**Balloon**

**Analysis**

The balloon program allows a user to input two angles of observation, and then the program proceeds to calculate the first observation height, the increase in height to the second observation, the speed of the balloon and the distance to the balloon at the second observation. The extended requirements however, allows the user to enter the two heights and then the program proceeds to calculate the two angle of observations, speed and distance to the balloon.

**Design**

There will be three classes; Balloon, BalloonExtended and BalloonUser. Balloon will contain four methods to calculate the four output values, as will BalloonExtended. BalloonUser will contain one method which will read in the values given by the user and pass them to Balloon. It will then output the values that Balloon or BalloonExtended calculates.

**Class Diagram**

|  |
| --- |
| Balloon |
| - angle1: double  - angle2: double  - distance: double  - time: double |
| + h1(): double  + h2(): double  + speed(): double  + balloonDistance(): double |

|  |
| --- |
| BalloonUser |
|  |
| + main(String[]) |

|  |
| --- |
| BalloonExtended |
| - height1: double  - height2: double  - distance: double  - time: double |
| + a1(): double  + a2(): double  + speed(): double  + balloonDistance(): double |

**Pseudo Code**

Public class BalloonUser

Main = string args

Create new scanner

Get user choice of program

Start IF

1 = mode1

End IF

Start else

2 = mode2

End else

Method = mode1

Call Balloon class

Create new scanner

Set angle2, distance and time values

Get angle1

Start IF

State error value if not between 0 and 90

End IF

Start else

Get angle2

End else

Start IF

State error if angle2 isn’t between 0 and 90

End IF

Start else

Get horizontal distance

End else

Start IF

State error if less than 0

End IF

Start else

Get time

End else

Start IF

State error if time less than 0

End IF

Start else

Create new Balloon class

Output values

End else

End

Method = mode2

Call BalloonExtended class

Create new scanner

Set height2, distance and time values

Get height1

Start IF

State error value if not above 0

End IF

Start else

Get height2

End else

Start IF

State error if height2 not above 0

End IF

Start else

Get horizontal distance

End else

Start IF

State error if less than 0

End IF

Start else

Get time

End else

Start IF

State error if time less than 0

End IF

Start else

Create new BalloonExtended class

Output values

End else

End

Public class Balloon

Create constructor

Method = h1

Calculate distance\*tan(angle1)

Return value

Method = h2

Calculate distance\*tan(angle2 – angle1)

Return value

Method = speed

Calculate h2/time

Return value

Method = balloonDistance

Calculate distance/cos(angle2)

Return value

Public class Balloon

Create constructor

Method = a1

Calculate arctan (height1/distance)

Return value

Method = a2

Calculate arctan ((height2/distance)+tan height1)

Return value

Method = speed

Calculate h2/time

Return value

Method = balloonDistance

Calculate distance/cos(a2)

Return value

**Testing**

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected** | **Actual** |
| Use integer values;  30, 50, 100 and 40 | Height: 57.7  Increase: 61.4  Speed: 1.5  Distance: 155.5 |  |
| Uses real values; 14.7, 11.1, 67.43 and 44.123 | Height: 17.6  Increase: -4.4 (showing a decrease in the height)  Speed: -0.1 (showing a decrease in the speed)  Distance: 68.7 |  |
| Use extreme values; 0, 0, 10000 and 10000 | Height: 0, Increase: 0, Speed:0, Distance: 10000 |  |
| Use extreme values; 90, 90, 10000 and 10000 | Height: 1.6, Increase: 0, Speed: 0, Distance: 1.6 |  |
| Use the values the original program uses to see if my extended works | Angle1: 30,  Angle2: 50,  Speed: 1.536,  Distance:155.5 |  |
| Test that user selecting 1 takes them to the correct program | Balloon |  |
| Test that user selecting 2 takes them to the correct program | BalloonExtended |  |