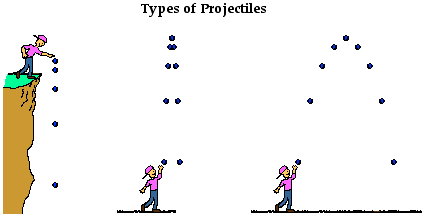
**CS201 Lab 9**

**40 points**  **Due**: 04/7/15

**Problem:** Your cousin Avigail is in her first physics class, but is having trouble visualizing projectile motion. There are three problems she needs help with, which are also demonstrated in the same order in the below figure:

1. How far away something falls if it rolls off a desk
2. Where an object is at a given time if it’s thrown up in the air and then caught.
3. How far something travels if it is thrown at an angle into the air and lands at the same height as from where it was thrown



(image from <http://www.physicsclassroom.com/class/vectors/Lesson-2/What-is-a-Projectile>)

Luckily, you can help her as you know how to create a program that will not only calculate how the items travel, but can also make graphs to visualize it!

**Purpose:** This lab gives you practice with:

* Programming classes
* Using more than 1 class in your program

**Details:**

You need to create a program that will ask Avigail which of the three problems to solve. Then, calculate the answers over time and output the results to System.out as well as outputting to a graph. You will need a calculation class with three methods (NO main method), one for each physics problem calculation. Here is the UML diagram you need to follow:

Plot

(many methods, but I am not listing them)

|  |
| --- |
| PhysicsCalculations |
|  |
| + static distanceRolled(velocity:double, height:double):double  + static distanceUp(time:int, velocity:double): double  + static distanceThrown(theta:double, velocity:double): double |

|  |
| --- |
| Lab9 |
|  |
| + static main(args:String[]):void  - static announce( ): void  - static menu(scan:Scanner): int  - static outputAnswers(menu:int):void |

|  |
| --- |
| ProtectedDataEntry |
|  |
| + static readDouble(scan:Scanner):double  + static readInt(scan:Scanner):int |

*For distanceRolled*: The distance traveled will be calculated based on the initial speed (v0), and how far above the ground the item was when it started falling (h), as assuming that g is the gravity constant 9.8.

*For distanceUp*: The position y of the projectile thrown directly upwards at time t is calculated as , where g is the gravity constant and v0 is the initial speed. Your method should make sure that it returns 0 when the calculated distance is negative (i.e. you can assume the item is in fact caught, and does not continue downward).

*For distanceThrown*: The range the item has traveled is calculated as , where theta is the angle in radians, v0 is the speed at which the item is traveling, and g is the gravity constant. Your method must accept the angle in degrees and convert it to radians using the Math class.

**Steps; We are doing iterative development. It will make your life easier:**

1. Write all of the physics calculations methods:
   1. Create a new Java file for the physics calculations (NO main method)
   2. Write each method following the UML diagram and the above description of the calculations. There should be NO user input or output in this class. You will need to use many methods and pi from the Math class.
   3. Compile until you have no compiler errors on this class. You now have all of your extra classes complete!
2. Complete the main method to run your program. You have parts of 3 methods already written, but need to add code to each of them and add an announce method. This class should also follow the UML diagram. Look over what is there and then do the following:
   1. Write the announce method.
   2. Complete the menu method. This should call another class’s method as part of it (NOT physics calculation class). Compile until you are error free.
   3. Add the first problem to the correct case in your outputAnswers method. You should output the results for initial speeds 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, with height always as 20. Compile. Test that it works.
   4. Add the second problem to the correct case in your outputAnswers method. You should output the results for time 1-10 with initial speed of 50. Compile. Test that it works.
   5. Add the third problem to the correct case in your outputAnswers method. You should output the results for angles 0-90 in increments of 5, with an initial speed of 10. Compile. Test that it works.
   6. Modify main so that the outputAnswers method is continuously called until the user chooses to end the program (there should be some specific input value that means to end). All you need to add is 1 line of code.
3. Add Plots. After all three problems work correctly, modify them such that you also output your results to a scatterplot. The Plot class does this for you. You should create the following plots:
   1. For problem 1, Plot plot1 = new Plot("distance", 0, 50, 5, 0, 100, 5);
   2. For problem 2, Plot plot2 = new Plot("distance", 0, 10, 1, 0, 150, 10);
   3. For problem 3, Plot plot3 = new Plot("distance", 0, 100, 5, 0, 12, 2);
   4. In each problem’s loop, call the addPoint method for each of your value pairs. The addPoint method from Plot takes 2 parameters: xvalue and yvalue. In all plots, the distance is the y value.
4. Write comments in your code to make it clear what it is doing.
5. Write comments for each method in your code.
6. Include an updated version of the class header comments. The Physics class needs a new set!

**Submit:**

1. To GitHub:
   * All Java files
2. On paper in class:
   * A hardcopy of the classes you wrote code (1 copy per pair)
   * A short (250 words or less) individual reflection about what you learned, and how pair programming worked for you this week.

Header comments for classes without main:

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*Programmers: [your name here]*

*Course: CS201.0X, Dr. Olsen [replace X with your section, 1 for 10AM, 2 for 11AM]*

*Date: [due date]*

*Lab Assignment: [number]*

*Purpose: [what is the purpose of this class? What does it represent or calculate?]*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/*