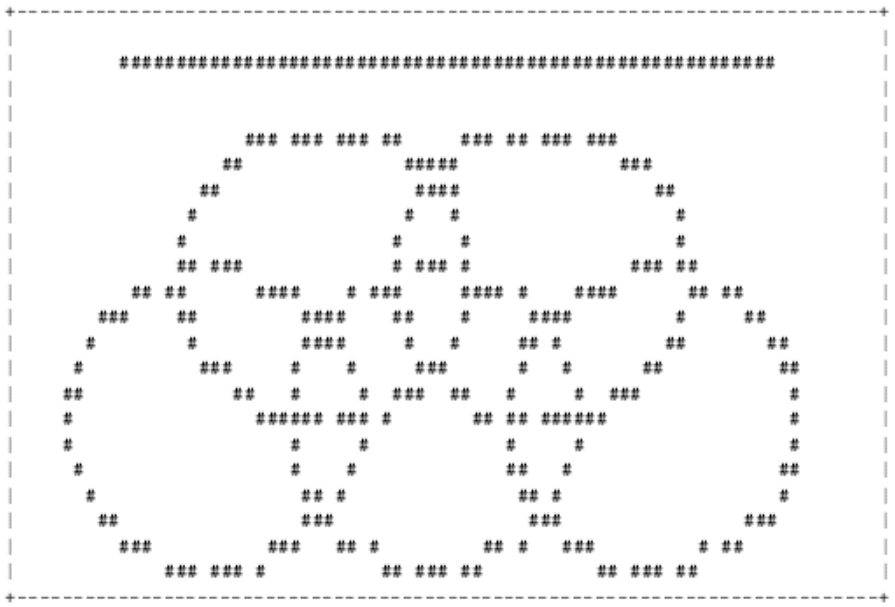
Questions: Design Pattern Version

In order to complete each task, you need to fill **NAME , START TIME and END TIME** for each task.

Program: Graphics Library

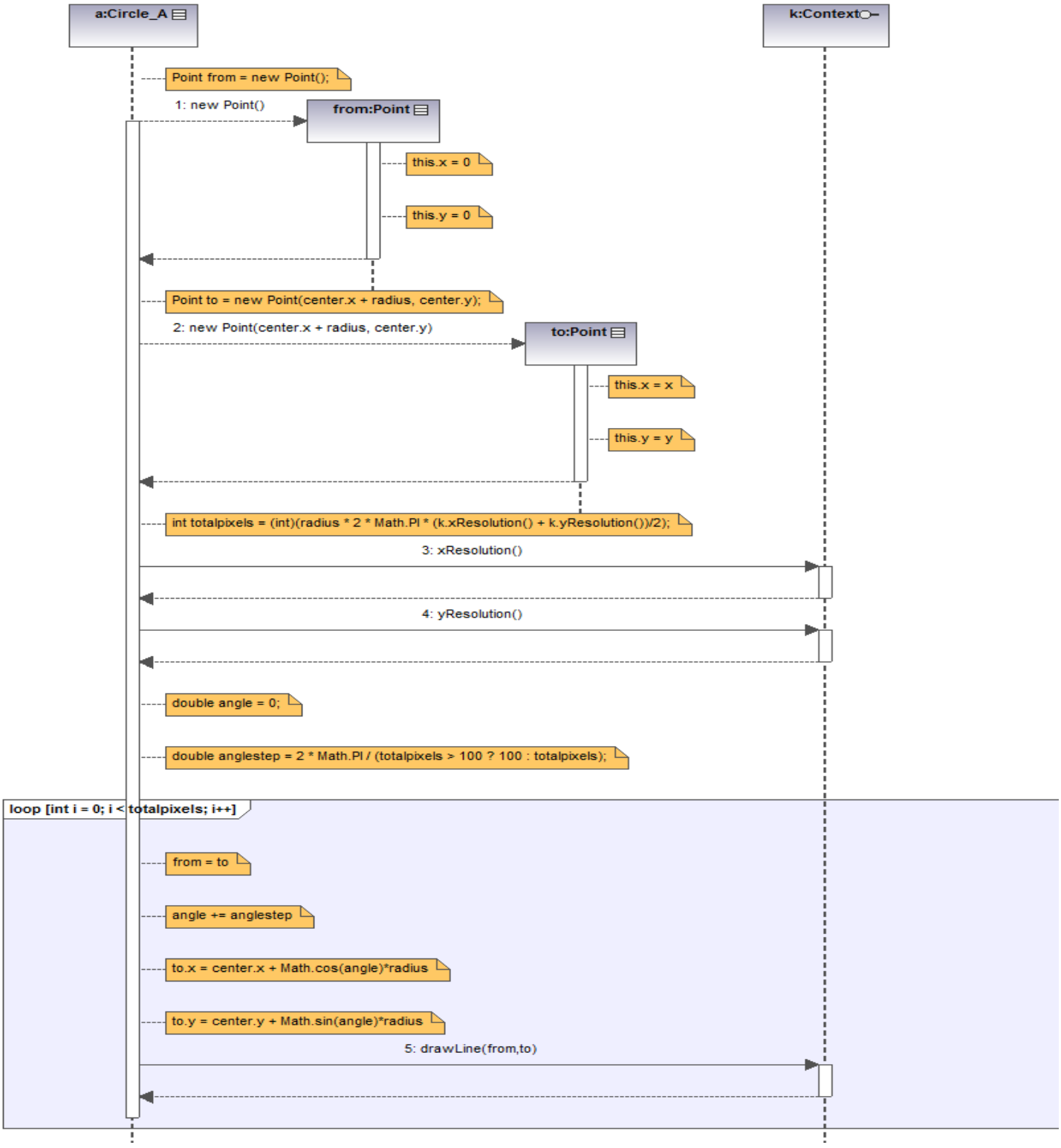
Description

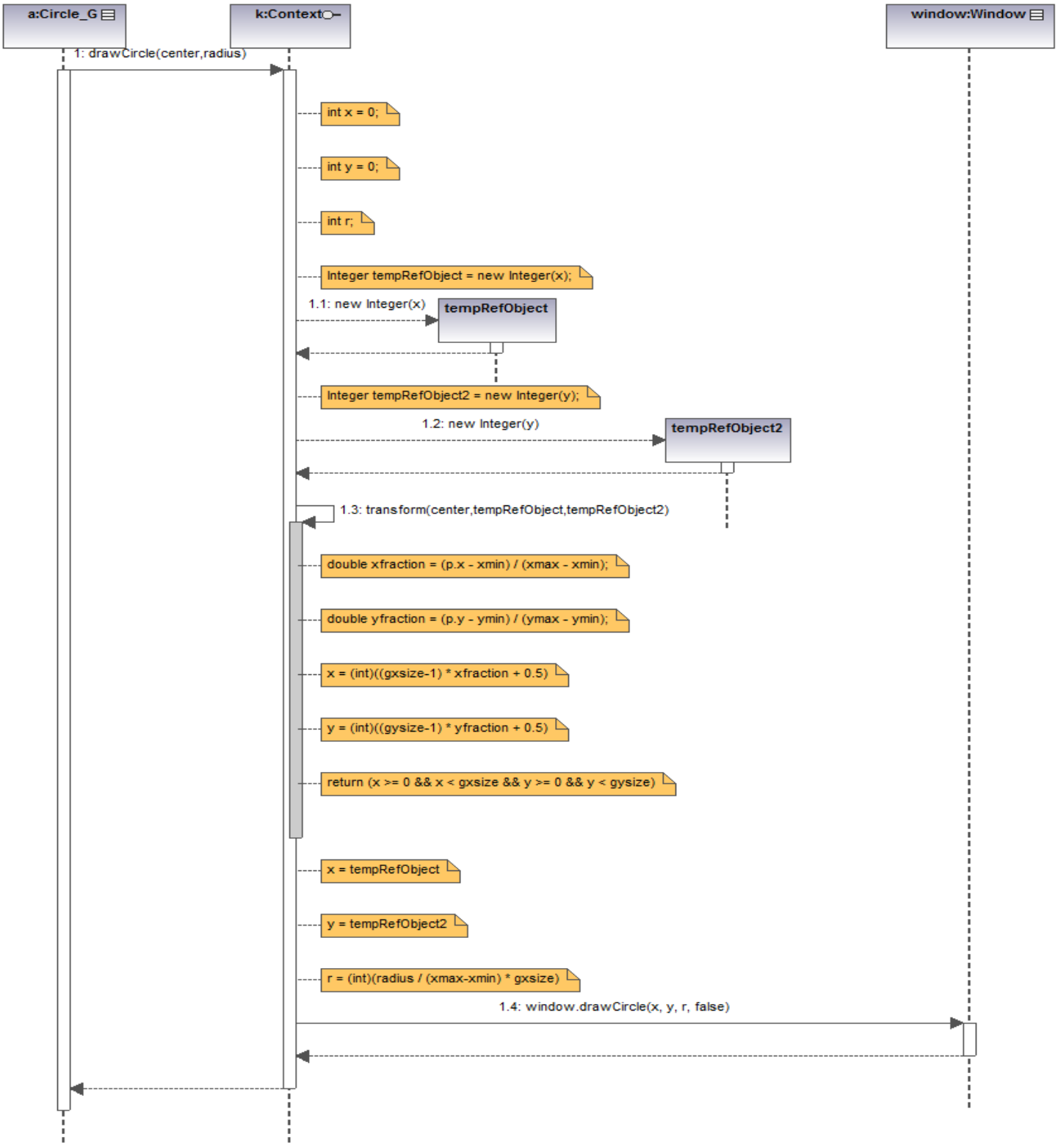
You will receive a "graphics library" design. It contains a library for creating, manipulating, and drawing simple types of graphical objects (*Line, Circle*) on different types of graphical devices (alpha-numerical output *A*, pixel graphic device *G*). Furthermore it contains a simple main program that uses some of the objects and operations. It displays the result shown below.

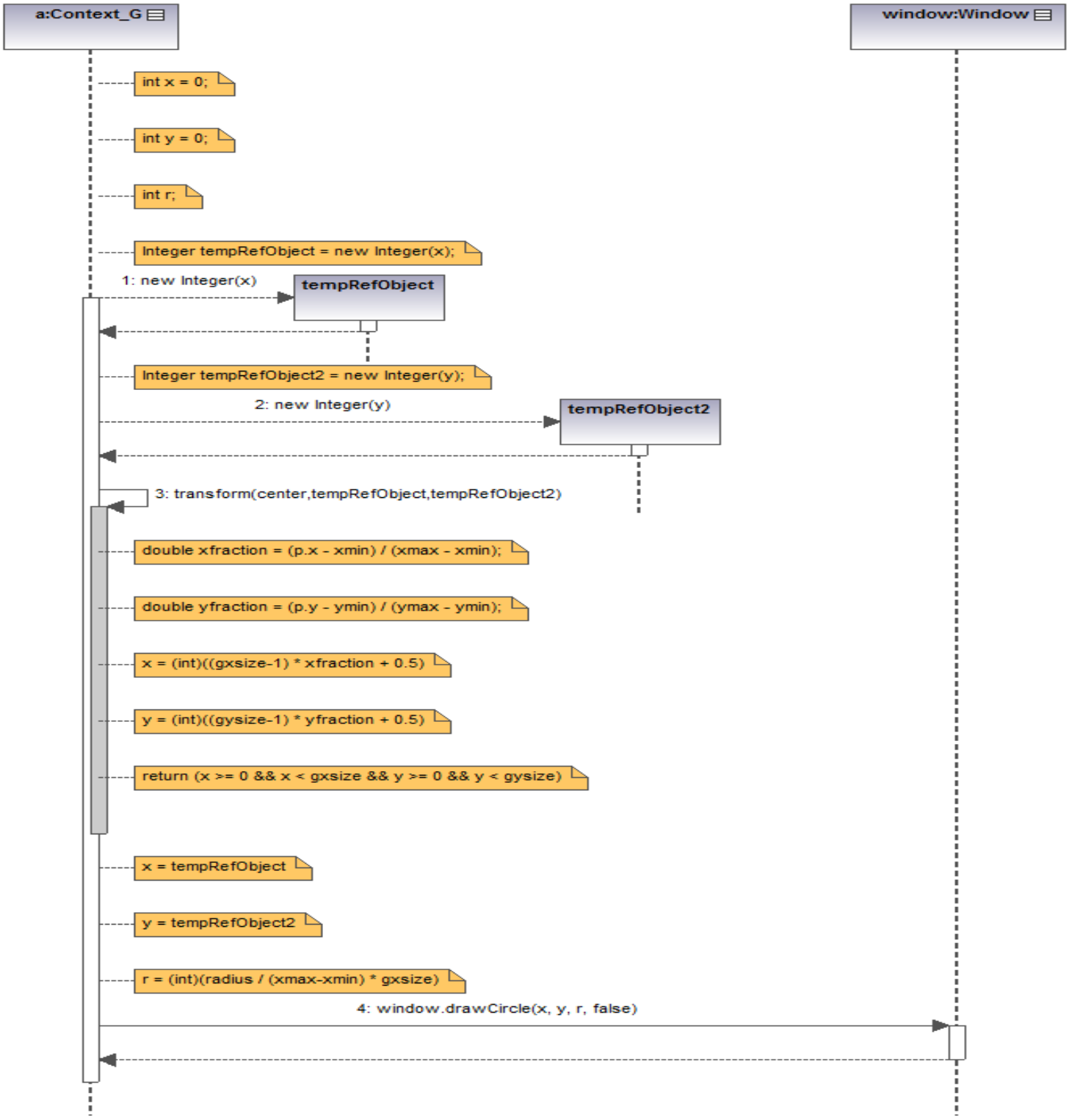


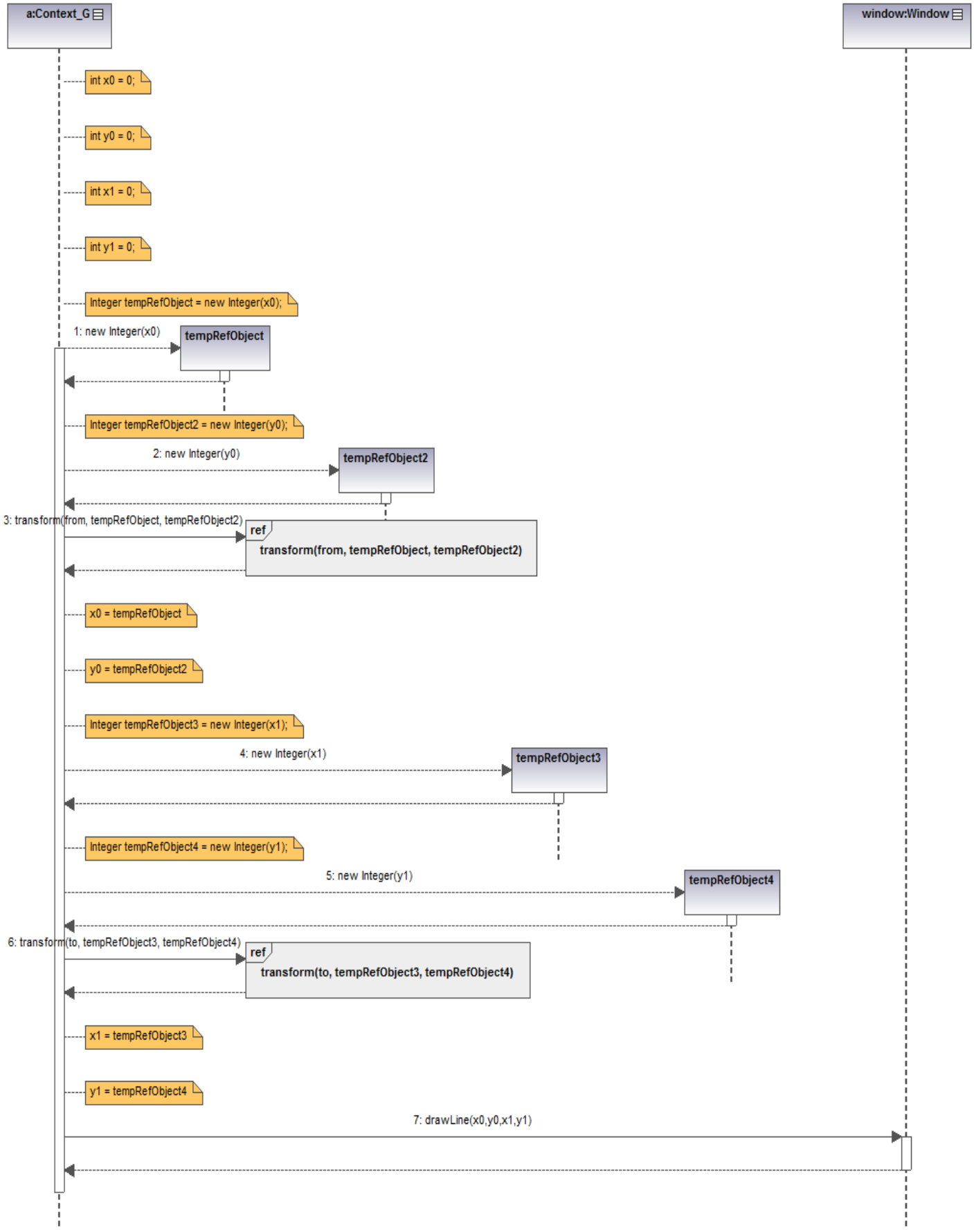
A collection of classes (*Context*) implements the primitive operations for different output devices. There is a minimal set of operations but some devices provide additional functionality. For this reason some objects (in this case *Line*) are implemented only once for all devices but other objects (in this case *Circle*) are implemented individually for each device. Which device is to be used can be selected at a central spot *Generator*.

