Back to Makefiles for just a minute…

How about hello world?

1. Create a new rule. When, I run “make hello”, it should print:
   - “Hello Professor Keefe!”
   - use the “echo” command inside make to print something, also try “@echo”

2. Then, on the next line, make it print:
   - It is now: Tue Apr 19 07:10:52 CDT 2016
   - Note: you can get the current date with the unix “date” command.

3. Then, instead of “Professor Keefe”, change the first line to say Hello to whatever name is specified in the file name.txt.
   - Use the unix command: cat name.txt

Typically, we use make to "make" new files.

- This is how we use it when we compile programs, we follow the rules specified in the Makefile in order to "make" a new file, our program.

- Let's update our Hello World so that rather than printing to the screen it prints the same information to a file named hello.txt.

    1. Update the name of the rule to be hello.txt, so now it should run when you type "make hello.txt".

    2. Use the unix shell redirect command ">" to redirect the echo output to hello.txt.

Let's talk about dependencies.

- Does "hello.txt" depend upon anything?

- Add "name.txt" as a dependency for the "hello.txt" rule in your Makefile.
- Now, try running "make hello.txt" a few times. What happens? What happens if you edit name.txt then re-run make?

Writing more complex rules: tips to keep in mind.

- For each line of a rule, make will start a **separate** shell using the unix sh command.
- This means if you want to run two commands in the same shell you need to put both commands on the same line separated by semicolons.
- After each line is executed, make checks to see if there is a non-zero exit code (i.e., an error). If so, it will print out “make: *** [rulename] Error 1” and then stop.

Here's another challenge:

- Create a new rule called "testfiles" that:
  - Creates a new file that contains one line of text that reads: "File #1".
  - Creates a second new file that contains one line of text that reads: "File #2".
  - Uses `diff --brief file1.txt file2.txt` to test to see if the two files differ.
  - If they differ, print out "They differ!" and if they are the same print out "They are the same!"

Creative Uses of Makefiles: A Brainstorming Exercise

- Example 1: My moment of “Makefile enlightenment” during graduate school :)
- Your examples??

Writing as Part of Releasing a Software Project / UML Sequence Diagrams

UNIVERSITY OF MINNESOTA
PROFESSOR DANIEL F. KEEFE

Relevant forms of writing

1. Source code (i.e., self-documenting code)
2. Source code comments and doxygen-style documentation
3. A project website that provides a project summary
4. Tutorials for users
5. Tutorials for developers
6. Code organization / design guide for developers
7. Coding style guide for developers
8. Developers' blogs

Relevant forms of writing

1. Source code (i.e., self-documenting code)
2. Source code comments and doxygen-style documentation
3. A project website that provides a project summary
4. Tutorials for users
5. Tutorials for developers
6. Code organization / design guide for developers
7. Coding style guide for developers
8. Developers' blogs

Find a great example, and tell us why it is great:

3. A project website that provides a project summary
4. Tutorials for users
5. Tutorials for developers
6. Code organization / design documentation for developers
7. Coding style guide for developers
8. Developers' blogs

Some that I found:

3. A project website that provides a project summary
   - http://www.gwtproject.org/overview.html
4. Tutorials for users
   - http://www.photoshopessentials.com/photo-editing/how-to-auto-align-and-composite-images-in-photoshop/
5. Tutorials for developers
   - http://unity3d.com/learn/tutorials/modules/beginner/scripting/arrays
   - http://developer.gimp.org/writing-a-plug-in/1/index.html
6. Code organization / design documentation for developers
   - http://gnudict.sourceforge.net/
   - https://github.com/snowplow/snowplow/wiki/Snowplow-technical-documentation
7. Coding style guide for developers
   - http://wiki.blender.org/index.php/Dev:Doc/Code_Style
   - http://unity3d.com/learn/tutorials/modules/intermediate/scripting/coding-practices
8. Developers' blogs
   - http://blog.golang.org/ (look at discussion of "Good Package Names")

