**Lab 18 – T-Shirt Launcher**

Open BlueJ, and create a new BlueJ project titled **Lab18-TShirtLauncher** in your CS\LABS folder.

Create a new class and **type** in our code skeleton:

//Name:

import java.util.\*;

public class PracticeProblems

{

public static void main(String[] args)

{

Scanner console = new Scanner(System.in);

}

}

Remember, methods are named blocks of code (miniature programs) that ***return*** a value. The information they need to run are supplied inside the parentheses, and are referred to as the method’s ***parameters***.

**Before each problem, insert a COMMENT with the problem number.**

1. Get the value of two double variables xand yfrom the keyboard, and write the code to evaluate the following expression AND PRINT THE ANSWER AS AN INT USING CASTING >>> π **(x6 - y6)**
2. Get the value of a double variable rfrom the keyboard, and write the code to evaluate the following expression AND USE CASTING TO ENSURE REAL DIVISION OCCURS >>> **4/3** π **r3**
3. Get the value of a double variable zfrom the keyboard, and write the code to evaluate the following expression >>> **|z4 - 1007C**

//The vertical bars (referred to as "pipes") mean absolute value

1. Get a new value for x and write the code to evaluate the following expression >>> **log |1 + x|**
2. Get a new value for xand write the code to evaluate the following expression AND PRINT THE ANSWER AS AN INT USING CASTING >>> **x2ex**

//Your code should use the value of the mathematical constant e

1. Get the value of two double variables a and b from the keyboard, and write the code to evaluate the following expression >>>
2. (Riddle) 
3. Airport runways are often given numbers determined by the direction in which planes travel as they move along the runways. The number of a runway is found by taking the bearing in degrees (to the nearest ten degrees) and dropping the final zero. For example, a runway with a bearing of 268° would have a runway number of 27. Write the code that asks the user for a bearing (from 0° to 360°) and then determines (prints) the corresponding runway number.
4. Get the value of integer variables i*,* j*,* and kfrom the keyboard. **After collecting the 3 numbers, use exactly one line of code** to determine and print the smallest value entered.
5. Complete the ‘Math methods and type casting’ worksheet (Excel file, in the labs folder). Make sure to include it when you turn in your folder.

**T-Shirt Launcher app**

Create a new class and **type** in our code skeleton:

//Name:

import java.util.\*;

public class TShirtLauncher

{

public static void main(String[] args)

{

Scanner console = new Scanner(System.in);

}

}

T-Shirt launchers are fun, and Lone Star would like to incorporate one at football games to bolster spirit and fan participation at games. As such, Lone Star’s cheerleaders purchased a t-shirt launcher, and upon using it for the first time, promptly launched all the t-shirts out of the stadium. They then realized that you must set the launch **velocity** (how fast they will travel) prior to shooting, in order to launch shirts the correct **distance**.

They also plan to use it at volleyball games, so another factor must be considered – the height of the arena (t-shirts launched directly into rafters don’t improve spirit).

As it doesn’t make sense to be performing calculations during a game, the cheerleaders have asked you to write a program that will quickly and easily calculate the **trajectory** of launched shirts.

Your program should calculate, given a launchAngle and a launchVelocity, the complete 'path' of the **projectile** (calculated every second), until the t-shirt reaches the ground (its y-position returns to 0).

Using this information (and the size of the field / arena of course), the cheerleaders will be launching shirts accurately in no time! To make calculations simpler, we’ll be assuming that air resistance friction is negligible and that the t-shirts are being launched from the ground. Sample program run (**user input shown in red**):

Enter launch velocity (m/s) >>> **30**

Enter launch angle (degrees) >>> **75**

Projectile's path:

Time: 0 seconds

x-pos: 0.0 meters

y-pos: 0.0 meters

Time: 1 seconds

x-pos: 7.764571353075622 meters

y-pos: 24.077774788672052 meters

Time: 2 seconds

x-pos: 15.529142706151244 meters

y-pos: 38.3555495773441 meters

Time: 3 seconds

x-pos: 23.293714059226865 meters

y-pos: 42.83332436601615 meters

Time: 4 seconds

x-pos: 31.05828541230249 meters

y-pos: 37.511099154688196 meters

Time: 5 seconds

x-pos: 38.82285676537811 meters

y-pos: 22.388873943360224 meters

Time: 6 seconds

x-pos: 46.58742811845373 meters

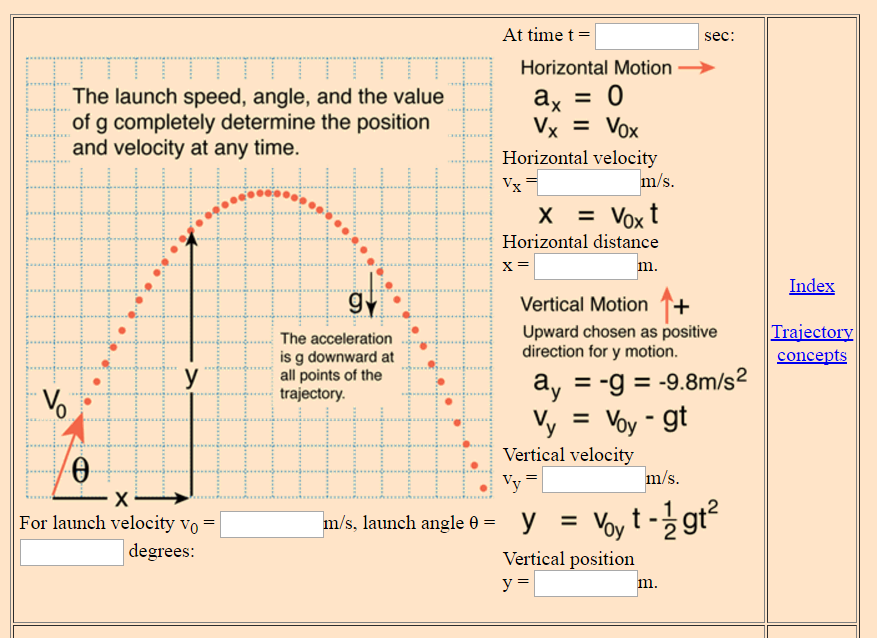
y-pos: 0.0 meters

If you haven't taken physics yet, you're probably going to need some help with these calculations! If you *have* taken physics, try to work out the calculations on your own first.

Help with the calculations can be found below.

**Help with the t-shirt launcher calculations**

To find the X and Y components of the t-shirt's displacement (distance) at a particular time, you need to first find the X and Y components of the t-shirt's velocity (the distance traveled divided by the time it took), then apply projectile motion equations. Here is an excellent graphic with the basic equations of ballistic trajectory:



[*http://hyperphysics.phy-astr.gsu.edu/hbase/traj.html#tra3*](http://hyperphysics.phy-astr.gsu.edu/hbase/traj.html#tra3)

A t-shirt's position (distance) can be found at any particular time with the following equations (with time in seconds, and using 9.8 for the gravity constant):

xPosition = cosine of launch angle \* time \* launchVelocity;

yPosition = sine of launch angle \* time \* launchVelocity – 0.5 \* 9.8 \* time2

Note that the Math.cos() and Math.sin() methods expect a value in **radians**! To calculate the sine of a 60 ***degree*** angle, you’ll need to convert the value in degrees to radians, like this:

Math.sin(Math.toRadians(60))

The method call Math.toRadians(60) **returns** the value of 60 degrees in radians, which is then passed to the Math.sin() method, that will **return** the sine of the **parameter**.

After some time, the t-shirt will hit the ground (its Y position goes back 0). T-shirts don't generally burrow into the ground, so we'll say the loop is finished at this point – print the final X location.

Pseudocode:

Get starting launch angle and velocity from the user

While the t-shirt hasn't hit the ground

Print current stats (time, location, etc.)

Increment time

Calculate current X and Y position

NOTE: Remember t-shirts don't generally burrow into the ground. They stop moving down when they hit the ground. Even though this equation (it’s a parabola) extends the y value into the negatives, you can just manually force it back to zero since we know the t-shirt won’t go underground.