        for (int k = i; k > 0; k--) {
            System.out.println(i);
            System.out.println(k);
        }
    }

    synchronized(this) {
        arg3.foobar(arg1);
        if (arg3.isInvalid())
            throw new MyException("State is invalid");
    }

    try {
        SomeClass sc = new SomeClass();
        Object baz_result = sc.baz(local1, this, null, 0, arg2);
        local1 = ((Cell)baz_result).value;
    }
    catch(RuntimeException e) {
        System.out.println("Run-time exception");
    }
    catch(Exception e) {
        System.out.println("General exception exception");
    }
    finally {
        arg3.cleanup();
    }

    switch(local1) {
        case 0:
            System.out.println("10");
            break;
        case 1:
            System.out.println("20");
            return 20;
        default:
            System.out.println("Something else");
    }

    return 0;
}
```

```

method int visualizationLegend arg1 int arg2 String arg3 SomeClass

  @Override()
  throws MyException

  -----
  int local1
  local1 ← 42
  String local2 ← arg2

  if arg1 ≥ 5
    System.out.println("Arg1 is greater or equal 4")
    return 1
  else if arg1 < 0
    System.out.println("Arg1 is negative")

    loop arg1 < 0
      ++arg1
      arg3.foo(arg1 local2)
    -----
    assert (arg1 ≥ 0 ∧ arg1 < 5)
    System.out.print("Arg1 is within the proper range.")

  int local3 ← arg3.bar(this local1)
  loop int i ← 0 i < local3 i++
    if i - 10 % 5 = 0
      continue
    loop int k ← i k > 0 k--
      System.out.println(i)
      System.out.println(k)

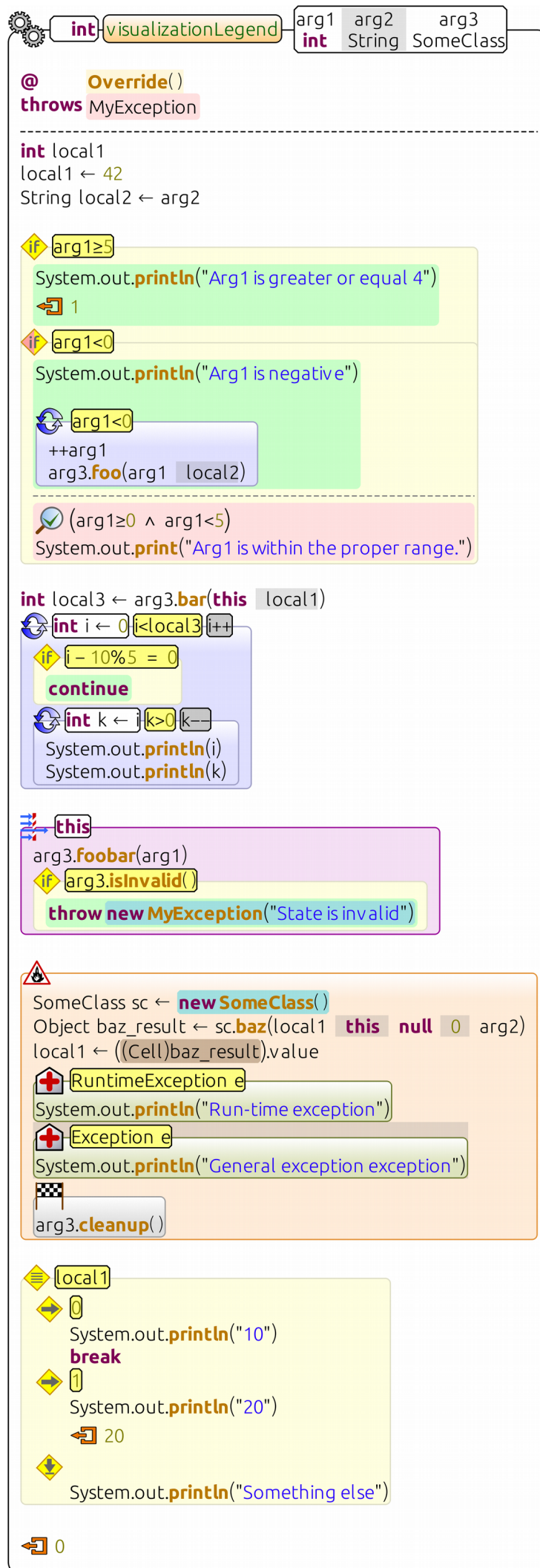
  synchronized this
    arg3.foobar(arg1)
    if arg3.isInvalid()
      throw new MyException("State is invalid")

  try
    SomeClass sc ← new SomeClass()
    Object baz_result ← sc.baz(local1 this null 0 arg2)
    local1 ← ((Cell)baz_result).value
    catch RuntimeException e
      System.out.println("Run-time exception")
    catch Exception e
      System.out.println("General exception exception")
    finally
      arg3.cleanup()

  switch local1
    case 0
      System.out.println("10")
      break
    case 1
      System.out.println("20")
      return 20
    default
      System.out.println("Something else")














  return 0

```



Comparison between *Eclipse*, *Envision reduced*, and *Envision default*

Differences:

Feature	Eclipse	Envision reduced	Envision default
block/body	{ }	outlines	
method icon	none	method	
method parameters	(String x, int y)	<div> <div>x</div> <div>String</div> <div>y</div> <div>int</div> </div>	
signature end/ body start	{ }	-----	
local variable	SomeType foo;	SomeType foo	
assignment	=	←	
comparison	==	=	
operators	<= >= != && !	≤ ≥ ≠ ∧ ∨ ¬	
semicolon	;	no semicolon	
lists (e.g. method args)	a, b, c, d	a b c d	
method/constructor call	foo(a, b);	foo(a b)	foo(a b)
object creation	new Foo(x);	new Foo(x)	new Foo(x)
cast	(String) foo	(String) foo	(String) foo
return	return x;	return x	 x
assert	assert x;	assert x	 x
if	if	if	 background
else if	else if	else if	 background
then branch	{ }	white background	background
else branch	else { }	-----	----- background
loops	for while do	loop	 background
try	try	try	 background
catch	catch	catch	 background
finally	finally	finally	 background
synchronized	synchronized	synchronized	 background
switch	switch	switch	 background
case	case	case	 background
default case	default	default	 background

No differences:

Feature	Eclipse	Envision reduced	Envision default
null	null	null	
this	this	this	
throw	throw	throw	
break, continue	break continue	break continue	
operators	+ - * / ++ --	+ - * / ++ --	