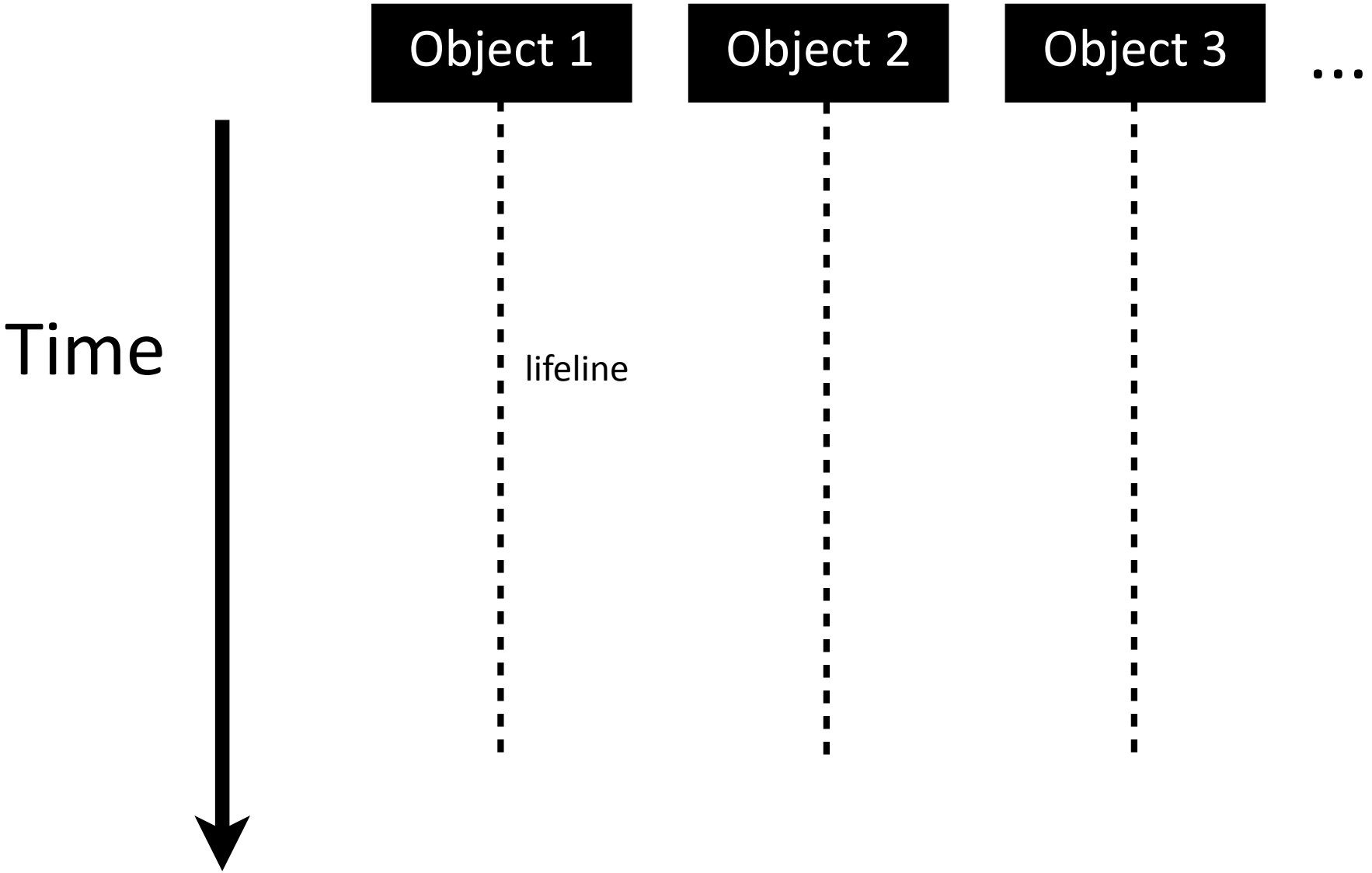
Part 2: Explaining and Understanding Code via UML Sequence Diagrams