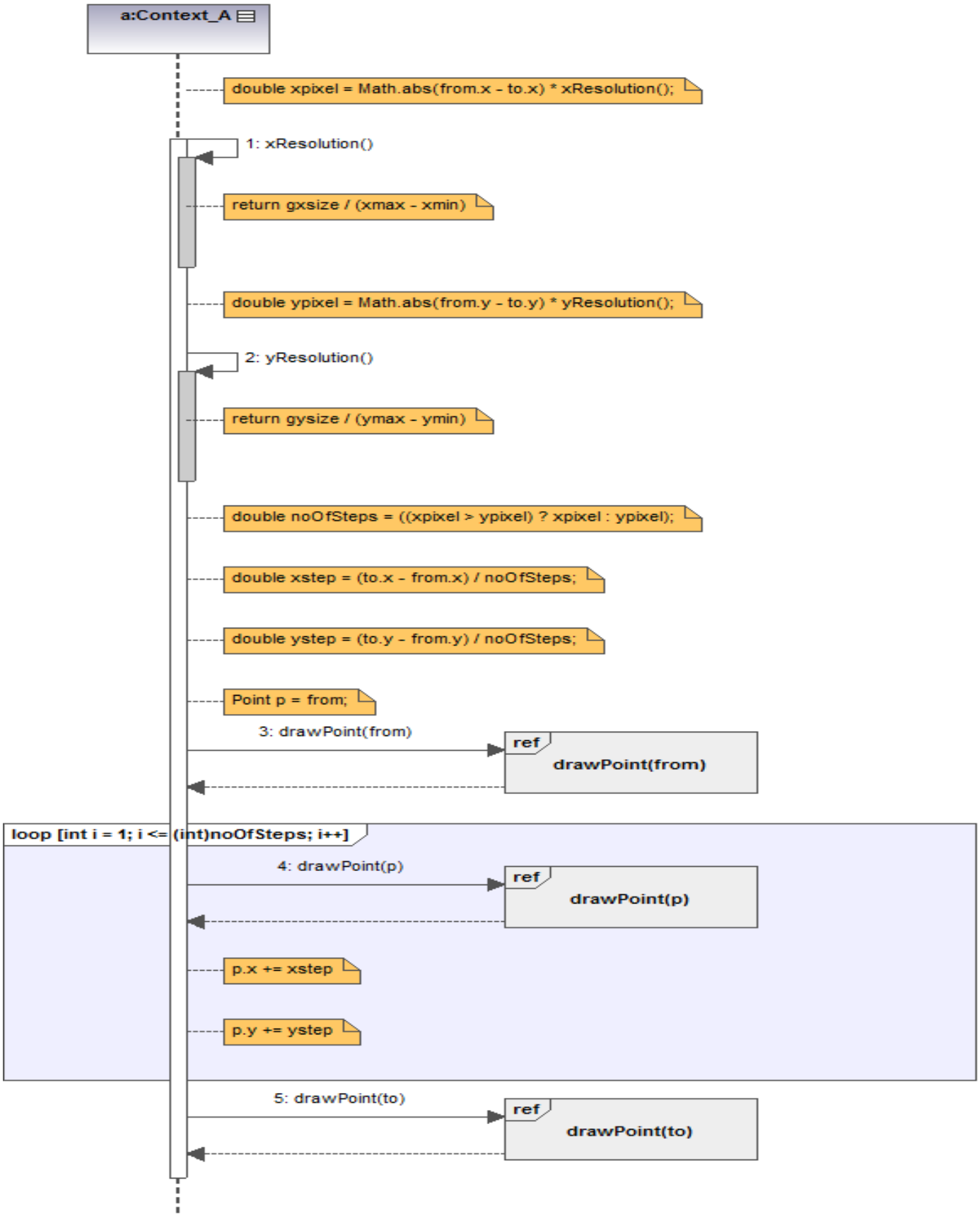

Context_A
gsize:int
gsize:int
drawingarea:char[][]
xmin:double
xmax:double
ymin:double
ymax:double
«construct» Context_A()
«construct» Context_A(in gsize:int, in gsize:int, in xmin:double, in ymin:double, in xmax:double, in ymax:double)
xResolution():double
yResolution():double
erase():void
drawPoint(in p:Point):void
drawLine(in fromPoint, in to:Point):void
display():void
«interface»
Context
erase():void
xResolution():double
yResolution():double
drawPoint(in NamelessParameter:Point):void
drawLine(in NamelessParameter1:Point, in NamelessParameter2:Point):void
display():void
testrun
main(in args:String[]):void
create(in generator:Generator):void
Context_G
gsize:int
gsize:int
xmin:double
xmax:double
ymin:double
ymax:double
window:Window
transform(in p:Point, in x:Integer, in y:Integer):boolean
«construct» Context_G()
«construct» Context_G(in gsize:int, in gsize:int, in xmin:double, in ymin:double, in xmax:double, in ymax:double)
xResolution():double
yResolution():double
erase():void
drawPoint(in p:Point):void
drawLine(in fromPoint, in to:Point):void
display():void
drawCircle(in centerPoint, in radius:double):void
«interface»
GraphicalObject
rotate(in centerPoint, in angle:double):void
shift(in dx:double, in dy:double):void
draw(in k:Context):void
Line
start:Point=new Point()
end:Point=new Point()
«construct» Line()
«construct» Line(in start:Point, in end:Point)
rotate(in centerPoint, in angle:double):void
shift(in dx:double, in dy:double):void
draw(in k:Context):void
Point
x:double
y:double
«construct» Point()
«construct» Point(in x:double, in y:double)
rotate(in centerPoint, in angle:double):void
shift(in dx:double, in dy:double):void
-start
-end
-element
*
#center
ObjectGroup
limit:int=100
element:GraphicalObject[]=new GraphicalObject[limit]
noOfElements:int
«construct» ObjectGroup()
add(in k:GraphicalObject):void
rotate(in centerPoint, in angle:double):void
shift(in dx:double, in dy:double):void
draw(in k:Context):void
Circle
centerPoint=new Point()
radius:double
«construct» Circle()
«construct» Circle(in centerPoint, in radius:double)
rotate(in centerPoint, in angle:double):void
shift(in dx:double, in dy:double):void
draw(in k:Context):void
Window
bold:int=1
italics:int=2
underline:int=4
black:int=8
white:int=1
gray:int=2
createFrame(in title:String, in width:int, in height:int, in storepixels:boolean):Window
fontHeight(in str:String):int
fontWidth(in str:String):int
height():int
width():int
clear():void
drawCircle(in xC:int, in yC:int, in radius:int, in fillet:boolean):void
drawLine(in x0:int, in y0:int, in x1:int, in y1:int):void
drawPoint(in x:int, in y:int):void
drawString(in str:char, in xpos:int, in ypos:int):void
fontAttributes(in attributes:int, in set:boolean):int
resize(in xsize:int, in ysize:int):void
setColor(in foreground:int, in background:int):void
setFont(in fontname:char, in pointsize:int):void
shiftContents(in xshift:int, in yshift:int):void
-window
Circle_A
«construct» Circle_A()
«construct» Circle_A(in centerPoint, in radius:double)
draw(in k:Context):void
Circle_G
«construct» Circle_G()
«construct» Circle_G(in centerPoint, in radius:double)
draw(in k:Context):void
AlphanumericalGenerator
newContext():Context
newContext(in gsize:int, in gsize:int, in xmin:double, in ymin:double, in xmax:double, in ymax:double):Context
newObjectgroup():ObjectGroup
newLine(in fromPoint, in to:Point):Line
newCircle(in centerPoint, in radius:double):Circle
Generator
newGenerator(in s:GeneratorType):Generator
newContext():Context
newContext(in gsize:int, in gsize:int, in xmin:double, in ymin:double, in xmax:double, in ymax:double):Context
newObjectgroup():ObjectGroup
newLine(in fromPoint, in to:Point):Line
newCircle(in centerPoint, in radius:double):Circle
GeneratorType
GraphicalGenerator
newContext():Context
newContext(in gsize:int, in gsize:int, in xmin:double, in ymin:double, in xmax:double, in ymax:double):Context
newObjectgroup():ObjectGroup
newLine(in fromPoint, in to:Point):Line
newCircle(in centerPoint, in radius:double):Circle

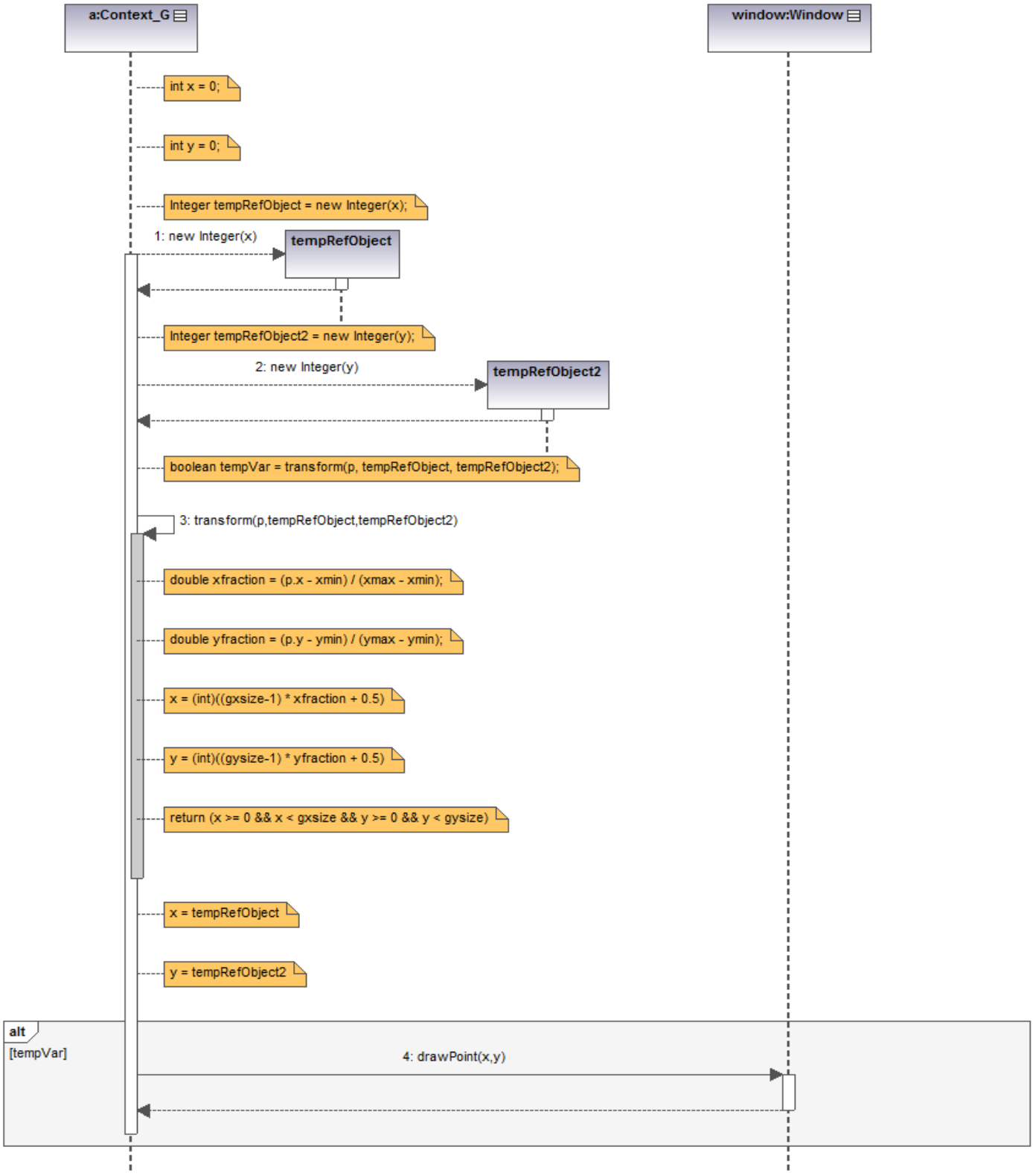

a:Circle_A
k:Context
Point from = new Point();
1: new Point()
from:Point
this.x = 0
this.y = 0
Point to = new Point(center.x + radius, center.y);
2: new Point(center.x + radius, center.y)
to:Point
this.x = x
this.y = y
int totalpixels = (int)(radius * 2 * Math.PI * (k.xResolution() + k.yResolution())/2);
3: xResolution()
4: yResolution()
double angle = 0;
double anglestep = 2 * Math.PI / (totalpixels > 100 ? 100 : totalpixels);
loop [int i = 0; i < totalpixels; i++]
from = to
angle += anglestep
to.x = center.x + Math.cos(angle)*radius
to.y = center.y + Math.sin(angle)*radius
5: drawLine(from,to)

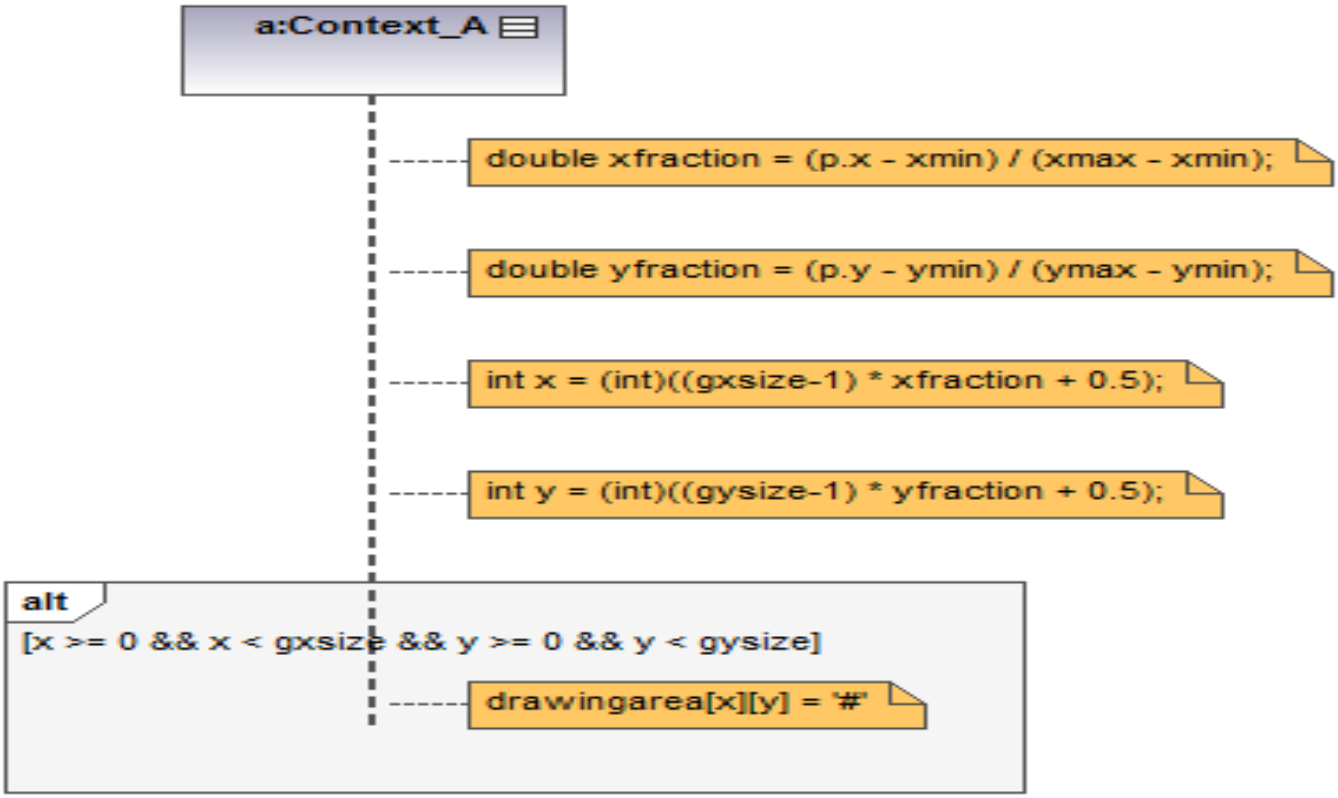

a:Circle_G
k:Context
window:Window
1: drawCircle(center,radius)
int x = 0;
int y = 0;
int r;
Integer tempRefObject = new Integer(x);
1.1: new Integer(x)
tempRefObject
Integer tempRefObject2 = new Integer(y);
1.2: new Integer(y)
tempRefObject2
1.3: transform(center,tempRefObject,tempRefObject2)
double xfraction = (p.x - xmin) / (xmax - xmin);
double yfraction = (p.y - ymin) / (ymax - ymin);
x = (int)((gxsize-1) * xfraction + 0.5)
y = (int)((gysize-1) * yfraction + 0.5)
return (x >= 0 && x < gxsize && y >= 0 && y < gysize)
x = tempRefObject
y = tempRefObject2
r = (int)(radius / (xmax-xmin) * gxsize)
1.4: window.drawCircle(x, y, r, false)


