# Personal Notebook Assignment 2

Jackson Wiebe

3519635

## Problem 1

Created required classes and sub classes.

Had to lookup escape characters for the double quotes to print properly.

Animal Test requirements were a bit unclear for processing. My approach was to create a new class for the test that returns an object of Animal using the sub type as requested by the user. TextIO library used to get character from user terminal.

## Problem 2

Create requested class Book with get and set functions.

Most variables are stored as strings for appropriate information like Author name and Publisher. Year has been chosen to store as a 32bit integer as the information will never exceed the bounds of a 32bit signed number (-2147483648 to 2147483647).

I wrote my main program to demonstrate two method of constructor use. The first constructor simple creates the class with null variables to be assigned later. The second constructor creates the instance of Book and assign the variables in one line of code. More constructor methods could also be created to handle most common cases. Class could also assign better default values with more context of the application.

## Problem 3

Creating an elevator class with different variables and functions.

I contemplated using 2 classes here. Elevator and Building. However the scope of the required functions would not extend past the Elevator so creating class building would be unnecessary.

Garbage collection in Java seems pretty straight forward. When GC() is called the virtual machine will try to free up any extra space from null objects. This function seems to take some time and I could not get the terminal to print nicely when the finalize() function actually ran was unpredictable. It also was recommended [not to write a finalize function](https://wiki.sei.cmu.edu/confluence/display/java/MET12-J.+Do+not+use+finalizers) for a variety of reasons:

* There is no fixed time at which finalizers must be executed because time of execution depends on the Java Virtual Machine (JVM)
* The JVM may terminate without invoking the finalizer on some or all unreachable objects.
* Use of finalizers can introduce synchronization issues even when the remainder of the program is single-threaded. The finalize() methods are invoked by the garbage collector from one or more threads of its choice.
* they increase garbage-collection time and introduce space overheads.

Finalize conditions are unclear for the elevator. Is the finalize() supposed to move the elevator, or simply check it has been returned to the first floor?

## Problem 4

Create a base class Rodent. I created the class with a variety of default functions and a default scientific name that covers all rodents.

Create custom child classes that extend Rodent and override all the default functions. Provided each class of Mouse, Gerbil, Hamster, and Guinea Pig with a variety of over ride functions. Not all child classes have overrides for a variety. Custom parameters were applied to the override functions.

Each child class could also pass a more specific call.

IE:

Rodent.eat(Food food)

Mouse.eat(Cheese cheese)

Gerbil.eat(Seed seed)

Where the you create a parent food, and then child specific foods. You could event pass in a habitat for the constructor which would determine where they sleep:

public Rodent(Habitat habitat)

And then sleep could pull form the specific habitat.

## Problem 5

I started by creating the point class. It holds x and y as integers. I also created the constructor to take 2 integer values. Since no units were given integers seem like a good choice for this problem. The add and subtract methods were a bit unclear to me, were they supposed to return a new point, or simply add X and Y to the object? Lastly I created the Show() method which is a simple printer.

Next I moved to the Shape class. This holds an array of size for the shape bounding box. A virtual box that contains all vertices of any shape. The constructor simply sets all the values to 0 to avoid null situations. Area and Circumference were created with the intention of being overloaded later. Lastly display, this required a few revisions as some object don’t return a circumference or an area. For objects that were created improperly they will result in null values.

Circle, rectangle and triangle classes were created as children to the Shape class and each contain overloads and unique values that pertain to their needs. Calculating the area of a triangle could be improved since the formula is messy and does not return properly when any value is null.

Lastly the main function was created simply as a show and tell to demonstrate all the classes. They have good cases and bad cases to show error handling and my thought process.