UNIVERSITY OF MINNESOTA
PROFESSOR DANIEL F. KEEFE

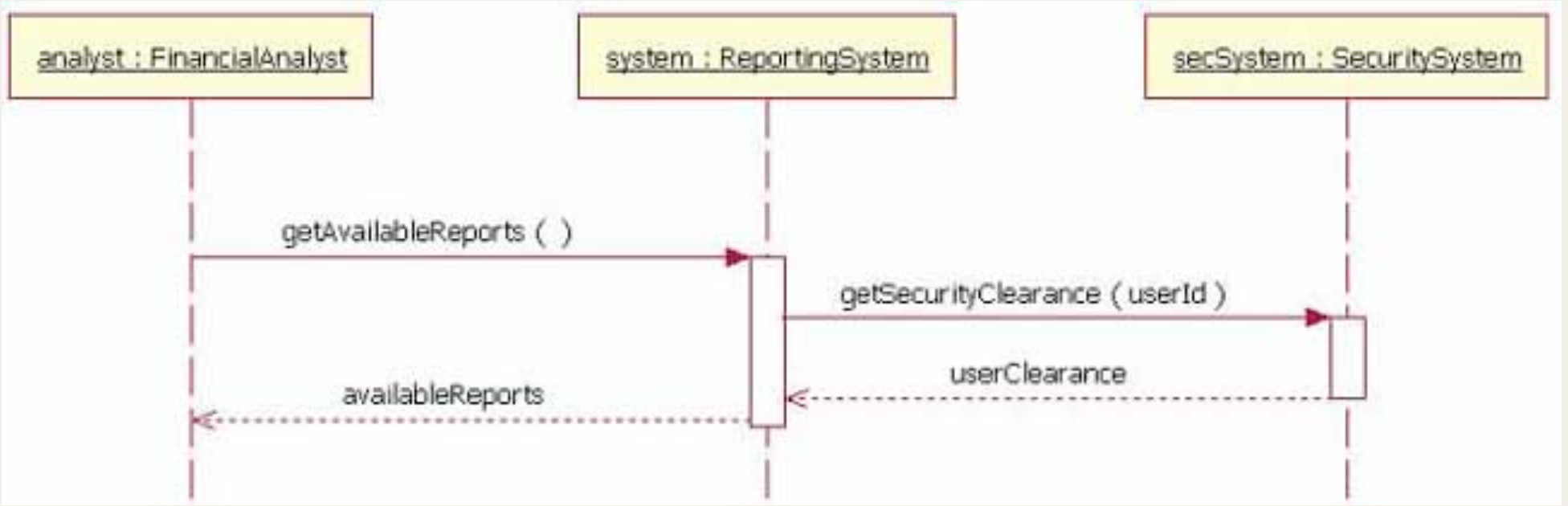
UML Sequence Diagrams

- Used primarily to show the interactions between objects and the sequential order in which those interactions occur.
- Define event sequences that result in some desired outcome.
- Focus is less on messages themselves and more on the order in which they occur, although most diagrams will do a pretty good job of identifying specific messages that are passed.