a:Context_G
window:Window
int x = 0;
int y = 0;
int r;
Integer tempRefObject = new Integer(x);
1: new Integer(x)
tempRefObject
Integer tempRefObject2 = new Integer(y);
2: new Integer(y)
tempRefObject2
3: transform(center,tempRefObject,tempRefObject2)
double xfraction = (p.x - xmin) / (xmax - xmin);
double yfraction = (p.y - ymin) / (ymax - ymin);
x = (int)((gxsize-1) * xfraction + 0.5)
y = (int)((gysize-1) * yfraction + 0.5)
return (x >= 0 && x < gxsize && y >= 0 && y < gysize)
x = tempRefObject
y = tempRefObject2
r = (int)(radius / (xmax-xmin) * gxsize)
4: window.drawCircle(x, y, r, false)


a:Context_G
window:Window
int x0 = 0;
int y0 = 0;
int x1 = 0;
int y1 = 0;
Integer tempRefObject = new Integer(x0);
1: new Integer(x0)
tempRefObject
Integer tempRefObject2 = new Integer(y0);
2: new Integer(y0)
tempRefObject2
3: transform(from, tempRefObject, tempRefObject2)
ref
transform(from, tempRefObject, tempRefObject2)
x0 = tempRefObject
y0 = tempRefObject2
Integer tempRefObject3 = new Integer(x1);
4: new Integer(x1)
tempRefObject3
Integer tempRefObject4 = new Integer(y1);
5: new Integer(y1)
tempRefObject4
6: transform(to, tempRefObject3, tempRefObject4)
ref
transform(to, tempRefObject3, tempRefObject4)
x1 = tempRefObject3
y1 = tempRefObject4
7: drawLine(x0,y0,x1,y1)


a:Context_A
double xpixel = Math.abs(from.x - to.x) * xResolution();
1: xResolution()
return gxsize / (xmax - xmin)
double ypixel = Math.abs(from.y - to.y) * yResolution();
2: yResolution()
return gysize / (ymax - ymin)
double noOfSteps = ((xpixel > ypixel) ? xpixel : ypixel);
double xstep = (to.x - from.x) / noOfSteps;
double ystep = (to.y - from.y) / noOfSteps;
Point p = from;
3: drawPoint(from)
ref
drawPoint(from)
loop [int i = 1; i <= (int)noOfSteps; i++]
4: drawPoint(p)
ref
drawPoint(p)
p.x += xstep
p.y += ystep
5: drawPoint(to)
ref
drawPoint(to)


a:Context_G
window:Window
int x = 0;
int y = 0;
Integer tempRefObject = new Integer(x);
1: new Integer(x)
tempRefObject
Integer tempRefObject2 = new Integer(y);
2: new Integer(y)
tempRefObject2
boolean tempVar = transform(p, tempRefObject, tempRefObject2);
3: transform(p,tempRefObject,tempRefObject2)
double xfraction = (p.x - xmin) / (xmax - xmin);
double yfraction = (p.y - ymin) / (ymax - ymin);
x = (int)((gxsize-1) * xfraction + 0.5)
y = (int)((gysize-1) * yfraction + 0.5)
return (x >= 0 && x < gxsize && y >= 0 && y < gysize)
x = tempRefObject
y = tempRefObject2
alt
[tempVar]
4: drawPoint(x,y)


a:Context_A
double xfraction = (p.x – xmin) / (xmax – xmin);
double yfraction = (p.y – ymin) / (ymax – ymin);
int x = (int)((gxsize-1) * xfraction + 0.5);
int y = (int)((gysize-1) * yfraction + 0.5);
alt
[x >= 0 && x < gxsize && y >= 0 && y < gysize]
drawingarea[x][y] = '#'

Task 1: Please answer all these questions.

Name: _____

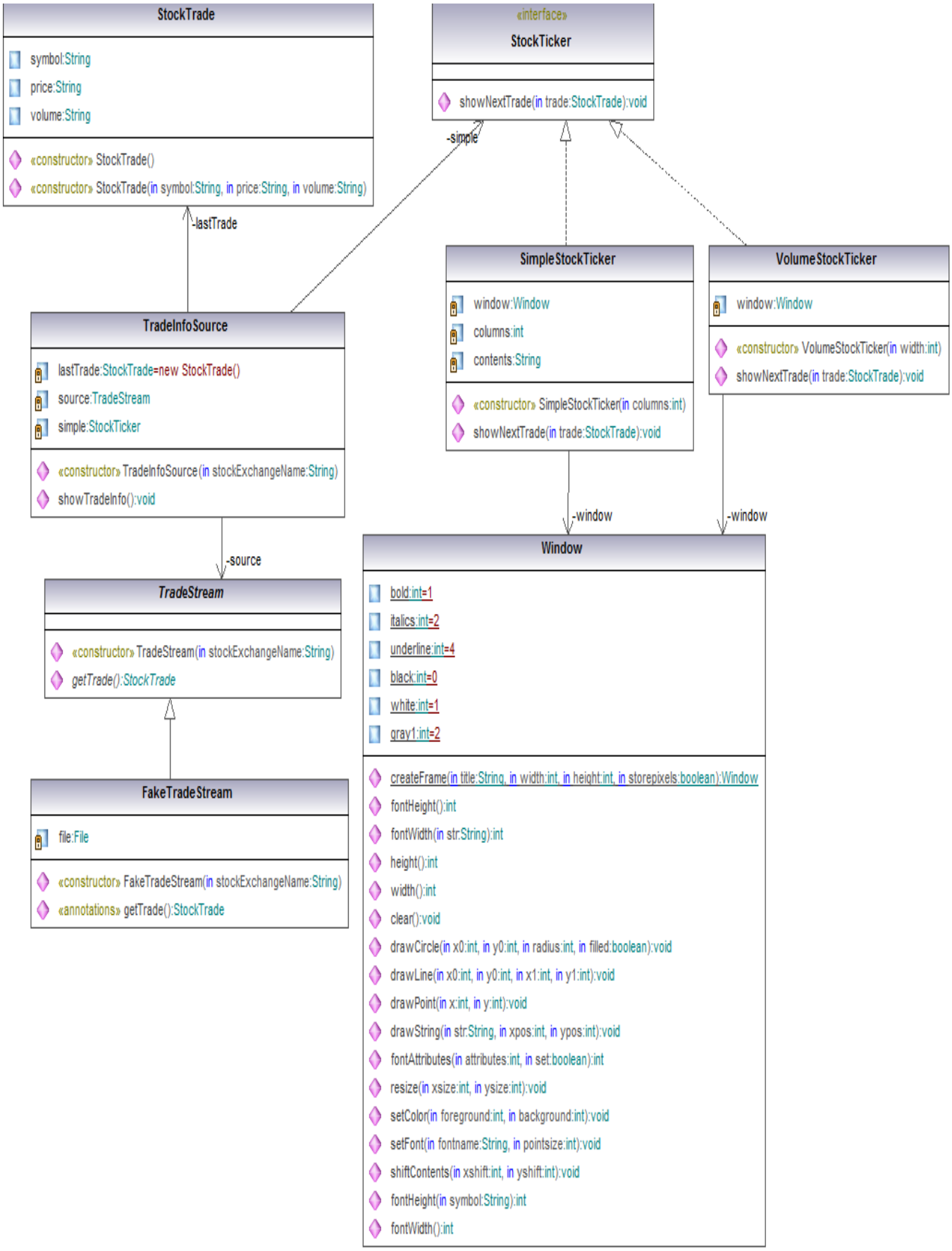
Start Time: ____:____ **(hh : mm)** ** The time that you begin answer the question.

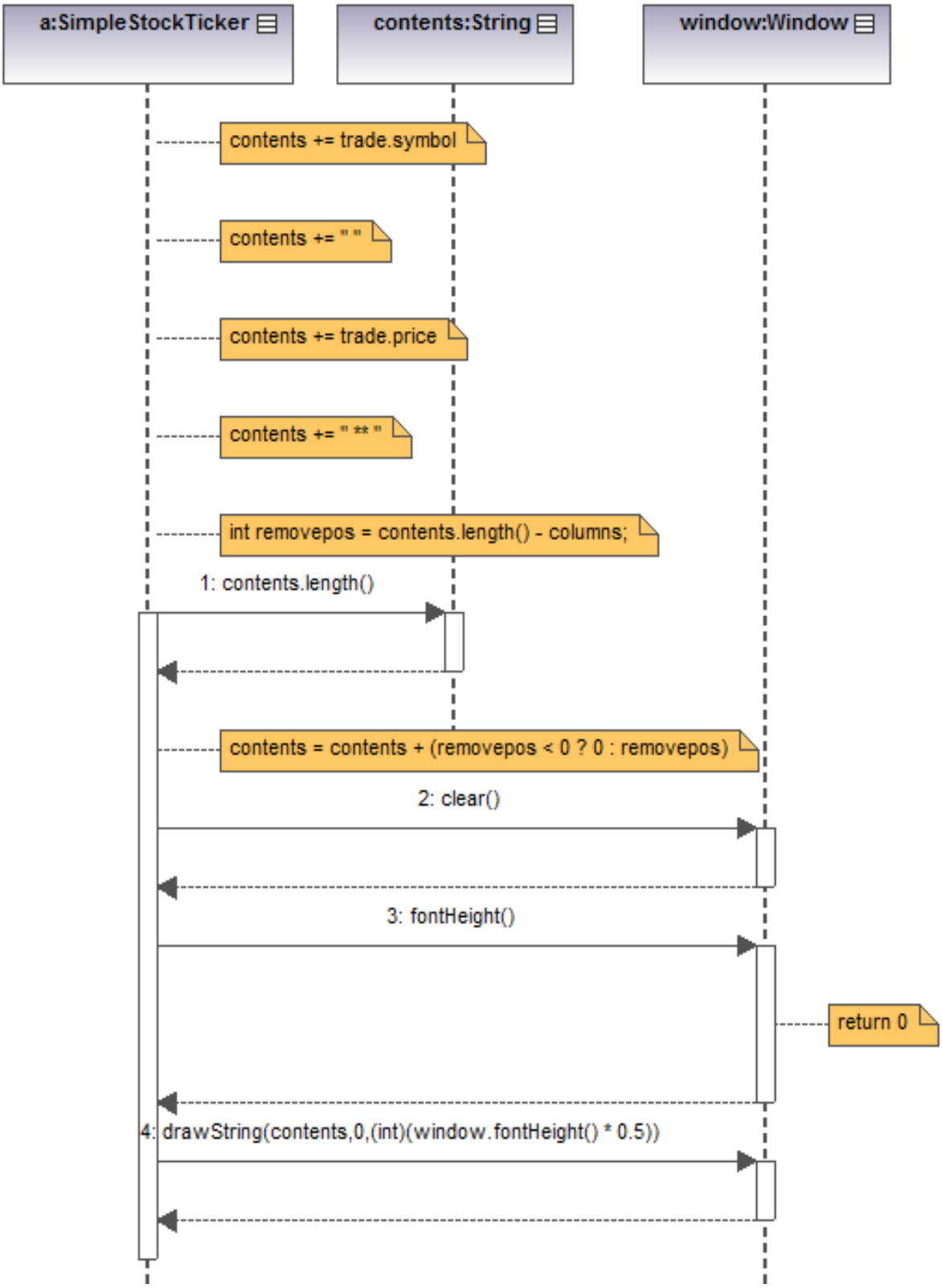
1. Provide all class names that plot the 2D point.
   _____
2. Provide the class name that control types of graphical.
   _____
3. Provide all class names that manipulate the GraphicalObject objects.
   _____
4. From the Question 3, provide all method names that collect the GraphicalObject objects.
   _____
5. Provide a base class of the graphical representation hierarchy.
   _____
6. Provide all class names that the Circle_A class uses to draw a circle.
   _____
7. Provide all class names that the Line class uses to draw a line.
   _____
8. Provide all class names that display circle on device G.
   _____
9. Provide all class names that call the drawCircle method in the Window class.
   _____
10. Provide all class names that invoke the Generator class.
    _____

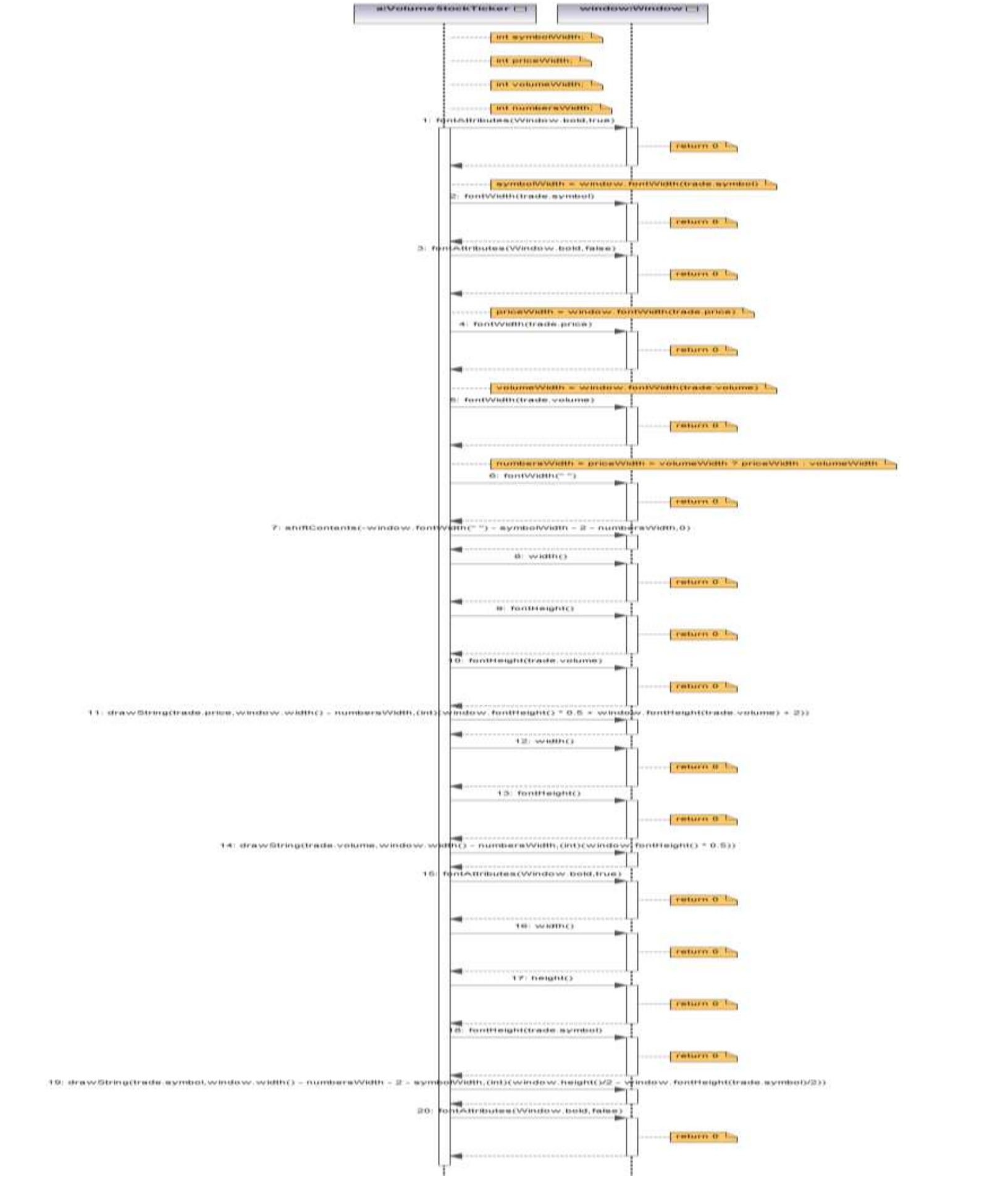
End Time: ____:____ **(hh : mm)** ** The time that you completely answer all questions.

