CS 160 – 02/03/04, Spring Semester 2018

Project 1
Compute the Projectile Motion

DUE: February 9th, 2018

Learning Outcomes

- Recognize and apply the software development phases (i)(k)
- Utilize Java syntax in fundamental programming algorithms (a)
- Recognize and apply the various input and output devices in programming (c)

Objectives

In completing this project, you will gain experience with the following Java features:

- definition of a Java class
- variable declaration and assignment
- JOptionPane dialog boxes for input and output
- the Math class
- the String class
- the String.format( ) method and the printf( ) method
- selection logic
- arithmetic expressions and data manipulations

Problem Statement

We encounter projectile motion whenever we attempt to catch a fly ball, hit a long drive, or shoot a free throw. The laws of physics, which describe projectile motion, are well known. They were first studied by Napoleon's generals.
To simplify the problem, we will assume that we can

- consider the surface of the Earth to be a plane
- ignore the effect of the drag from air friction

Under the simplifying conditions above, the following formulas hold for a projectile with an initial velocity of **v** feet/sec and a launch angle of **r** (note that these expressions are not written according to the syntax of the Java language).

1. Time span of the flight: $T = 2\,\mathbf{v}\,\mathbf{\mathit{sin}}\,\mathbf{r} / g$
2. Maximum height: $y_{max} = (\mathbf{v}\,\mathbf{\mathit{sin}}\,\mathbf{r})\,t - \frac{1}{2} g t^2$, where $t = T/2$
3. Range (distance traveled by the projectile): $x_{max} = (\mathbf{v}\,\mathbf{\mathit{cos}}\,\mathbf{r})\,T$

Here g is the gravitational acceleration $g = 32$ ft/sec^2 (in English units) and **r** is measured in radians.
This project simulates the process of repeated attempts to hit a target with a projectile. The goal is to shoot the projectile within a 1-foot distance from the target, since such a short miss is accepted as a hit. You will construct a Java program that

- can determine the trajectory data of a projectile for a given initial velocity and a launch angle, in particular for an initial choice of a 45-degree angle

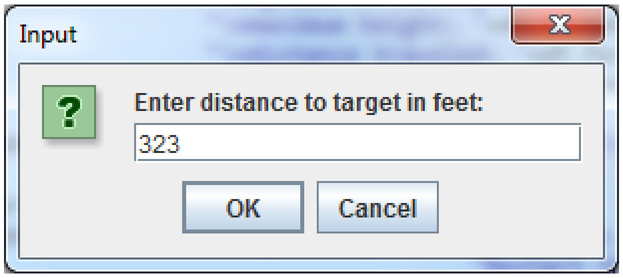
- can check if the initial attempt overshoots the target with more than 1 foot; if so the process is terminated and then restarted with an increased initial velocity (note that for any initial velocity the longest range is attained if the launch angle is of 45 degrees)
- can determines the error of a shot (error = projectile range – distance to target)
- can check if the error is less than 1 foot in absolute value; if so, the user is notified about the result and the process terminates
- can offer the user four chances to modify the launch angle and try to hit the target
- can keep track of the smallest error produced by the subsequent attempts; the best result is reported to the user

Analysis and requirements

The analysis describes the logical structure of the problem in a way which helps us to plan (design) a solution.

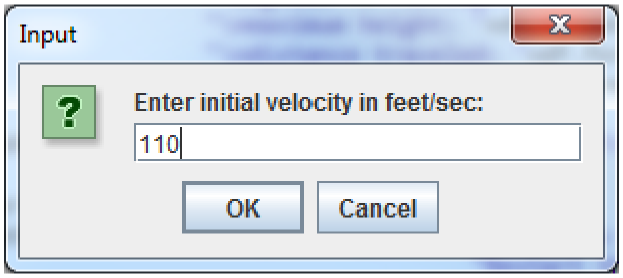
Input

Initial input values are
(i) **initial velocity** (feet/sec)
(ii) **distance** to the desired target (feet)
(iii) the gravitational acceleration (a constant)
Additional input values are the launch angles for the repeated attempts if applicable. The angle must always be in the range of 0.0 – 45.0 degrees.
Distance, velocity and launch angle are solicited from the user on JOptionPane input windows. See Figures 1, 2, 3.