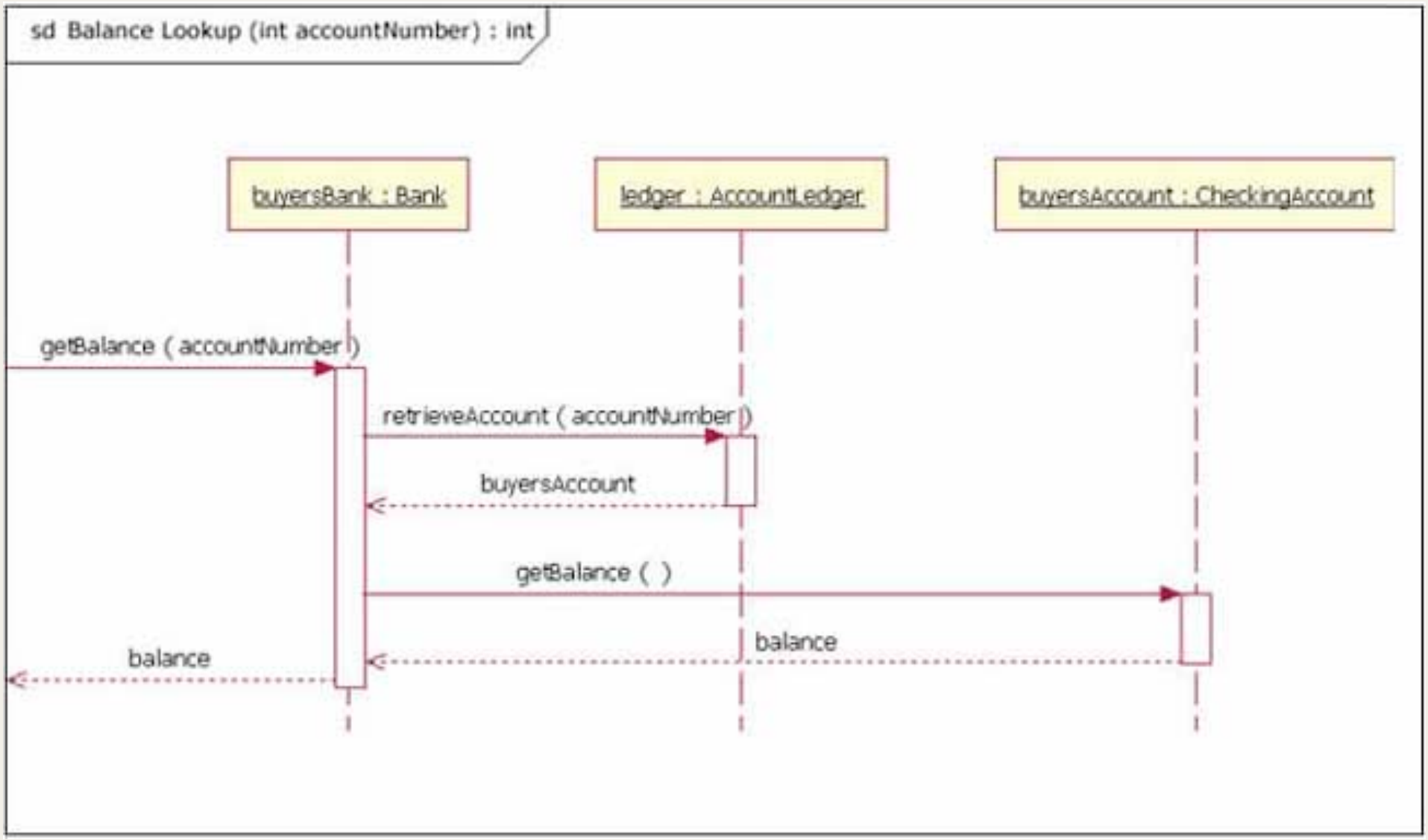
Diagram Format



Example


analyst : FinancialAnalyst
system : ReportingSystem
secSystem : SecuritySystem
getAvailableReports ( )
getSecurityClearance (userId)
userClearance
availableReports