Program: Stock Ticker

Description:
Shortly, you will receive a "stock ticker" design. This incomplete program is used for directing a continuous stream of stock trades (title, number, unit, price) from a stock market to one or more displays, which are also a part of the program. The displays advertise the information or part of it.
The data (of type *StockTrade*) come from a *FakeTradeStream* which simulates the trade datastream by reading the data repeatedly from a file. This is done for simplification.
In the present program version only *one* display shows the stock data. The type of this display is *SimpleStockTicker*. A second display named VolumeStockTicker is implemented but presently not used. Both types of display are shown in the figure below


StockTrade
symbol:String
price:String
volume:String
«constructor» StockTrade()
«constructor» StockTrade(in symbol:String, in price:String, in volume:String)
«interface»
StockTicker
showNextTrade(in trade:StockTrade):void
-simple
-lastTrade
SimpleStockTicker
window:Window
columns:int
contents:String
«constructor» SimpleStockTicker(in columns:int)
showNextTrade(in trade:StockTrade):void
VolumeStockTicker
window:Window
«constructor» VolumeStockTicker(in width:int)
showNextTrade(in trade:StockTrade):void
TradeInfoSource
lastTrade:StockTrade=new StockTrade()
source:TradeStream
simple:StockTicker
«constructor» TradeInfoSource(in stockExchangeName:String)
showTradeInfo():void
-source
-window
-window
TradeStream
«constructor» TradeStream(in stockExchangeName:String)
getTrade():StockTrade
Window
bold:int=1
italics:int=2
underline:int=4
black:int=0
white:int=1
gray1:int=2
createFrame(in title:String, in width:int, in height:int, in storepixels:boolean):Window
fontHeight():int
fontWidth(in str:String):int
height():int
width():int
clear():void
drawCircle(in x0:int, in y0:int, in radius:int, in filled:boolean):void
drawLine(in x0:int, in y0:int, in x1:int, in y1:int):void
drawPoint(in x:int, in y:int):void
drawString(in str:String, in xpos:int, in ypos:int):void
fontAttributes(in attributes:int, in set:boolean):int
resize(in xsize:int, in ysize:int):void
setColor(in foreground:int, in background:int):void
setFont(in fontname:String, in pointsize:int):void
shiftContents(in xshift:int, in yshift:int):void
fontHeight(in symbol:String):int
fontWidth():int
FakeTradeStream
file:File
«constructor» FakeTradeStream(in stockExchangeName:String)
«annotations» getTrade():StockTrade


a:SimpleStockTicker
contents:String
window:Window
contents += trade.symbol
contents += " "
contents += trade.price
contents += " ** "
int removepos = contents.length() - columns;
1: contents.length()
contents = contents + (removepos < 0 ? 0 : removepos)
2: clear()
3: fontHeight()
return 0
4: drawString(contents,0,(int)(window.fontHeight() * 0.5))


a:VolumeStockTicker
window:Window
int symbolWidth;
int priceWidth;
int volumeWidth;
int numbersWidth;
1: fontAttributes(Window.bold,true)
return 0;
symbolWidth = window.fontWidth(trade.symbol);
2: fontWidth(trade.symbol)
return 0;
3: fontAttributes(Window.bold,false)
return 0;
priceWidth = window.fontWidth(trade.price);
4: fontWidth(trade.price)
return 0;
volumeWidth = window.fontWidth(trade.volume);
5: fontWidth(trade.volume)
return 0;
numbersWidth = priceWidth > volumeWidth ? priceWidth : volumeWidth;
6: fontWidth(" ")
return 0;
7: shiftContents(-window.fontWidth(" ") - symbolWidth - 2 - numbersWidth,0)
8: width()
return 0;
9: fontHeight()
return 0;
10: fontHeight(trade.volume)
return 0;
11: drawString(trade.price,window.width() - numbersWidth,(int)(window.fontHeight() * 0.5 + window.fontHeight(trade.volume) + 2));
12: width()
return 0;
13: fontHeight()
return 0;
14: drawString(trade.volume,window.width() - numbersWidth,(int)(window.fontHeight() * 0.5));
15: fontAttributes(Window.bold,true)
return 0;
16: width()
return 0;
17: height()
return 0;
18: fontHeight(trade.symbol)
return 0;
19: drawString(trade.symbol,window.width() - numbersWidth - 2 - symbolWidth,(int)(window.height()/2 - window.fontHeight(trade.symbol)/2));
20: fontAttributes(Window.bold,false)
return 0;

Task 2: Please answer all these questions.

Name: _____

Start Time: ____:____ **(hh : mm)** ** The time that you begin answer the question.

1. Provide all class names that access the TradeInfoSource class for getting the data.
   _____
2. Provide all class names that access the access the createFrame method in the Window class.
   _____
3. Provide the class names that are invoked by a constructor of the TradeInfoSource class.
   _____

End Time: ____:____ **(hh : mm)** ** The time that you completely answer all questions.

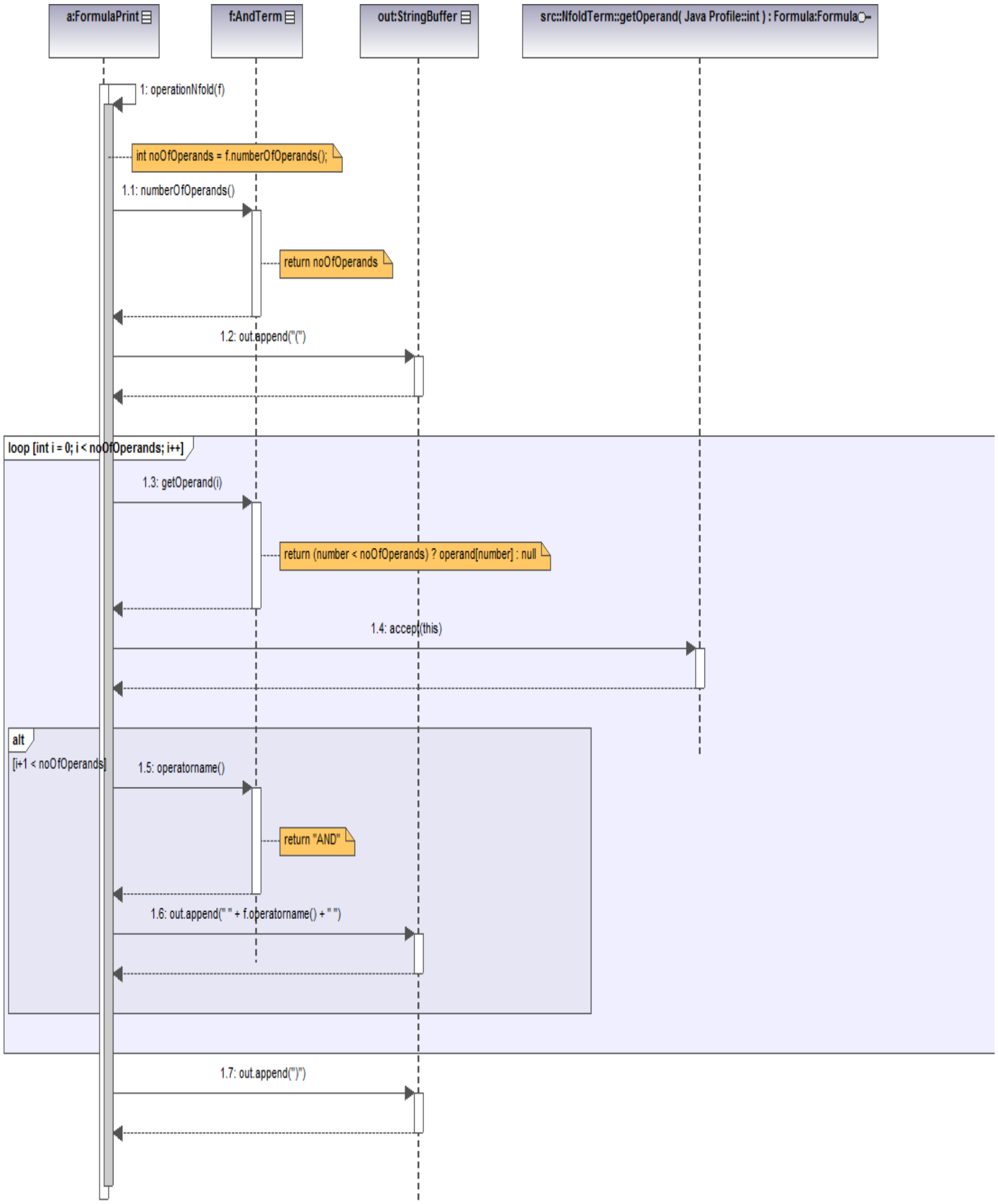
Program: Boolean Formulas (Design Pattern)

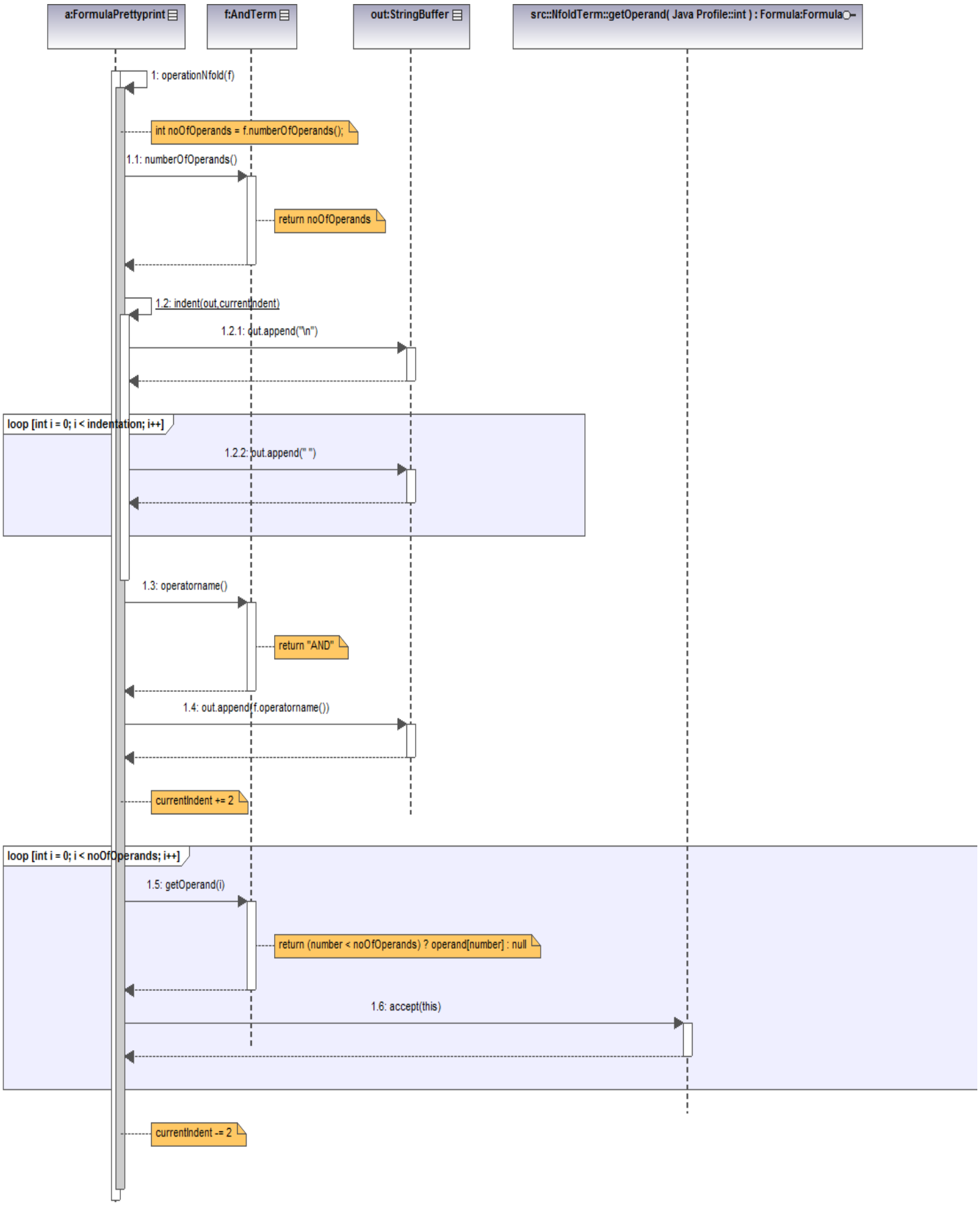
Description:

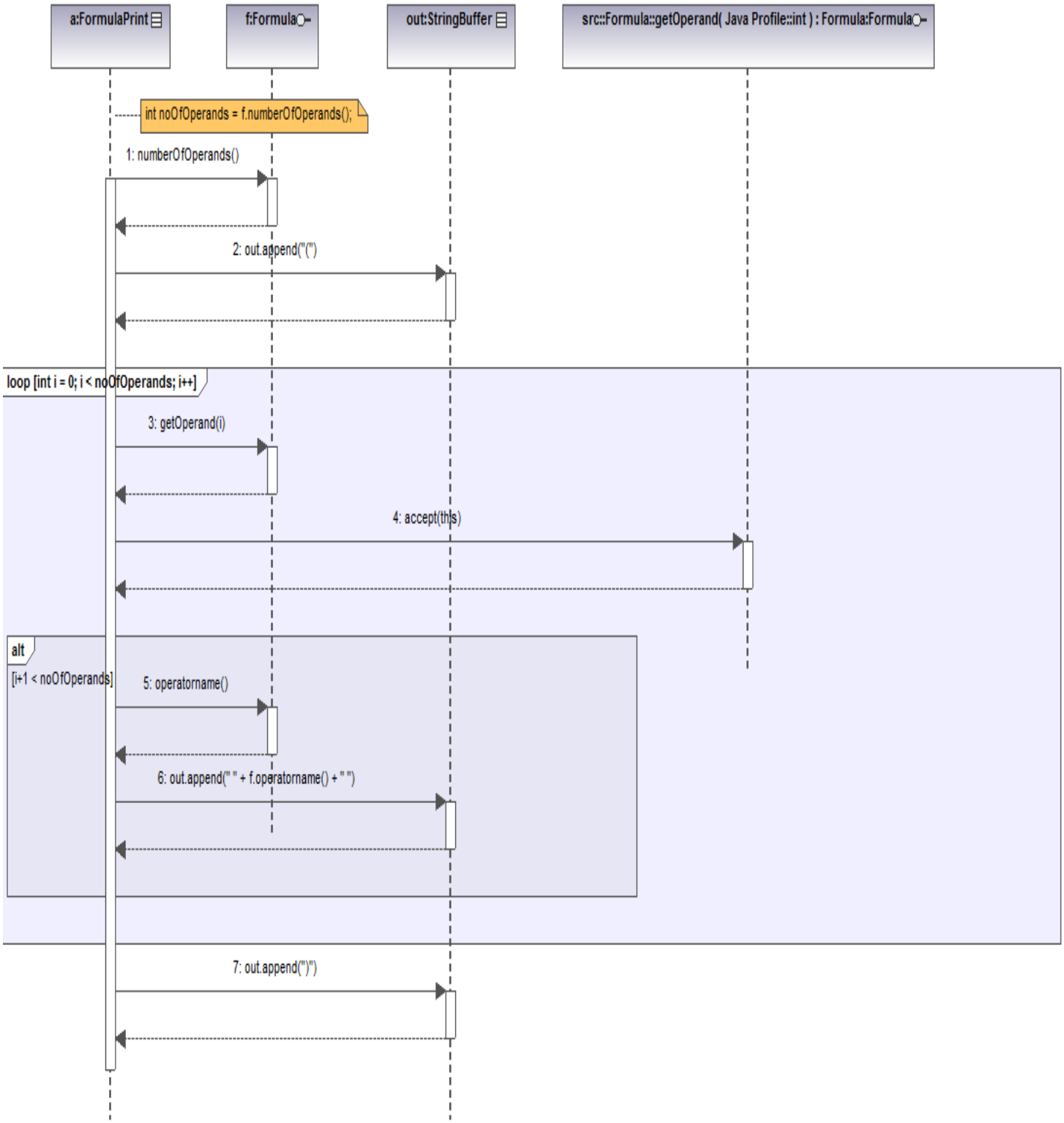
You will receive a "boolean formulas" design. It contains a library for representing boolean formulas (using AND, OR, XOR, NOT, and variables) and for printing the formulas in two different styles: in infix notation on a single line or prefix notation on multiple lines with indentations. Furthermore, it contains a small main program, which generates a formula and invokes both printing routines this leads to the following