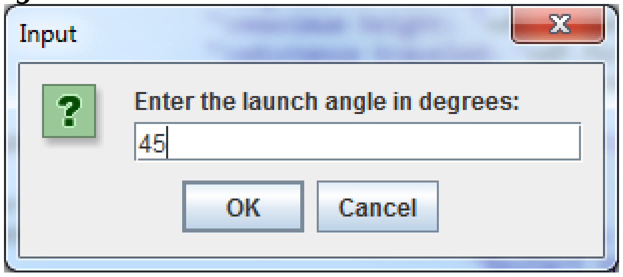
Enter distance to target in feet:
323

Figure 1



Enter initial velocity in feet/sec:
110

Figure 2

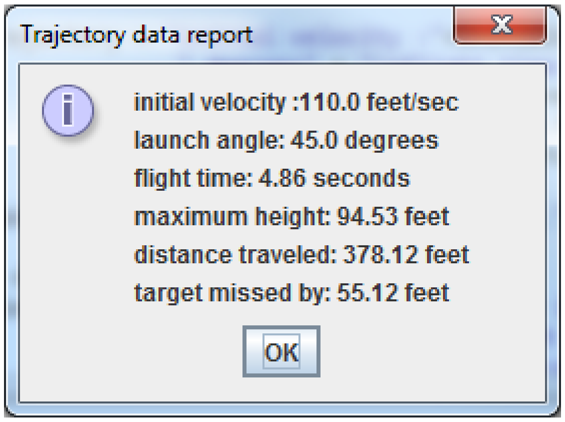


Enter the launch angle in degrees:
45

Figure 3

Output

Output messages are displayed both on JOptionPane windows and on the console. Every time a launch has been executed by the program, a report providing the details of the trajectory must be displayed as shown on Figure 4.



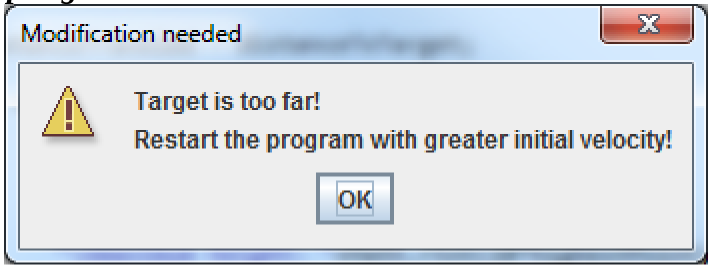
```
initial velocity :110.0 feet/sec
launch angle: 45.0 degrees
flight time: 4.86 seconds
maximum height: 94.53 feet
distance traveled: 378.12 feet
target missed by: 55.12 feet
```

Figure 4

Each report must contain

(1) the initial velocity in feet/sec;
(2) the current launch angle in degrees;
(3) the flight time in seconds;
(4) the maximum height attained;
(5) the distance traveled by the projectile (range);
(6) the error with which the projectile missed the target

Note that the error is a positive value when the projectile overshoots, and negative for short shots.
All real numbers displayed must be rounded to the nearest hundredth.
The first report is based upon launch angle of 45 degrees (which provides the longest possible range) to see if the target is within reach at all for the given velocity. If the first attempt falls short of the target, another window as shown on Figure 5 displays the information and then the program exits.



Target is too far!
Restart the program with greater initial velocity!

Figure 5

If the first shot is long enough, the window of Figure 3 shall be used to input all subsequent launch angle modifications as chosen by the user. The corresponding reports of Figure 4 show the re-calculated trajectories. Naturally, the user will try to modify the angle so as to make the projectile land nearer and nearer to the target. After each unsuccessful attempt a warning is printed to the console, see a sample on Figure 6.

```
Shot fell short of the target. Increase the launch angle!
```

Figure 6

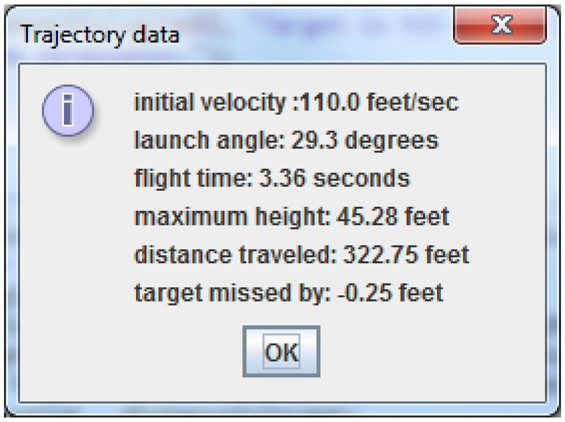
Figure 7 shows a sample output on the console after all four angle modifications failed to hit the target.

```
Shot went beyond the target. Decrease the launch angle!
Shot went beyond the target. Decrease the launch angle!
Shot fell short of the target. Increase the launch angle!
Shot fell short of the target. Increase the launch angle!
Your best shot missed the target with 4.47 feet.
```

Figure 7

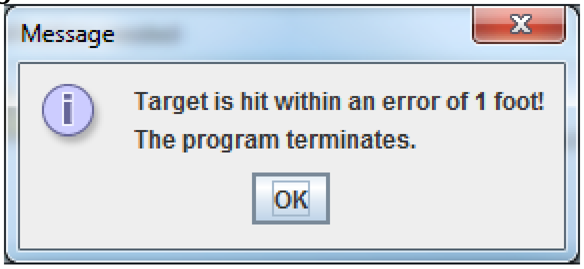
Note that it is necessary to keep track of the least absolute error occurred in the series of attempts, since it has to be reported as the last line in Figure 7 shows.