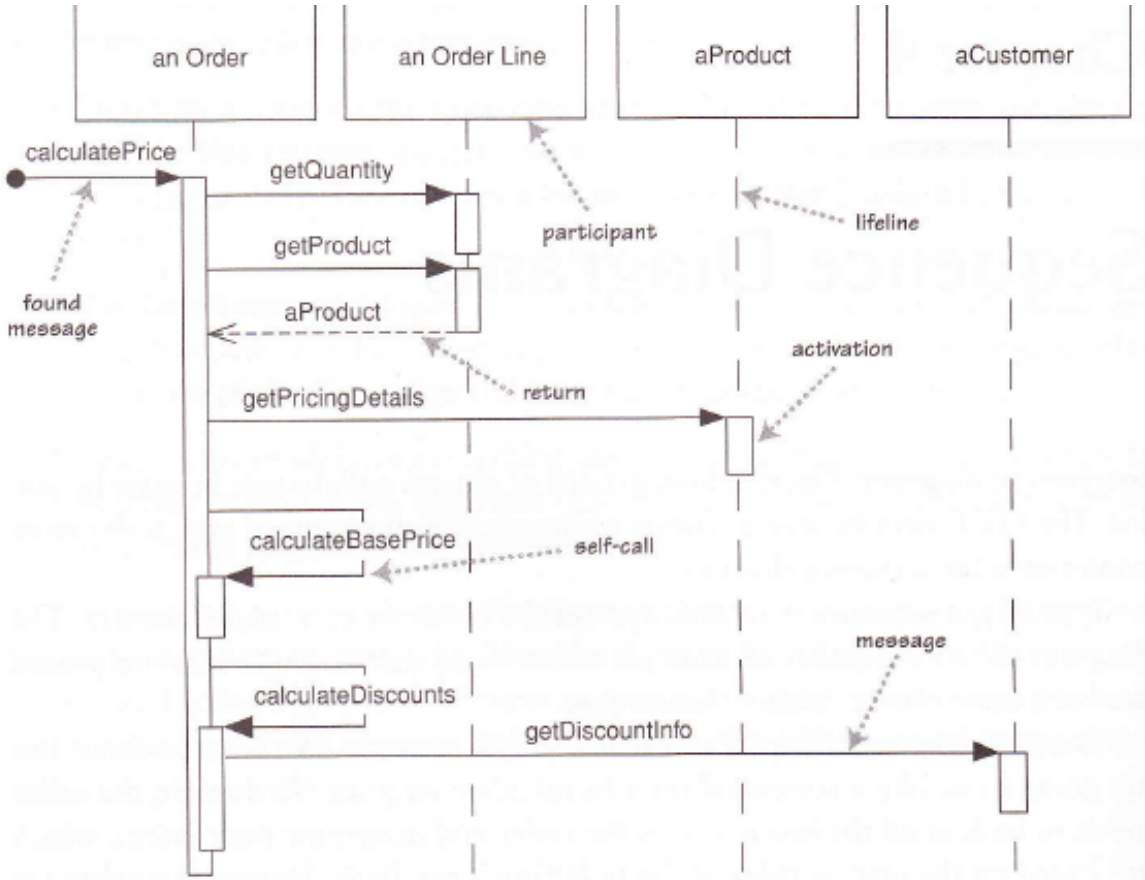
Example 2


sd Balance Lookup (int accountNumber) : int
buyersBank : Bank
ledger : AccountLedger
buyersAccount : CheckingAccount
getBalance (accountNumber)
retrieveAccount (accountNumber)
buyersAccount
getBalance ( )
balance
balance

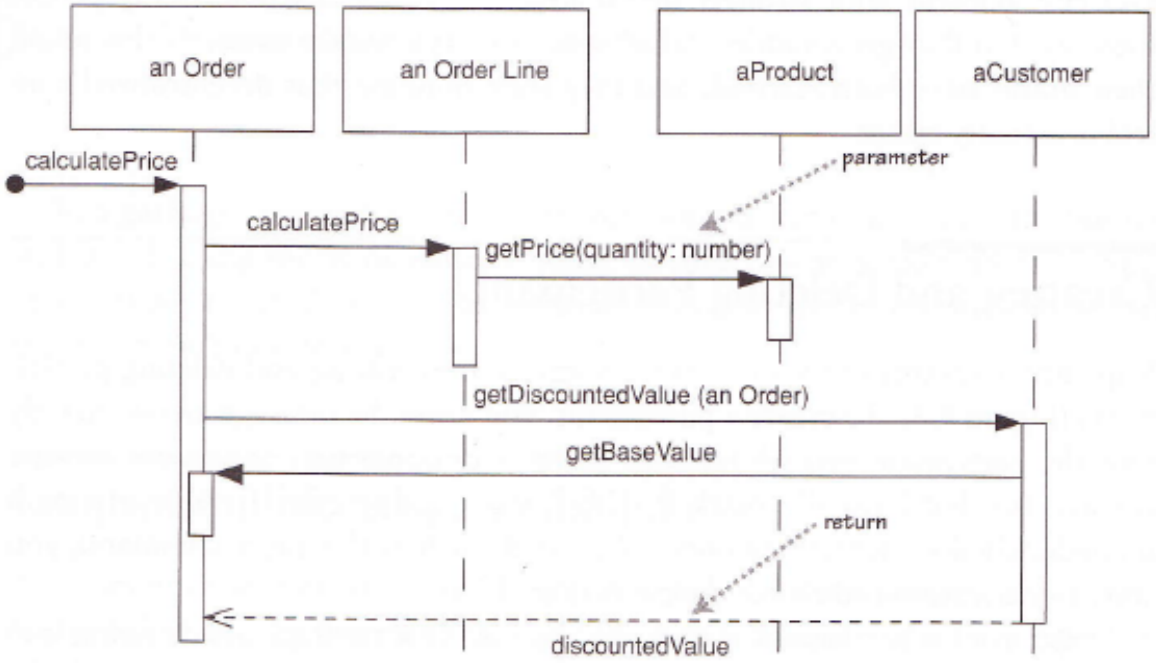
Fowler's Examples Fig 4.1 vs Fig 4.2

- Same classes, same functionality, but different "sequences".
- Consider this in pairs...

1.


an Order
an Order Line
aProduct
aCustomer
calculatePrice
getQuantity
participant
lifeline
getProduct
found message
aProduct
activation
return
getPricingDetails
calculateBasePrice
self-call
message
calculateDiscounts
getDiscountInfo

2.

