**COMP 3002 Winter 2025 Assignment #1a**

**Getting Used to Smalltalk**

First lecture Mon Jan 6, 6pm, Assignment due date Sun Midnight Jan 12.

Note that I’ve provided a SMALLTALK CHEAT sheet should you want to look something up. Ultimately, something in the cheat sheet can be found by asking the browser for something appropriate (the internet is NOT a good place to look)

**Starting Up**

0. Assuming you are running on a Windows machine, the first thing to do after you unzip the Smaltalk folder into one of your folders such as a folder called “3002.2025” is to read the file “**SETUP INSTRUCTIONS.txt**”. It tells you how to setup the system’s **PATH** variable to refer to Smalltalk. DO NOT RESET YOUR PATH VARIABLE (THIS CAN DESTROY YOUR SYSTEM), INSTEAD ADD A NEW ENTRY. If you did it correctly, Smalltalk should start up when you double click on “**VDEVW.EXE**”. If it starts to come up but hangs there, the only thing I can think of that could cause that is that your default language for your machine is not English.

1. Use the Windows key + I keyboard shortcut to open the Settings app. (On my computer, the 4 keys on the bottom left are CTRL, Fn, the Windows key, and ALT).
2. Click Time & language.
3. Click Language.
4. Under Languages, make English the temporary default while you use Smalltalk.

**Just the very basics**

1. Create a workspace to work in by selecting “Smalltalk\New Workspace”. The advantage of a workspace is that you do not have to declare your variables. Try it by executing

test := 1

To execute it, you must type it in, select it, right click it to bring up a pop-up menu and select either “**Do it**”, “**Show it**”, or “**Inspect it**”. The first executes it without showing anything. The second executes it and prints the result (which is selected allowing you to easily delete it by clicking on the delete key). The third brings up an inspector allowing you to additionally look inside the object.

Now try the same thing in the Transcript window (the light blue-green window). You will get ‘**undeclared**’ appearing in front of it. Note that it is selected allowing you to easily delete it by clicking on the delete key. To do this without an error message in the Transcript window, you will need to write the following instead

| test | test := 1

Now try it. Most programming languages require your variables to be declared including Smalltalk (but Smalltalk doesn’t want to know what type of variable it is).

**Using inspectors to tell what something is!**

Open an inspector on each one of the following to find out what it is (the title at the top of the inspector tells you what it is). You can double click on any field to further look inside it.

nil true false 'hi' #hi $h 10 10.2 #(10 20)

**Two simple classes just to get the hang of Smalltalk**

2. The following question is to ensure you know how to use the browser to create classes

with instance variables and to create both instance and class methods (and consequently instance

and class categories). Also, you need to know how to create get and set methods for your instance variables

and how to use those get and set methods..

Implement a temperature class so that you can ask for one by saying

aTemperature := Temperature centigrade: 10.

and change it be saying

aTemperature centigrade: 30.

and so that you can compare them as follows

(Temperature centigrade: 10) = (Temperature centigrade: 10)

(Temperature centigrade: 10) = (Temperature centigrade: 20)

(Temperature centigrade: 10) = 'hello'

Also, make sure that a temperature object can print itself the same way you made it; e.g., the first

one you made above should print itself as “Temperature centigrade: 10”.

Also, make sure you can ask a temperature object for its hash. All primitive objects have a hash,

so you can make the temperature object’s hash be the hash of what it contains. Why do we care?   
 If an object has a hash, you can use it as a dictionary key.

(Temperature centigrade: 10) hash

which means you could write something like

dictionary := Dictionary new  
 at: (Temperature centigrade: 10) put: 'cool';  
 at: (Temperature centigrade: 100) put: 'hot;

yourself.

Providing you have access to this dictionary, you should be able to ask

(dictionary at: (Temperature centigrade: 100))

As an aside, even though all the above uses integers like 10 and 20,

can you create a temperature object with a float or a fraction like ¾ in it.

Provide example methods to illustrate what you can do with your temperature object as I’ve done in question

2 below.

3. Create a class Truck with the following example class method that will run. You run it by selecting the code inside the comment on the first line. Anything this needs to make it work, you will have to add.

example1

"Truck example1"

| aTruck |

aTruck := Truck new driver: #Jim.

aTruck addPassenger: #Tom; addPassenger: #Dayton.

aTruck addLoad: #Wheelbarrow; addLoad: #Ladder; addLoad: #Cement.

aTruck driverDo: [:driver |

Transcript cr; << 'The driver is '; << driver].

aTruck driverDo: [:driver |

Transcript cr; << 'One more time, the driver is '; << driver].

aTruck passengersDo: [:passenger |

Transcript cr; << 'One passenger is '; << passenger].

aTruck loadDo: [:load |

Transcript cr; << 'The back of the truck contains a '; << load].

"Method do: sequences over everything in the truck."

aTruck do: [:anObject |

Transcript cr; << 'In the truck, there is a '; << anObject].

To submit this question for assignment #1, select your two classes and choose “FileOutAll” (in some appropriate menu). It will provide a submenu. Choose the last choice in the submenu, namely VISUALWORKS. It’s the nicest format for reimporting the class into someone else’s Smalltalk.