Figure 8 shows a successful launch. Such a report is followed by the message of Figure 9.



initial velocity :110.0 feet/sec
launch angle: 29.3 degrees
flight time: 3.36 seconds
maximum height: 45.28 feet
distance traveled: 322.75 feet
target missed by: -0.25 feet

Figure 8



```
Target is hit within an error of 1 foot!
The program terminates.
```

Figure 9

Relevant Formulas

The basic formulas 1, 2 and 3 in the **Problem Statement** will be used for all trajectory computations.
The input value for a launch angle shall be solicited and given in degrees. The previous formulas shall be implemented by making use of the trig methods Math.sin(angle) and Math.cos(angle) from the Math class. These methods require the parameter angle in radians. Therefore an angle given in degrees must be converted to radians. The conversion formula runs as follows:

(angle in radians) = (angle in degrees) π /180

Note that it can be used in either direction, from degrees to radians or vice versa.
The value of π (PI) is available as a named constant **`Math.PI`** from the Math class of the Java library.

Design

For this project you shall design a single class which contains all the necessary data and operations. A suggested class name is **`Projectile`**. You must decide upon the necessary import(s).

The **`Projectile`** class contains the main method, which in turn contains all the variable declarations.
The main method should carry out the following tasks in the order given:

- declares and assigns a named constant for the gravitational acceleration; the value is 32
- solicits and stores the input values as explained in the Analysis section
- computes all the trajectory data and saves those in variables
- builds up and stores the output message in a single string variable (content of Figures 4, 8)
- displays the output windows
- numbers in the output are formatted to the nearest hundredth; for this purpose the String.format( ) and printf( ) methods are used
- uses if and/or if-else logic to decide if additional launches are necessary and repeats the operations at most for times if needed
- terminates the program when it is due

See also the detailed class description below. You may declare any additional local variables in the main method.

Projectile	
METHOD	
main	The method opens input windows (see Figures 1, 2, 3) to obtain the initial data. The input values are assigned to the corresponding data fields. The method computes the flight time, maximum height and range. The results displayed on a message window (see Figure 4). Displays Figure 5 – 9 as needed After the last attempt, the program exits.
Local variables in main	
GRAVITATION	Named constant, the value is 32.
distanceToTarget	double; to store the distance in feet to the desired target (input)
initialVelocity	double; to store the projectile's initial velocity in feet/second (input)
launchAngle	double; to store the current input for launch angle in degrees (input)
radian	double; to store the value of angle in radians (calculated)
flightTime	double; to store the flight time of the projectile as defined by formula 1 in the description (calculated)
highestPoint	double; to store the maximum height of the projectile as defined by formula 2 in the description (calculated)
distanceTraveled	double; to store the distance traveled by the projectile as defined by formula 3 in the description (calculated)
error	double; to store the difference between distance traveled and distance to target (calculated)
minError	double; to store the least absolute error; updated every time a new error value was generated (calculated)
trajectory	String; to store the current output message built upon the trajectory data

Testing

Test your program with input values shown in the description. See if your results match the output data.

Requirements and Hints

1. Having your program tested, run the program with a new set of input data of your own choice. The distance to target must be at least 500 feet. Copy your trajectory report as a comment block after the class code.
2. You must have a comment block preceding the header of each of your classes that has the following content:

```
/*
 * <your name>
 * CS160 - 01 Spring 2018
 * Project 1: Compute the Projectile Motion
 *
 * Description. <Summarize the purpose of the class here.>
 *
 */
```

3. You are NOT allowed to use iteration structures (loops) to code the repeated attempts. The relevant part of the code that calculates and displays the trajectory results must be copied four times into the program. Here is the pseudo-code version of the code you must copy four times in your program:

```
if error positive
    send overshoot message to the console
else
    send undershot message to the console

solicit new input for launch angle
calculate radian
re-calculate all trajectory data
calculate error
update minError
compose output message
display output message

if absolute error less than 1
    display hit on window
    terminate the program
```

4. Use the Math.abs( ) method to calculate the absolute value of the error, when you make a decision about the accuracy of the shot.
5. Set up a variable minError as suggested in the class description. Every time the program generates a new error, update minError such that it stores the least absolute error occurred this far. Use the Math.abs( ) and Math.min( ) methods for this purpose.
6. Use the Math.sin( ) and Math.cos( ) methods for the sin and cosine functions.
7. The first input for launch angle must be 45 degrees, you choose the four additional modifications

8. Do not round the numeric values while needed in calculations. Apply formatting only when the output message is composed

Evaluation

Documentation and style: 10 points.

Your program must conform to the Computer Science Department's Java Documentation and Style Requirements. Emphasis will be placed on having the required banner and internal comments, indentation, and overall stylish appearance. Comments must be clearly written with correct grammar and spelling. Do not neglect the documentation and style.

Correctness: **90 points**.

These points will be allocated as follows:

1. Error-free compilation of the Java source file…………….….…… 10 points
2. Correct declaration of the variables………………………………....20 points
3. Correct implementation of the projectile formulas………………….15 points
4. Correct use of selection logic………………………………………..15 points
5. Correct application of dialog windows……………………………...10 points
6. Correct display of correct formatted output messages ……..……….20 points

Total: **100 points**