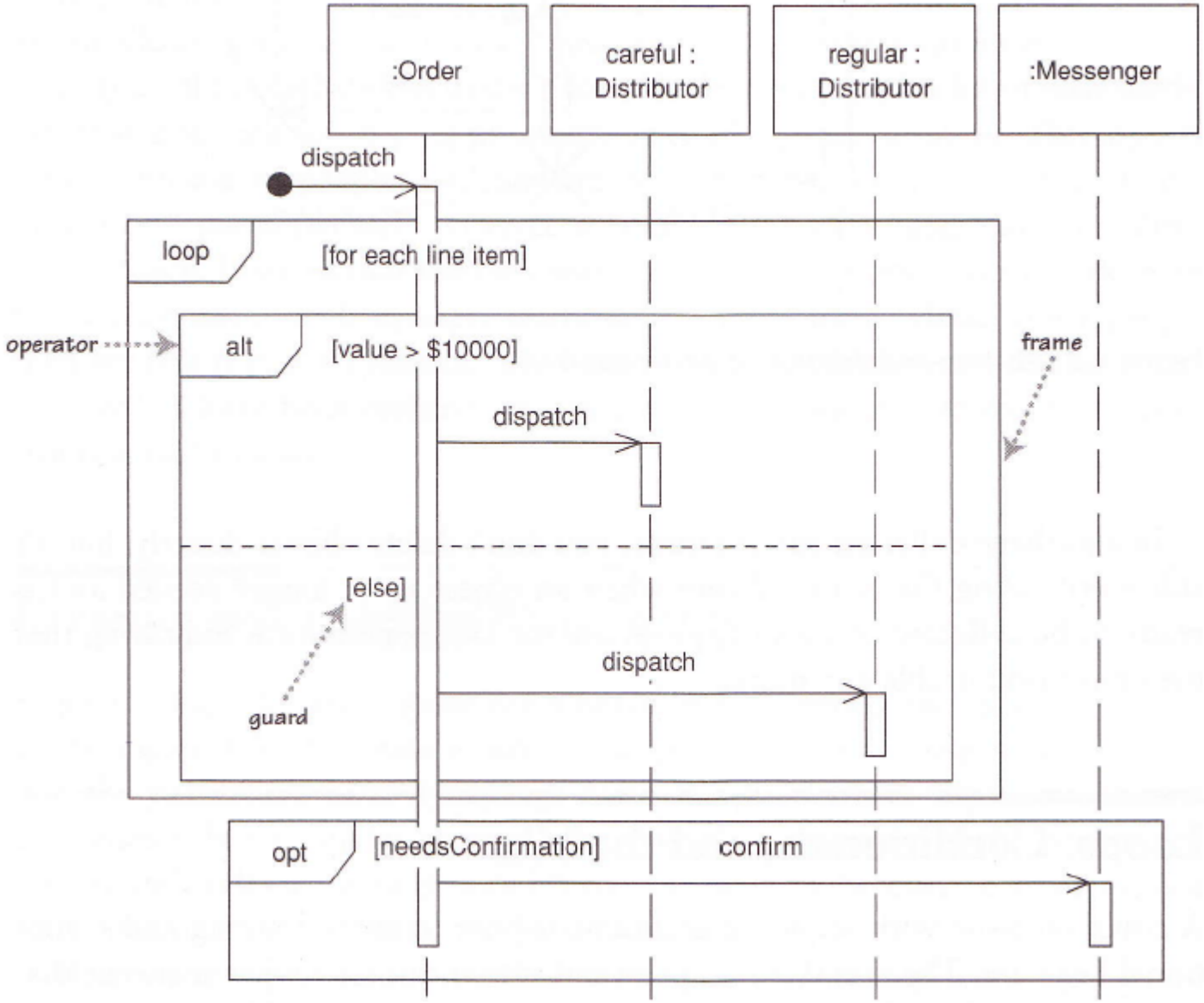

an Order
an Order Line
aProduct
aCustomer
calculatePrice
parameter
calculatePrice
getPrice(quantity: number)
getDiscountedValue (an Order)
getBaseValue
return
discountedValue