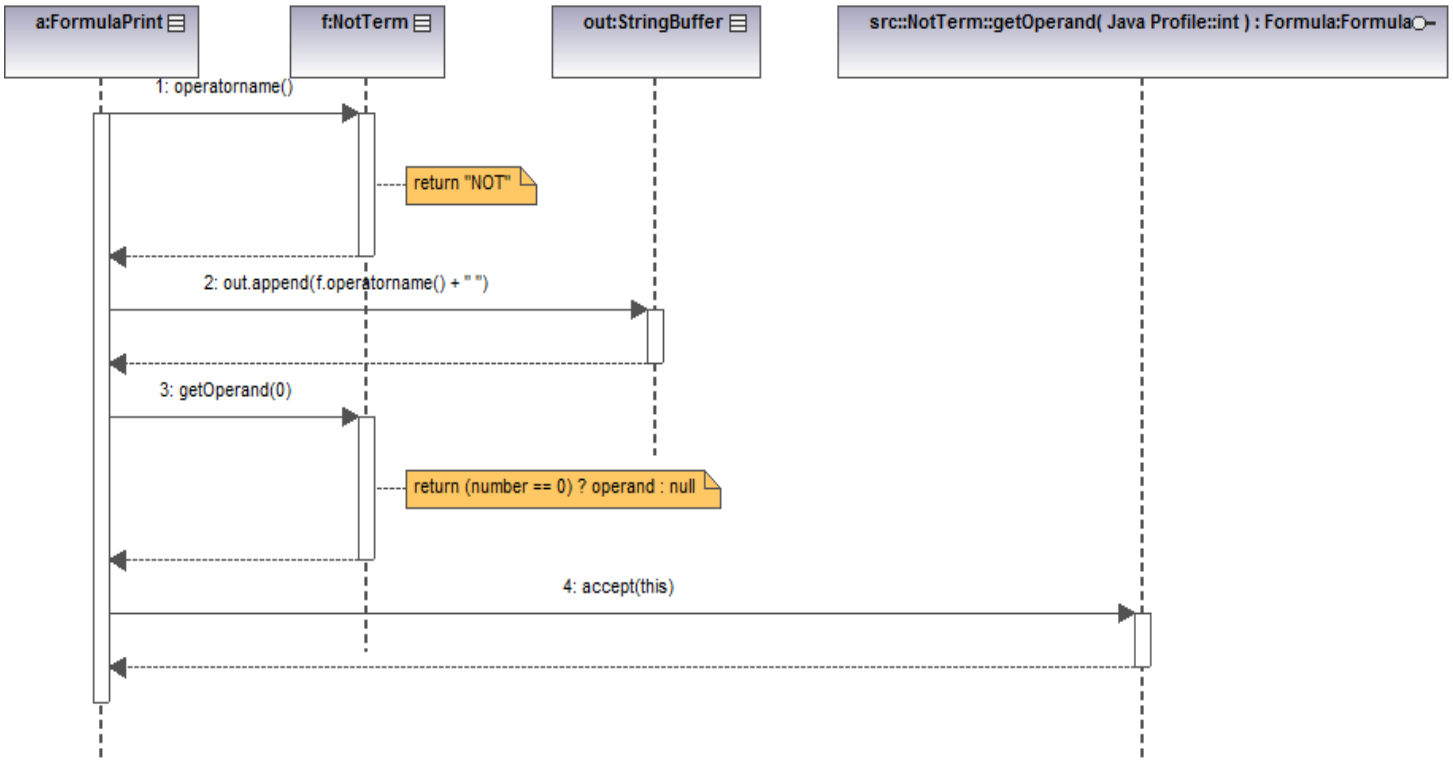
```
((a XOR NOT b) AND (NOT x1 OR NOT x2))

AND
  XOR
    a
    NOT
      b
  OR
    NOT
      x1
    NOT
      x2
```

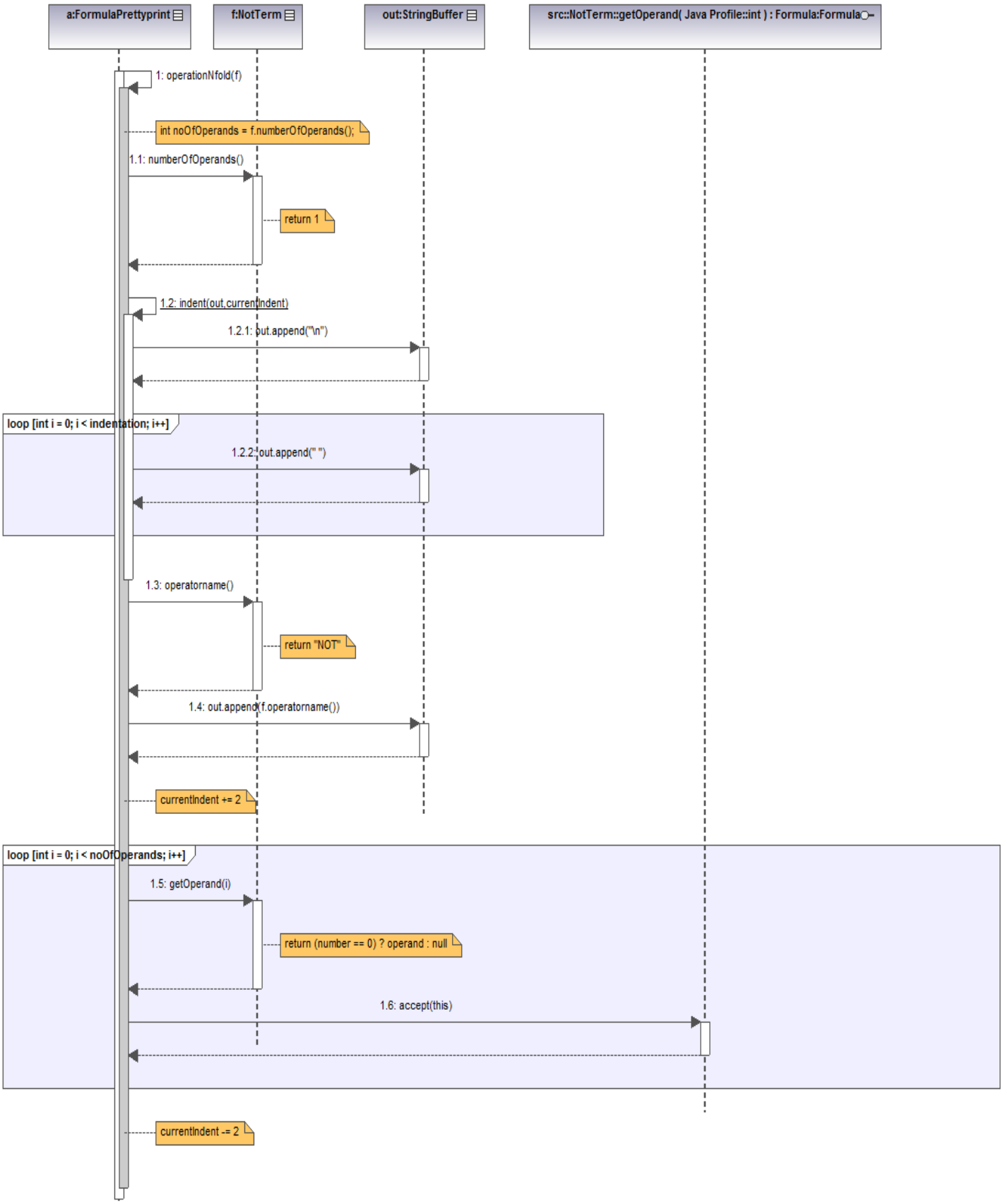

«interface»
Formula
accept(in op:FormulaOperation):void
getOperand(in number:int):Formula
numberOfOperands():int
operatorname():String
XorTerm
left:Formula
right:Formula
«constructor» XorTerm(in a:Formula, in b:Formula)
accept(in op:FormulaOperation):void
getOperand(in number:int):Formula
numberOfOperands():int
operatorname():String
-left
-right
-operand
-operand
*
NotTerm
operand:Formula
«constructor» NotTerm(in a:Formula)
accept(in op:FormulaOperation):void
getOperand(in number:int):Formula
numberOfOperands():int
operatorname():String
VarTerm
name:String
«constructor» VarTerm(in varname:String)
accept(in op:FormulaOperation):void
getOperand(in number:int):Formula
numberOfOperands():int
operatorname():String
NfoldTerm
maxNoOfOperands:int=50
operand:Formula[*]=new Formula[maxNoOfOperands]
noOfOperands:int
«constructor» NfoldTerm()
addOperand(in newoperand:Formula):int
getOperand(in number:int):Formula
numberOfOperands():int
«annotations» accept(in op:FormulaOperation):void
«annotations» operatorname():String
ValuesOfVariables
s:LinkedList
«constructor» ValuesOfVariables()
setTrue(in variable:String):void
setFalse(in variable:String):void
isTrue(in variable:String):boolean
AndTerm
«constructor» AndTerm()
«constructor» AndTerm(in left:Formula, in right:Formula)
accept(in op:FormulaOperation):void
operatorname():String
OrTerm
«constructor» OrTerm()
«constructor» OrTerm(in left:Formula, in right:Formula)
accept(in op:FormulaOperation):void
operatorname():String
exec
main(in args:String[*]):void
«interface»
FormulaOperation
operationAnd(in f:AndTerm):void
operationOr(in f:OrTerm):void
operationXor(in f:XorTerm):void
operationNot(in f:NotTerm):void
operationVar(in v:VarTerm):void
FormulaPrint
out:StringBuffer
«constructor» FormulaPrint(in outstream:StringBuffer, in f:Formula)
operationNfold(in f:Formula):void
operationAnd(in f:AndTerm):void
operationOr(in f:OrTerm):void
operationXor(in f:XorTerm):void
operationNot(in f:NotTerm):void
operationVar(in f:VarTerm):void
FormulaPrettyprint
out:StringBuffer
currentIndent:int
«constructor» FormulaPrettyprint(in outstream:StringBuffer, in f:Formula)
operationNfold(in f:Formula):void
indent(in out:StringBuffer, in indentation:int):void
operationAnd(in f:AndTerm):void
operationOr(in f:OrTerm):void
operationXor(in f:XorTerm):void
operationNot(in f:NotTerm):void
operationVar(in f:VarTerm):void

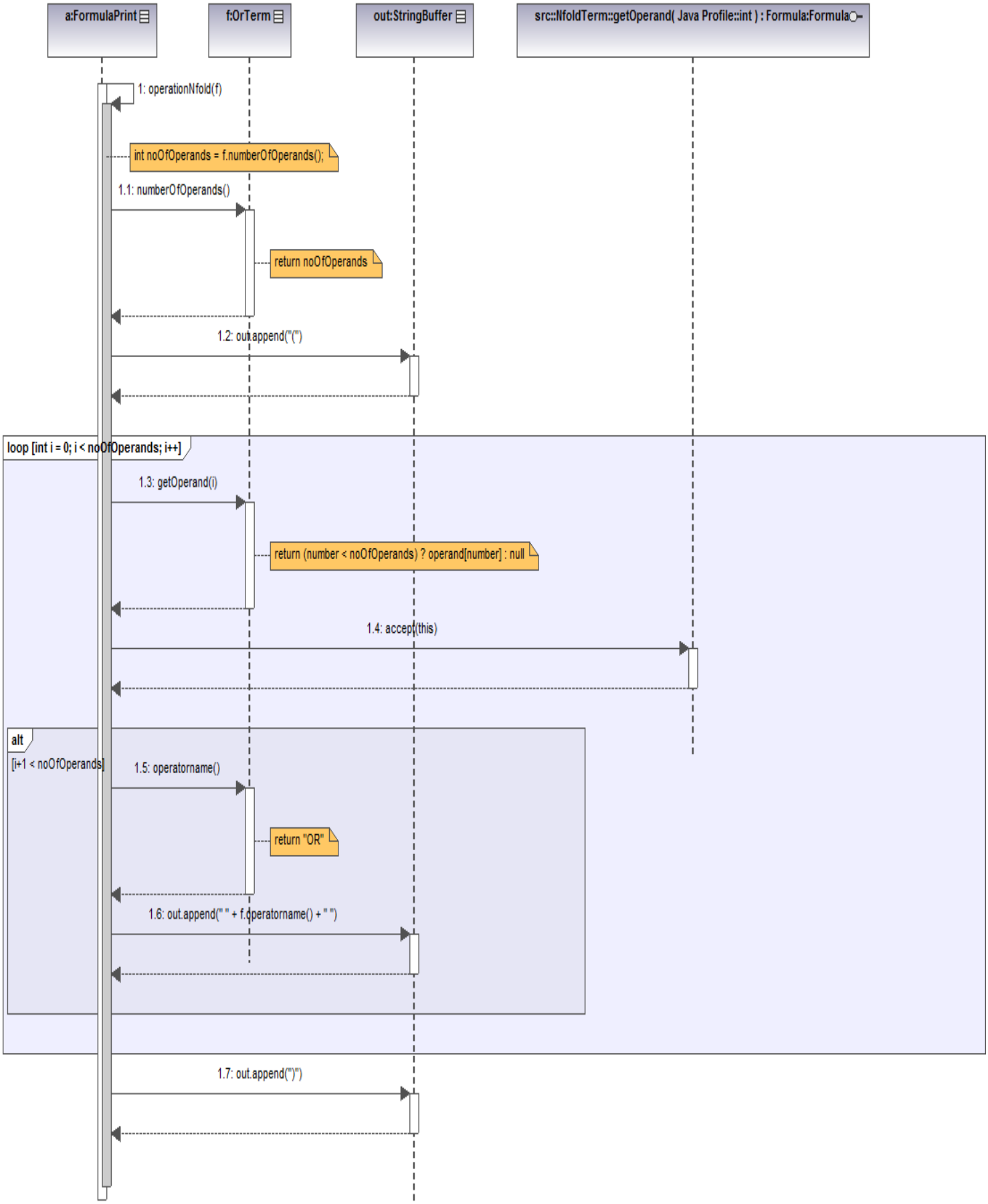

a:FormulaPrint
f:AndTerm
out:StringBuffer
src::NfoldTerm::getOperand(Java Profile::int) : Formula:Formula
1: operationNfold(f)
int noOfOperands = f.numberOfOperands();
1.1: numberOfOperands()
return noOfOperands
1.2: out.append("(")
loop [int i = 0; i < noOfOperands; i++]
1.3: getOperand(i)
return (number < noOfOperands) ? operand[number] : null
1.4: accept(this)
alt
[i+1 < noOfOperands]
1.5: operatorname()
return "AND"
1.6: out.append(" " + f.operatorname() + " ")
1.7: out.append(")")

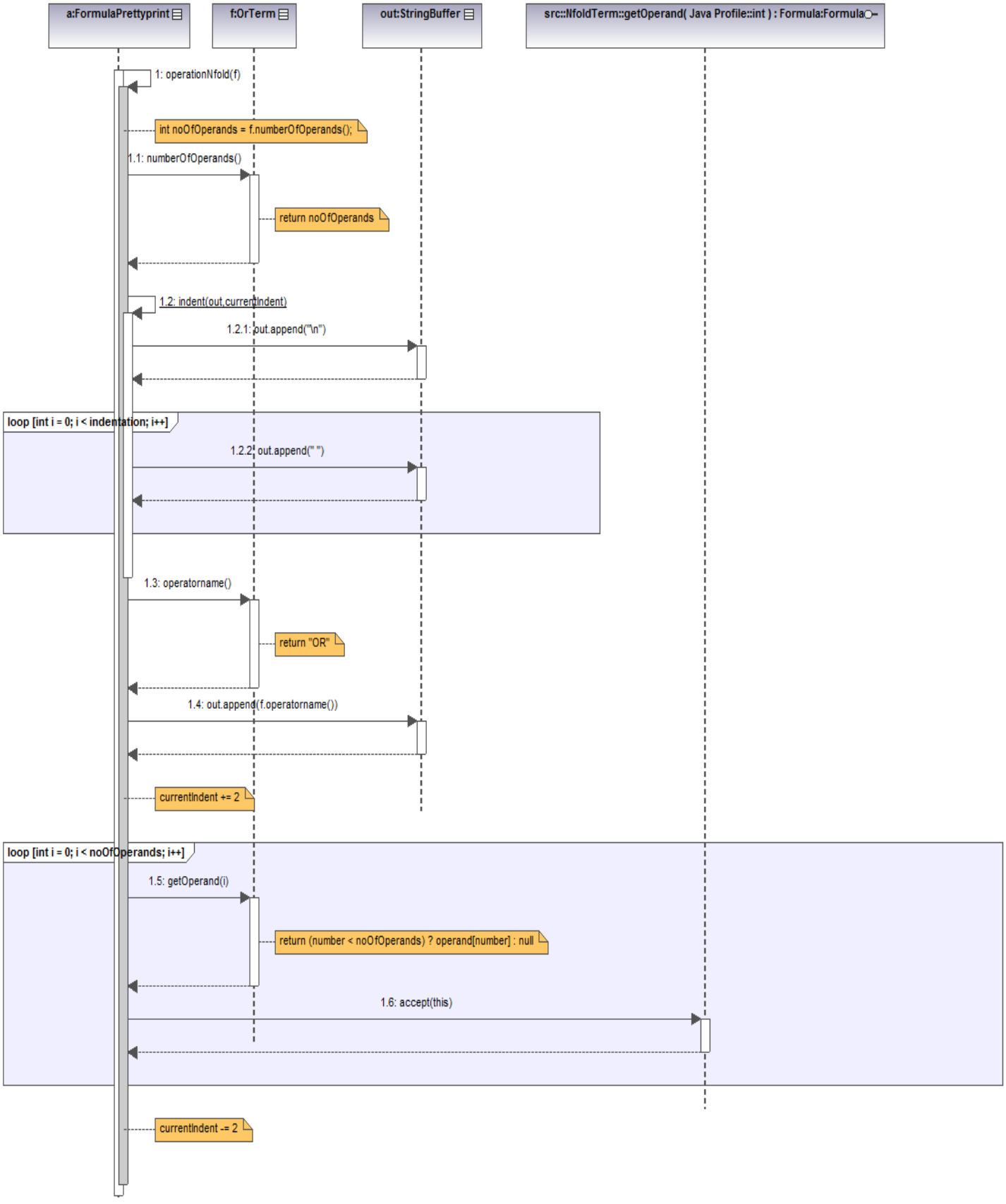

a:FormulaPrettyprint
f:AndTerm
out:StringBuffer
src::NfoldTerm::getOperand(Java Profile::int) : Formula:Formula
1: operationNfold(f)
int noOfOperands = f.numberOfOperands();
1.1: numberOfOperands()
return noOfOperands
1.2: indent(out,currentIndent)
1.2.1: out.append("\n")
loop [int i = 0; i < indentation; i++]
1.2.2: out.append(" ")
1.3: operatorname()
return "AND"
1.4: out.append(f.operatorname())
currentIndent += 2
loop [int i = 0; i < noOfOperands; i++]
1.5: getOperand(i)
return (number < noOfOperands) ? operand[number] : null
1.6: accept(this)
currentIndent -= 2