Interaction Frames

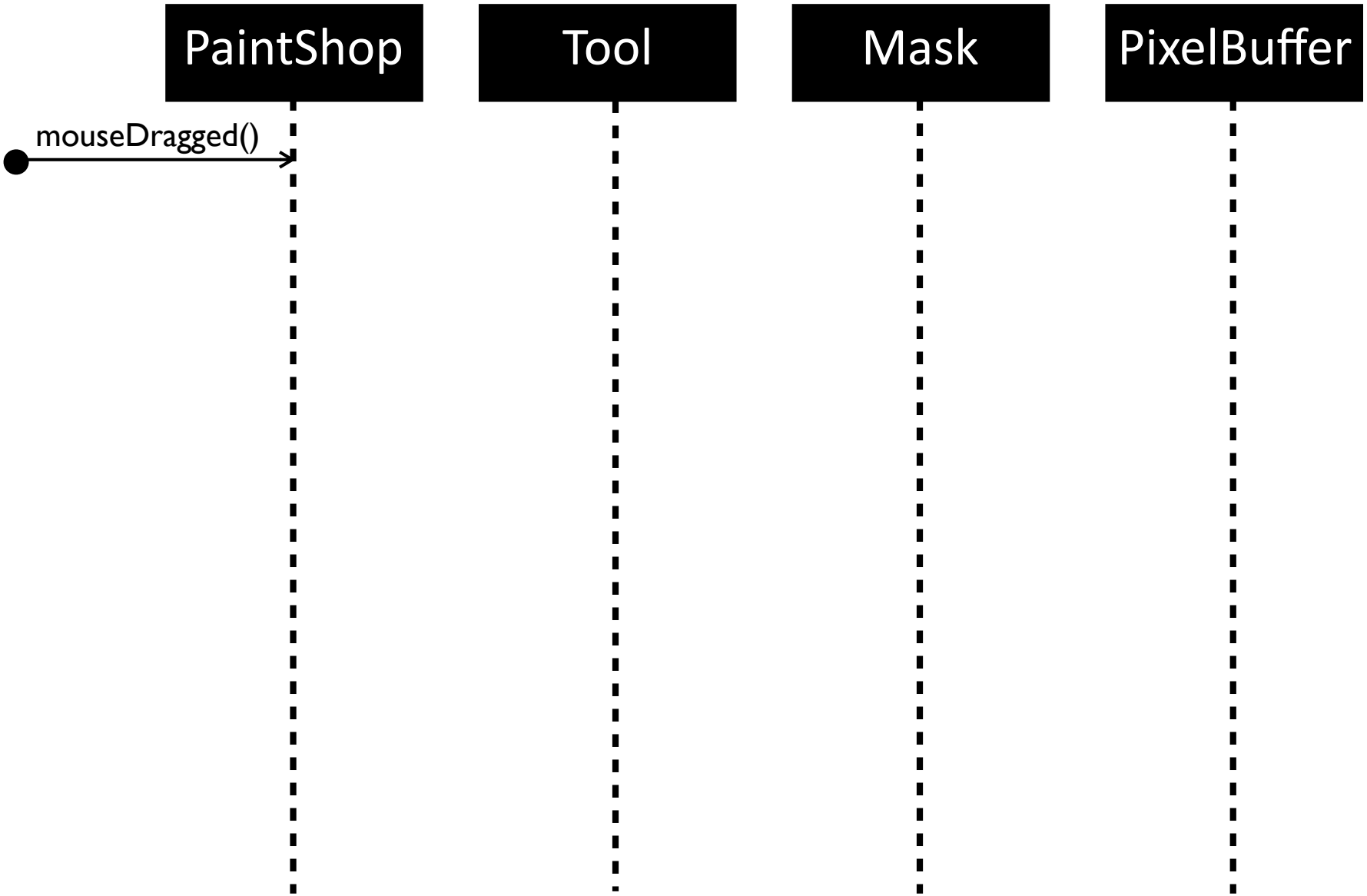
- Used for:
  - looping
  - conditionals
  - etc..

Fowler Fig. 4.4

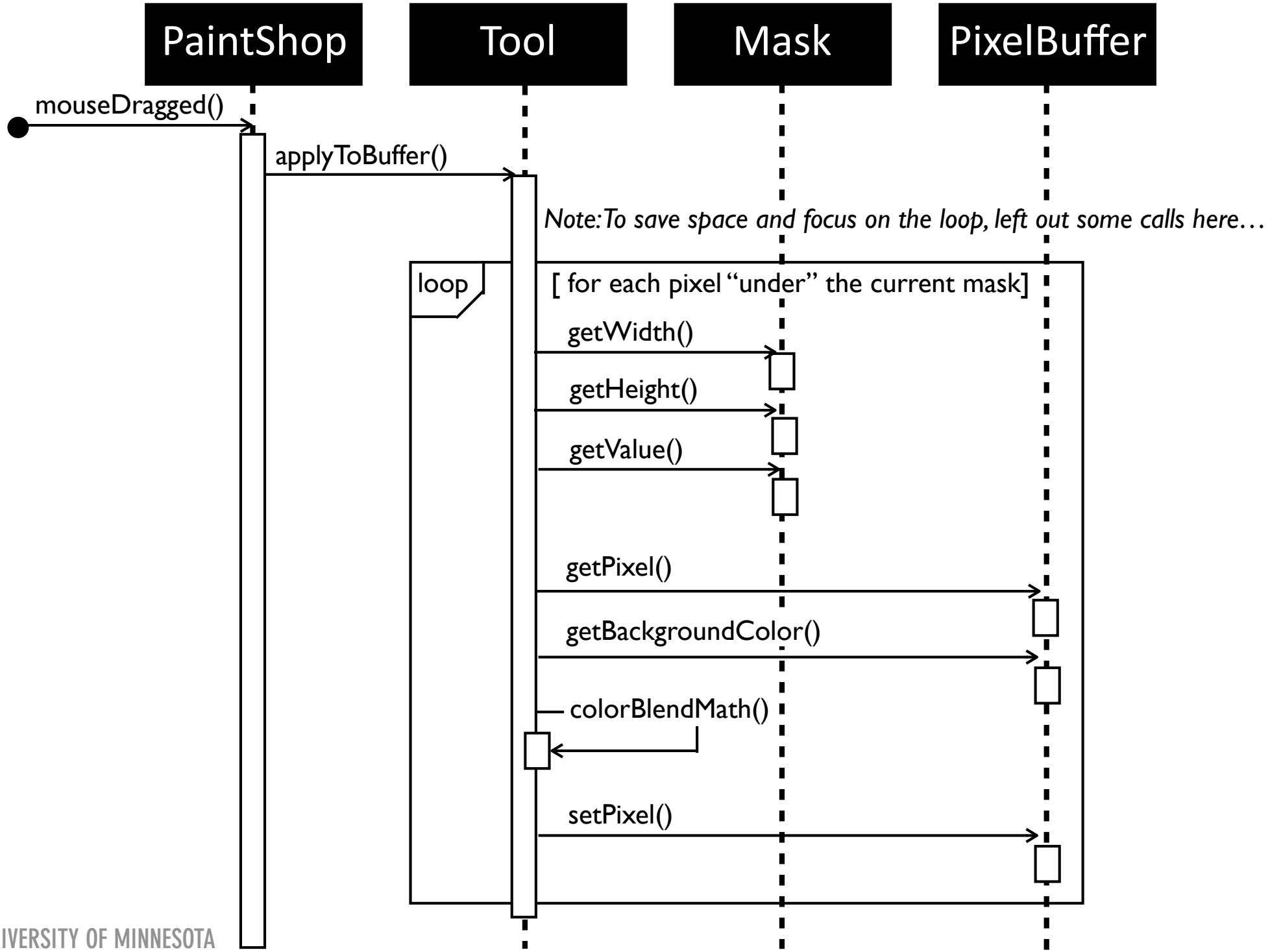
```
procedure dispatch
  foreach (lineitem)
    if (product.value > $10K)
      careful.dispatch
    else
      regular.dispatch
    end if
  end for
  if (needsConfirmation) messenger.confirm
end procedure
```



Sequence for Applying a Tool to Canvas



```
void Tool::applyToBuffer(int toolX, int toolY, ColorData toolColor, PixelBuffer* buffer) {

   if (m_mask == NULL) {
      return;
   }

   int left_bound  = std::max(toolX - m_mask->getWidth()/2, 0);
   int right_bound = std::min(toolX + m_mask->getWidth()/2, buffer->getWidth()-1);
   int lower_bound = std::max(toolY - m_mask->getHeight()/2, 0);
   int upper_bound = std::min(toolY + m_mask->getHeight()/2, buffer->getHeight()-1);

   for (int y = lower_bound; y <= upper_bound; y++) {
      for (int x = left_bound; x <= right_bound; x++) {
       int mask_x = x - (toolX - m_mask->getWidth()/2);
       int mask_y = y - (toolY - m_mask->getHeight()/2);
       float mask_value = m_mask->getValue(mask_x, mask_y);

       ColorData current = buffer->getPixel(x, y);

       float slimmed_mask_value = powf(mask_value,3);
       ColorData c = colorBlendMath(slimmed_mask_value, toolColor, current,
                                    buffer->getBackgroundColor());

       buffer->setPixel(x, y, c);
      }
   }

}
```


PaintShop
Tool
Mask
PixelBuffer
mouseDragged()
applyToBuffer()
Note:To save space and focus on the loop, left out some calls here…
loop
[for each pixel "under" the current mask]
getWidth()
getHeight()
getValue()
getPixel()
getBackgroundColor()
colorBlendMath()
setPixel()

Putting these ideas together…

Using diagrams effectively in your Iteration #3 writing

- What types of diagrams can you use and where?
- What guidelines can you infer from our uses of diagrams and figures in class?
- How about from the examples you found online today at the beginning of class?
- How do you weight the importance of these diagrams relative to writing in a more narrative form?