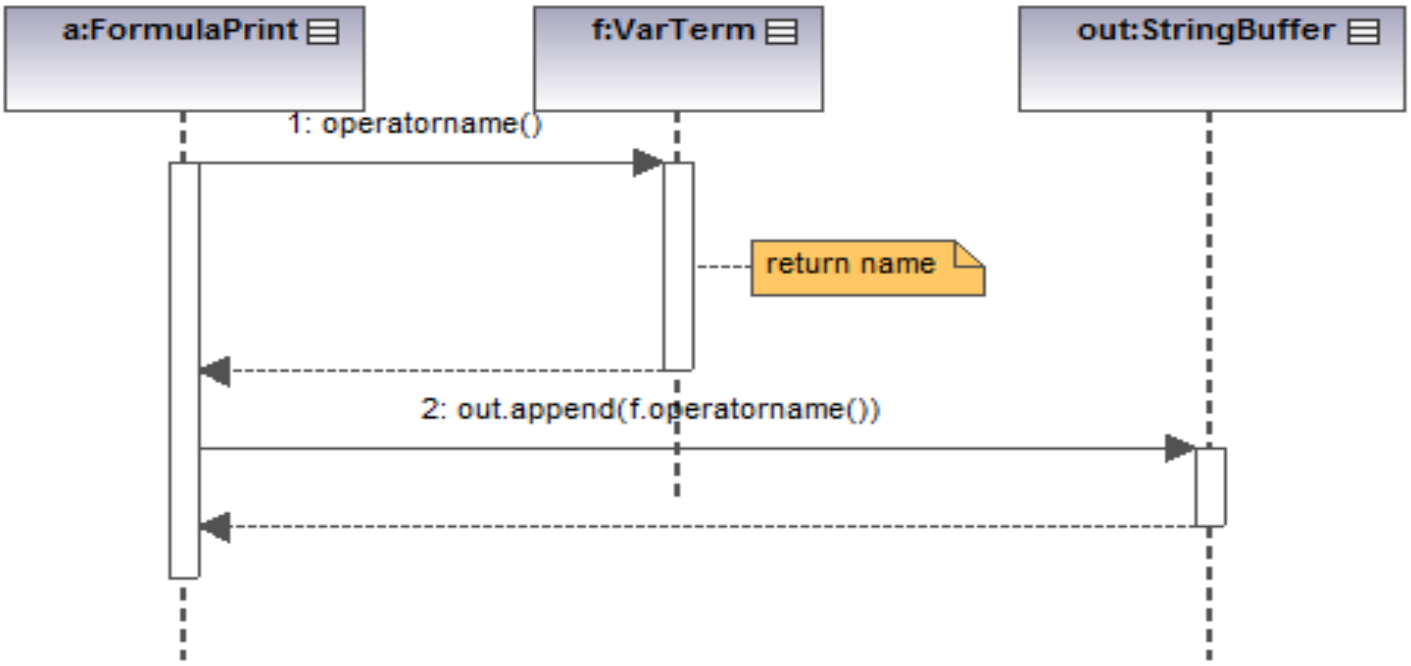

a:FormulaPrint
f:Formula
out:StringBuffer
src::Formula::getOperand(Java Profile::int) : Formula:Formula
int noOfOperands = f.numberOfOperands();
1: numberOfOperands()
2: out.append("(")
loop [int i = 0; i < noOfOperands; i++]
3: getOperand(i)
4: accept(this)
alt
[i+1 < noOfOperands]
5: operatorname()
6: out.append(" " + f.operatorname() + " ")
7: out.append(")")

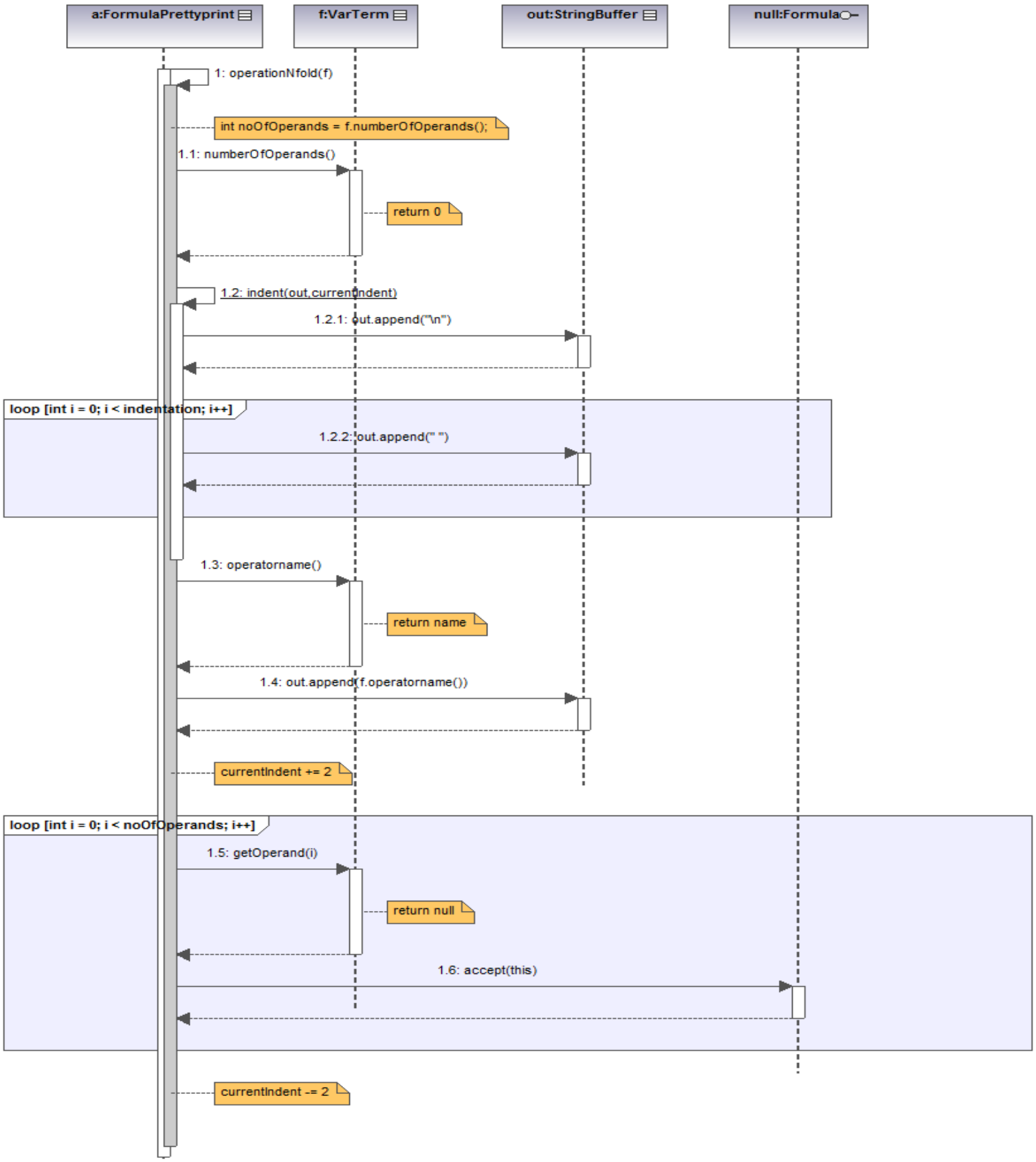

a:FormulaPrint
f:NotTerm
out:StringBuffer
src::NotTerm::getOperand(Java Profile::int) : Formula:Formula
1: operatorname()
return "NOT"
2: out.append(f.operatorname() + " ")
3: getOperand(0)
return (number == 0) ? operand : null
4: accept(this)

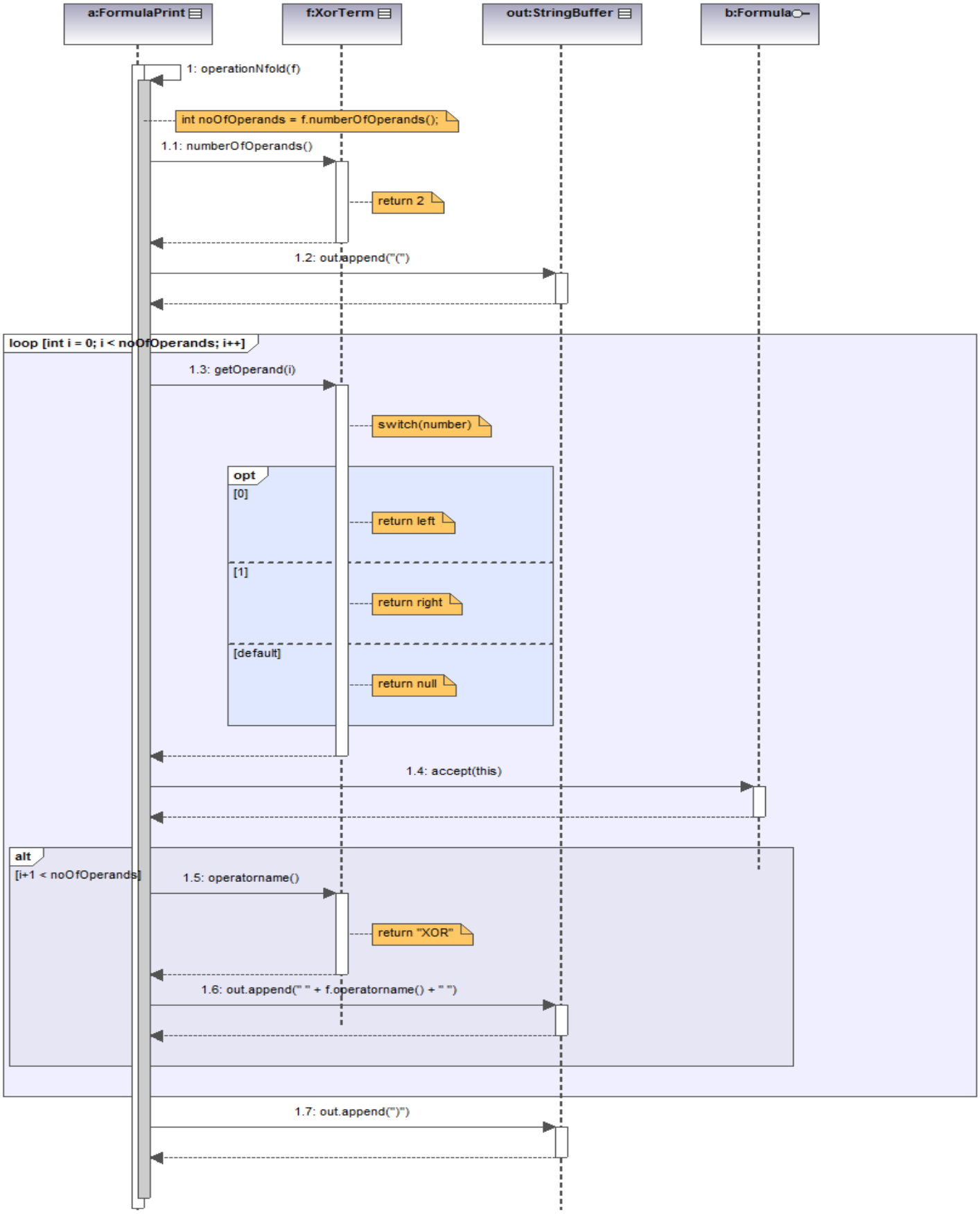

a:FormulaPrettyprint
f:NotTerm
out:StringBuffer
src::NotTerm::getOperand(Java Profile::int) : Formula:Formula
1: operationNfold(f)
int noOfOperands = f.numberOfOperands();
1.1: numberOfOperands()
return 1
1.2: indent(out,currentIndent)
1.2.1: out.append("\n")
loop [int i = 0; i < indentation; i++]
1.2.2: out.append(" ")
1.3: operatorname()
return "NOT"
1.4: out.append(f.operatorname())
currentIndent += 2
loop [int i = 0; i < noOfOperands; i++]
1.5: getOperand(i)
return (number == 0) ? operand : null
1.6: accept(this)
currentIndent -= 2

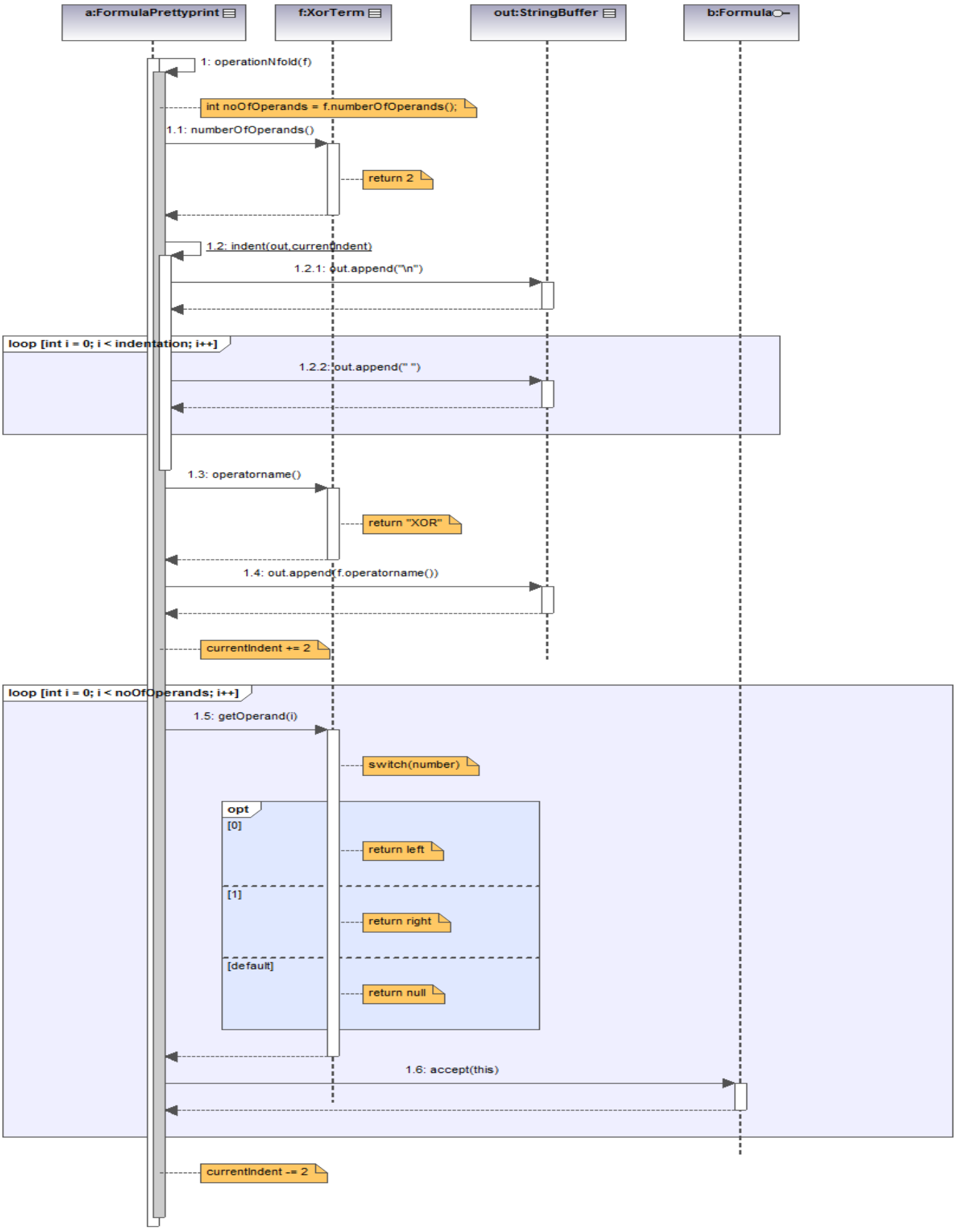

a:FormulaPrint
f:OrTerm
out:StringBuffer
src::NfoldTerm::getOperand(Java Profile::int) : Formula:Formula
1: operationNfold(f)
int noOfOperands = f.numberOfOperands();
1.1: numberOfOperands()
return noOfOperands
1.2: out.append("(")
loop [int i = 0; i < noOfOperands; i++]
1.3: getOperand(i)
return (number < noOfOperands) ? operand[number] : null
1.4: accept(this)
alt
[i+1 < noOfOperands]
1.5: operatorname()
return "OR"
1.6: out.append(" " + f.operatorname() + " ")
1.7: out.append(")")


a:FormulaPrettyprint
f:OrTerm
out:StringBuffer
src::NfoldTerm::getOperand(Java Profile::int) : Formula:Formula
1: operationNfold(f)
int noOfOperands = f.numberOfOperands();
1.1: numberOfOperands()
return noOfOperands
1.2: indent(out.currentIndent)
1.2.1: out.append("\n")
loop [int i = 0; i < indentation; i++]
1.2.2: out.append(" ")
1.3: operatorname()
return "OR"
1.4: out.append(f.operatorname())
currentIndent += 2
loop [int i = 0; i < noOfOperands; i++]
1.5: getOperand(i)
return (number < noOfOperands) ? operand[number] : null
1.6: accept(this)
currentIndent -= 2


a:FormulaPrint
f:VarTerm
out:StringBuffer
1: operatorname()
return name
2: out.append(f.operatorname())


a:FormulaPrettyprint
f:VarTerm
out:StringBuffer
null:Formula
1: operationNfold(f)
int noOfOperands = f.numberOfOperands();
1.1: numberOfOperands()
return 0
1.2: indent(out,currentIndent)
1.2.1: out.append("\n")
loop [int i = 0; i < indentation; i++]
1.2.2: out.append(" ")
1.3: operatorname()
return name
1.4: out.append(f.operatorname())
currentIndent += 2
loop [int i = 0; i < noOfOperands; i++]
1.5: getOperand(i)
return null
1.6: accept(this)
currentIndent -= 2


a:FormulaPrint
f:XorTerm
out:StringBuffer
b:Formula
1: operationNfold(f)
int noOfOperands = f.numberOfOperands();
1.1: numberOfOperands()
return 2
1.2: out.append("(")
loop [int i = 0; i < noOfOperands; i++]
1.3: getOperand(i)
switch(number)
opt
[0]
return left
[1]
return right
[default]
return null
1.4: accept(this)
alt
[i+1 < noOfOperands]
1.5: operatorname()
return "XOR"
1.6: out.append(" " + f.operatorname() + " ")
1.7: out.append(")")


a:FormulaPrettyprint
f:XorTerm
out:StringBuffer
b:Formula
1: operationNfold(f)
int noOfOperands = f.numberOfOperands();
1.1: numberOfOperands()
return 2
1.2: indent(out,currentIndent)
1.2.1: out.append("\n")
loop [int i = 0; i < indentation; i++]
1.2.2: out.append(" ")
1.3: operatorname()
return "XOR"
1.4: out.append(f.operatorname())
currentIndent += 2
loop [int i = 0; i < noOfOperands; i++]
1.5: getOperand(i)
switch(number)
opt
[0]
return left
[1]
return right
[default]
return null
1.6: accept(this)
currentIndent -= 2

Task 3: Please answer all these questions.

Name: _____

Start Time: ____:____ **(hh : mm)** ** The time that you begin answer the question.

1. Provide all class names that invoke the numberofOperands method in the Andterm class.
   _____

2. Provide all methods name that are invoked by a constructor of the Orterm class.
   _____

3. Provide the returned value of the getOperandmehod in the VerTerm class.
   _____

4. Provide the returned value of the numberOfOperands method in the XorTerm class.
   _____

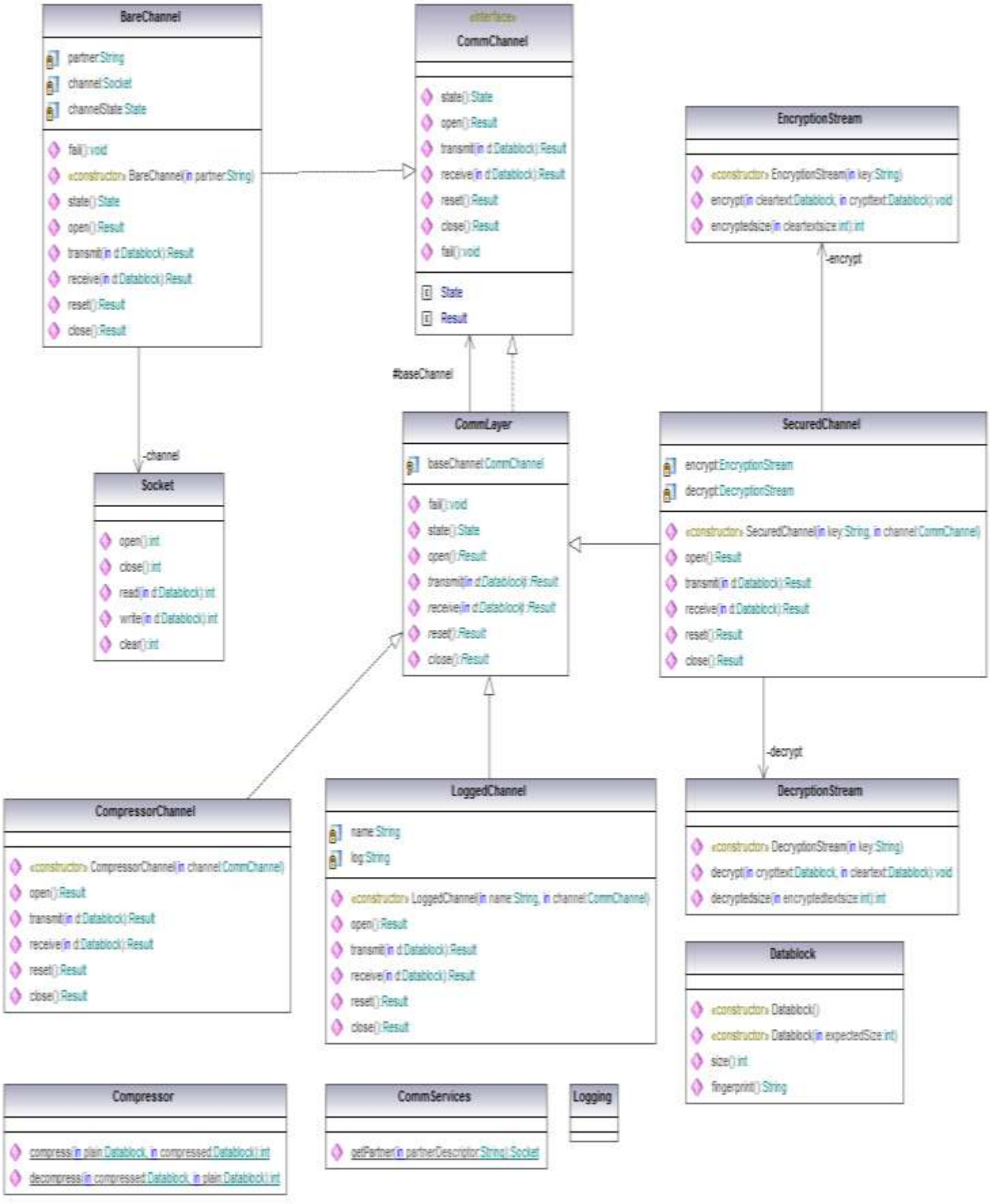
5. Provide the returned value of the numberOfOperands method in the VarTerm class.
   _____

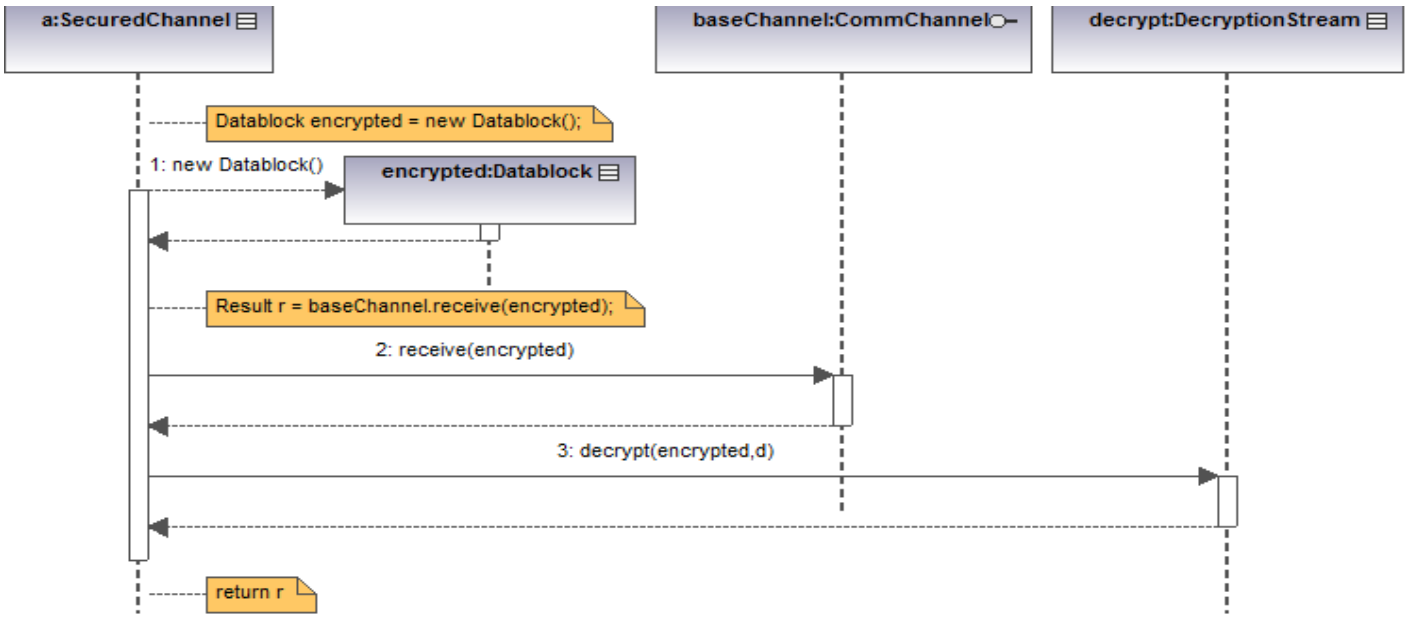
6. Provide the returned value of the numberOfOperands method in the Notterm class.
   _____

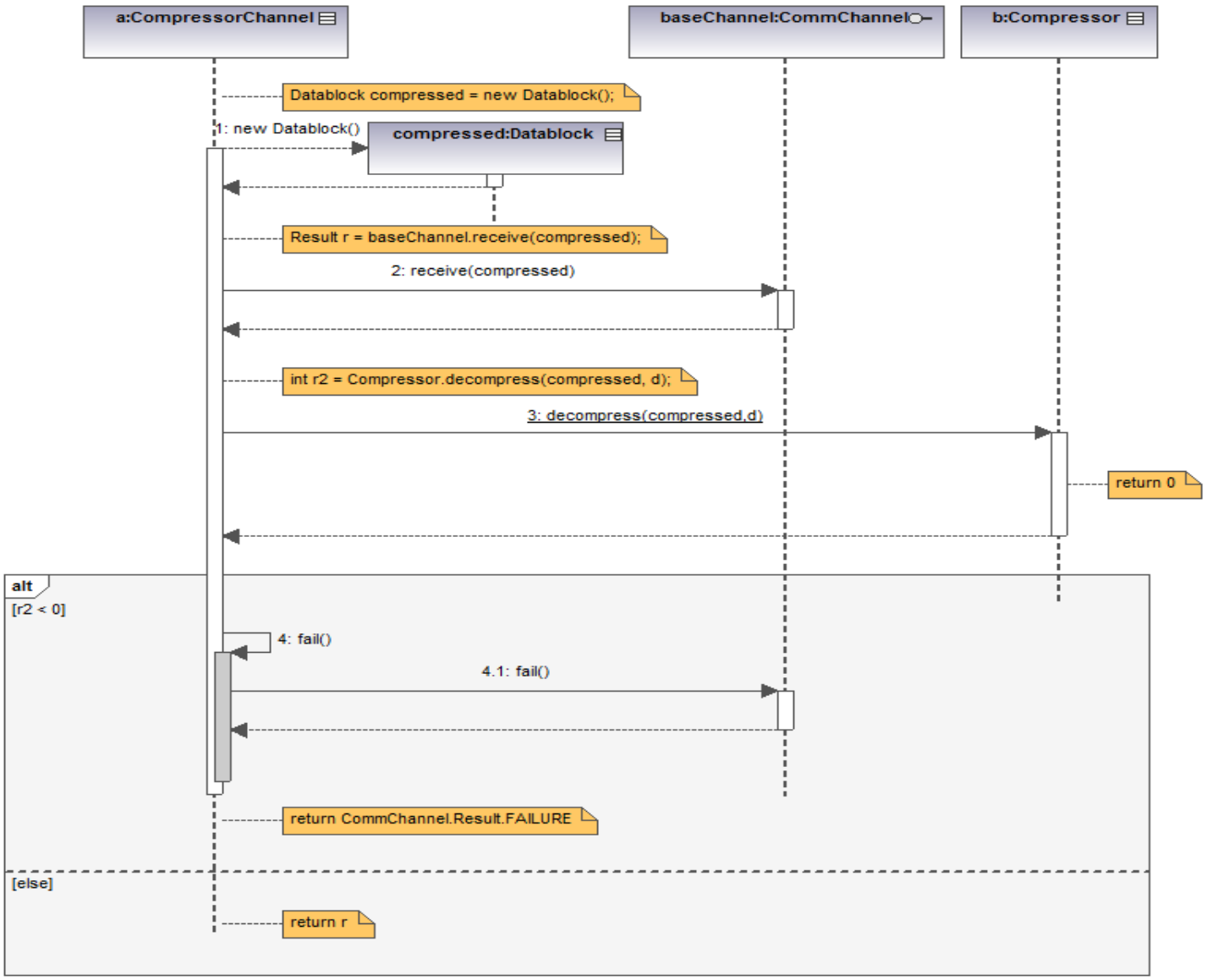
End Time: ____:____ **(hh : mm)** ** The time that you completely answer all questions.

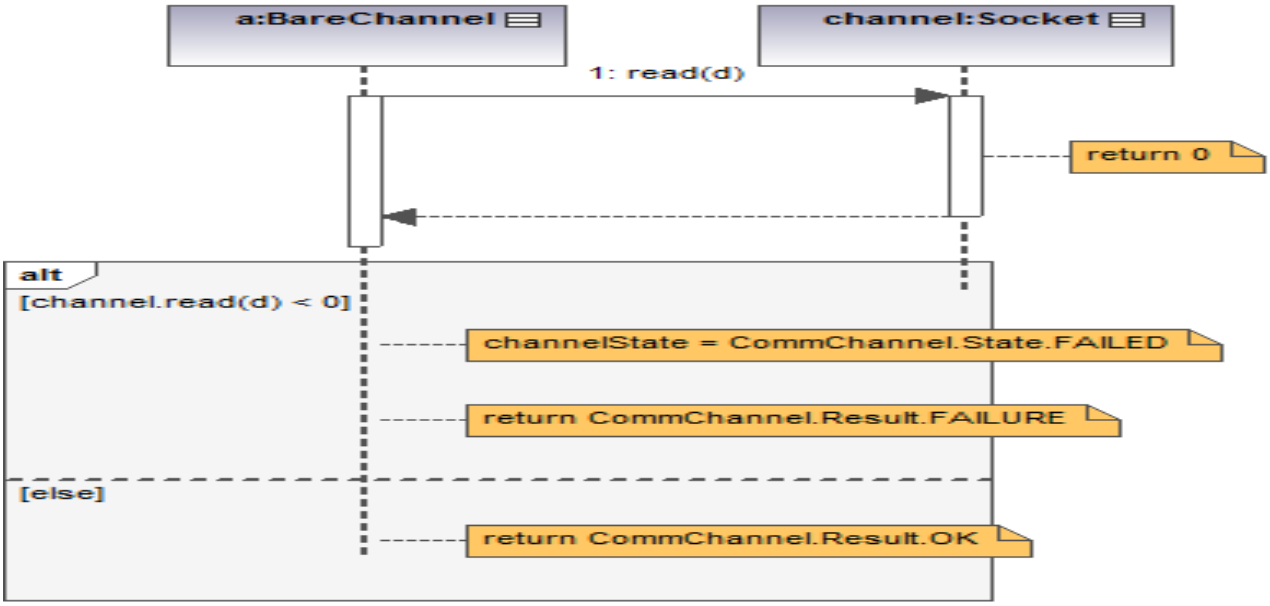
Program: Communication Channel Library (Design Patterns)

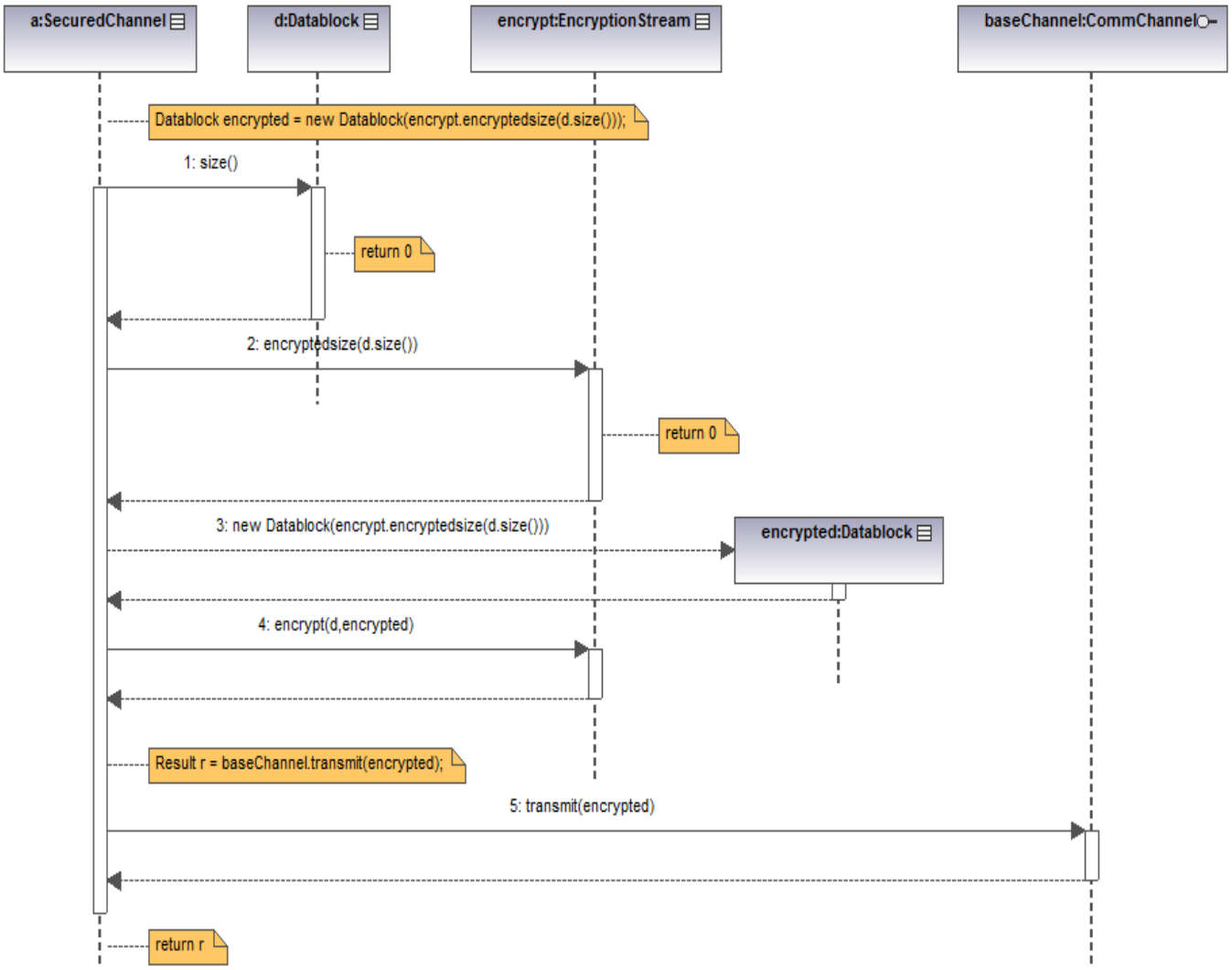
Description:
Shortly, you will receive a "communication channels" design.
It consists of a library for configuring and running a communication channel. Such a channel CommChannel establishes a communication connection and one can add logging, data compression, and encryption.
The library does not implement the functionality itself, instead it implements a simplified interface for the *combination* of the parts. The names of the operations indicate approximately the functionality; an exact definition is not necessary for the tasks.

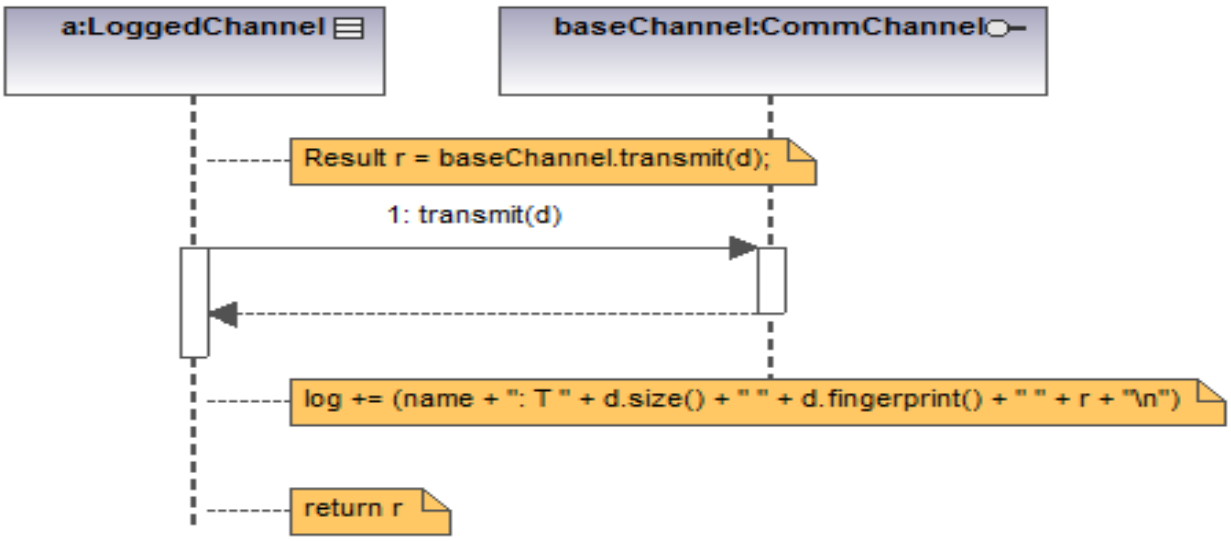

BareChannel
partner:String
channel:Socket
channelState:State
fail():void
«constructor» BareChannel(in partner:String)
state():State
open():Result
transmit(in d:Datablock):Result
receive(in d:Datablock):Result
reset():Result
close():Result
«interface»
CommChannel
state():State
open():Result
transmit(in d:Datablock):Result
receive(in d:Datablock):Result
reset():Result
close():Result
fail():void
State
Result
EncryptionStream
«constructor» EncryptionStream(in key:String)
encrypt(in cleartext:Datablock, in crypttext:Datablock):void
encryptedsize(in cleartextsize:int):int
-encrypt
#baseChannel
-channel
Socket
open():int
close():int
read(in d:Datablock):int
write(in d:Datablock):int
clear():int
CommLayer
baseChannel:CommChannel
fail():void
state():State
open():Result
transmit(in d:Datablock):Result
receive(in d:Datablock):Result
reset():Result
close():Result
SecuredChannel
encrypt:EncryptionStream
decrypt:DecryptionStream
«constructor» SecuredChannel(in key:String, in channel:CommChannel)
open():Result
transmit(in d:Datablock):Result
receive(in d:Datablock):Result
reset():Result
close():Result
-decrypt
CompressorChannel
«constructor» CompressorChannel(in channel:CommChannel)
open():Result
transmit(in d:Datablock):Result
receive(in d:Datablock):Result
reset():Result
close():Result
LoggedChannel
name:String
log:String
«constructor» LoggedChannel(in name:String, in channel:CommChannel)
open():Result
transmit(in d:Datablock):Result
receive(in d:Datablock):Result
reset():Result
close():Result
DecryptionStream
«constructor» DecryptionStream(in key:String)
decrypt(in crypttext:Datablock, in cleartext:Datablock):void
decryptedsize(in encrypteditextsize:int):int
Datablock
«constructor» Datablock()
«constructor» Datablock(in expectedSize:int)
size():int
fingerprint():String
Compressor
compress(in plain:Datablock, in compressed:Datablock):int
decompress(in compressed:Datablock, in plain:Datablock):int
CommServices
getPartner(in partnerDescriptor:String):Socket
Logging

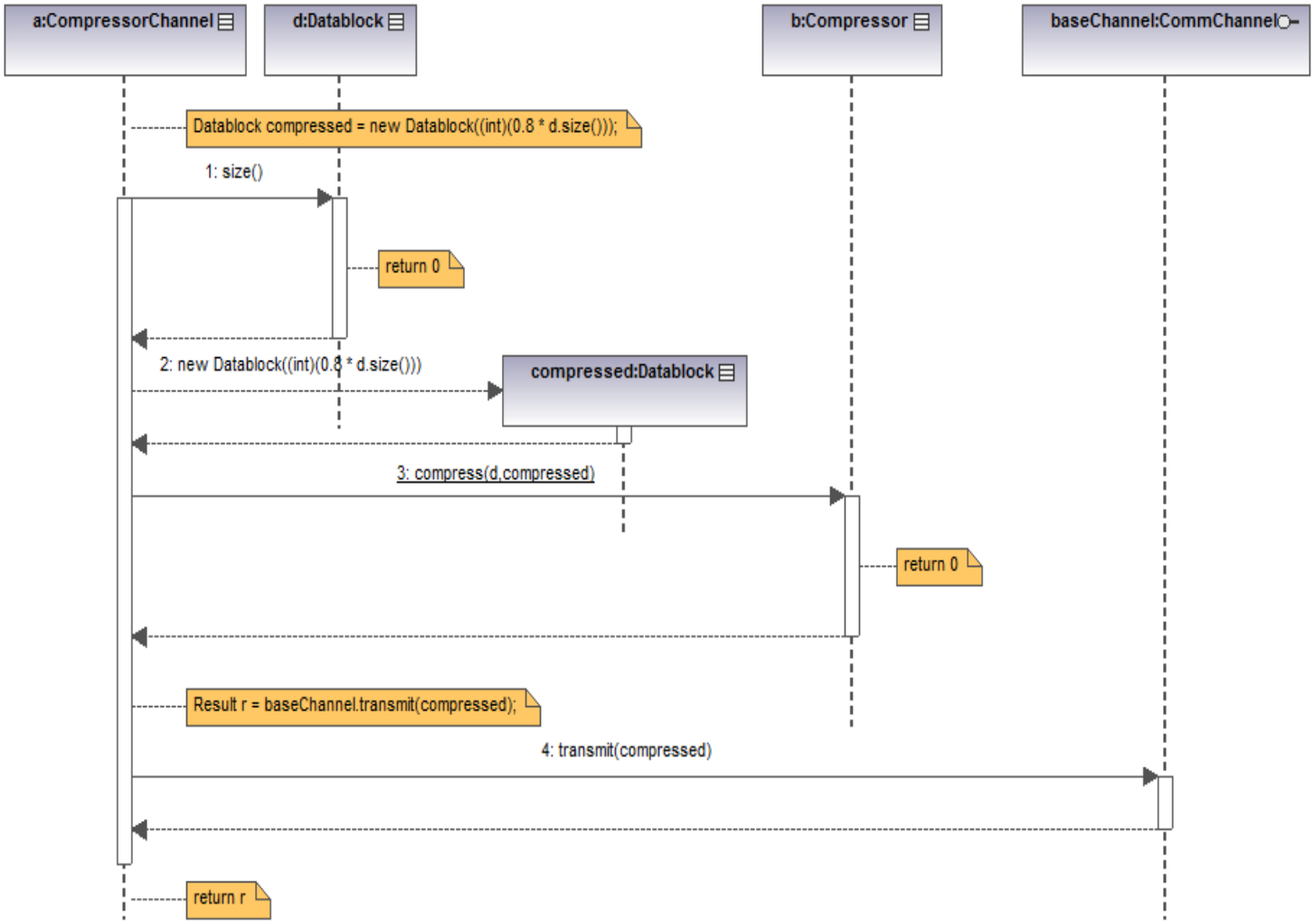

a:SecuredChannel
baseChannel:CommChannel
decrypt:DecryptionStream
Datablock encrypted = new Datablock();
1: new Datablock()
encrypted:Datablock
Result r = baseChannel.receive(encrypted);
2: receive(encrypted)
3: decrypt(encrypted,d)
return r

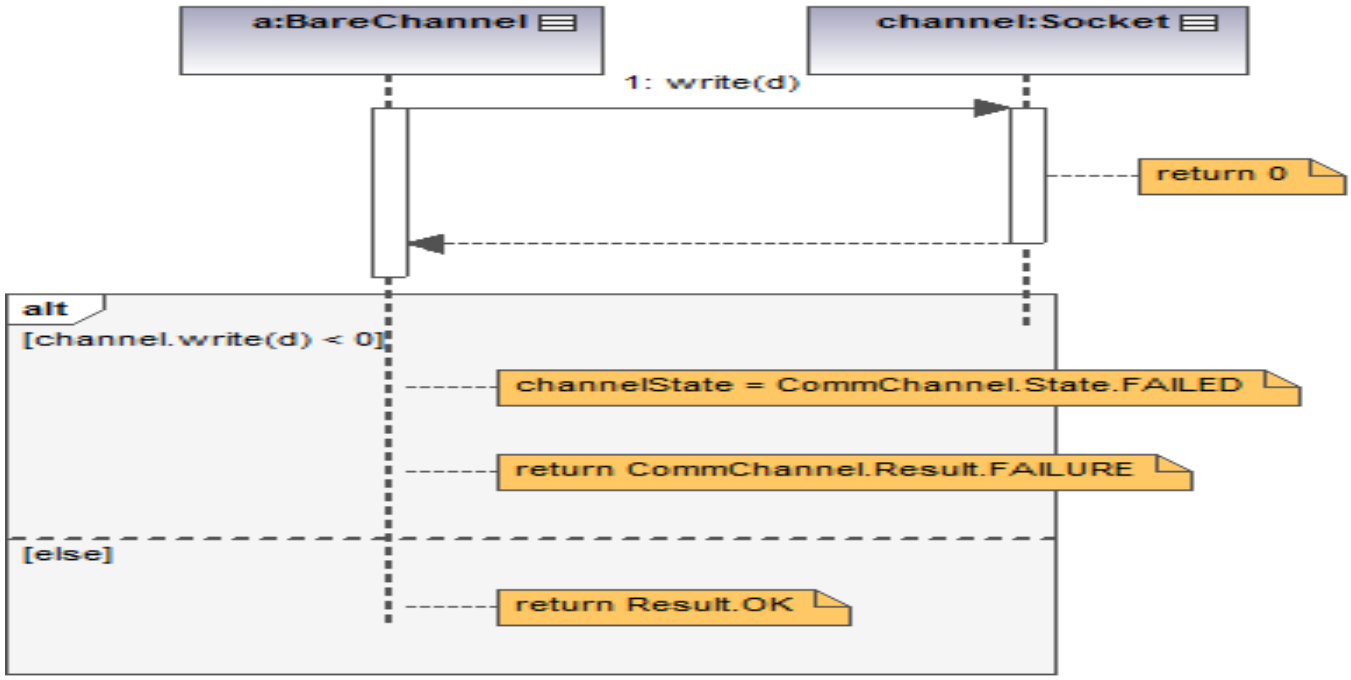

a:CompressorChannel
baseChannel:CommChannel
b:Compressor
Datablock compressed = new Datablock();
1: new Datablock()
compressed:Datablock
Result r = baseChannel.receive(compressed);
2: receive(compressed)
int r2 = Compressor.decompress(compressed, d);
3: decompress(compressed,d)
return 0
alt
[r2 < 0]
4: fail()
4.1: fail()
return CommChannel.Result.FAILURE
[else]
return r


a:BareChannel
channel:Socket
1: read(d)
return 0
alt
[channel.read(d) < 0]
channelState = CommChannel.State.FAILED
return CommChannel.Result.FAILURE
[else]
return CommChannel.Result.OK


a:SecuredChannel
d:Datablock
encrypt:EncryptionStream
baseChannel:CommChannel
Datablock encrypted = new Datablock(encrypt.encryptedsize(d.size()));
1: size()
return 0
2: encryptedsize(d.size())
return 0
3: new Datablock(encrypt.encryptedsize(d.size()))
encrypted:Datablock
4: encrypt(d,encrypted)
Result r = baseChannel.transmit(encrypted);
5: transmit(encrypted)
return r


a:LoggedChannel
baseChannel:CommChannel
Result r = baseChannel.transmit(d);
1: transmit(d)
log += (name + ": T " + d.size() + " " + d.fingerprint() + " " + r + "\n")
return r


a:CompressorChannel
d:Datablock
b:Compressor
baseChannel:CommChannel
Datablock compressed = new Datablock((int)(0.8 * d.size()));
1: size()
return 0
2: new Datablock((int)(0.8 * d.size()))
compressed:Datablock
3: compress(d,compressed)
return 0
Result r = baseChannel.transmit(compressed);
4: transmit(compressed)
return r


a:BareChannel
channel:Socket
1: write(d)
return 0
alt
[channel.write(d) < 0]
channelState = CommChannel.State.FAILED
return CommChannel.Result.FAILURE
[else]
return Result.OK

Task 4: Please answer all these questions.

Name: _____

Start Time: ____:____ **(hh : mm)** ** The time that you begin answer the question.

1. List the class name(s) that are specialize for data decryption.
   _____

2. List the class name(s) that are specialize for data encryption.
   _____

3. List the class name(s) that respond to compress the data.
   _____

4. List the class name(s) that represent the unit of data between computers.
   _____

5. List the class name(s) that control a communication channel in the computer.
   _____

6. List the class name(s) that use the function from the class Logging.
   _____

7. List the class name(s) that provide stream for writing into a log file.
   _____

8 List the class name(s) that establishes a connection between two Sockets on different computers.
   _____

9. List the method name(s) in the class CommChannel, which are accessed by the class Datablock.
   _____

10. List the class name(s) that do not access the class Datablock.
    _____

End Time: ____:____ **(hh : mm)** ** The time that you completely answer all questions.