Regis University CC&IS
CS210 Introduction to Programming
Alice Programming Assignment 5: Loops

Program Requirements

This Alice program will contain 2 scenes. In the first scene, two cars will pop wheelies multiple times. In the second scene, the two cars will race towards an object in the middle of the scene and see who wins.

Create an Alice program that ***minimally*** does the following:

Setting the Scene

- Start with a blank slate.
- Add two sports car objects (from the Automobile class, within the Transport class) to the scene, and adjust the car heights to both be size 1.
- Turn the cars so they are facing each other.
- Use the middle camera adjustment arrow to zoom out a bit so you can see more of the scene (click on arrow at least 5 times).
- Move the cars to either side of the screen, at the same Z-axis position -- one on the far left side of the screen facing right, and one on the far right side of the screen facing left (make sure both cars are fully visible when you run your code).
- Place an object from the Prop class in the middle of the scene between the two cars, also at the same Z-axis position. Make the Prop object fairly small.

Creating Methods

- Add two ***scene*** methods, **doScene1** and **doScene2**.
- Add a ***shot*** method called **race**, which will run from within the **doScene2**.
- Within the **Automobile** class, add a **procedure** method called **popWheelie**, which causes an automobile to pop a wheelie.

Coding the Actions

Add code to the **popWheelie** method to make a sports car:

- Simultaneously:
  - Spin (turn) its back wheels several rotations.
  - Raise the front end of the car (turn) a small amount off the ground.
- Return the car to the ground.

Add code to the **doScene1** method to:

- Create a variable and store the ***half of the height*** of the Prop object in the variable.
- In a **count** loop, make the Prop object perform the following tasks three times in a row:
  - Spin around 360 degrees.
  - Move up by half its height
  - Move down by half its height
- Prompt the user to enter how many times the cars should pop wheelies.

- In a **while** loop, validate the number of times entered:
    - If the number of times entered is less than 1
        - Have one of the cars say that it must pop at least one wheelie
        - Read another value from the user

      Loop until a valid value is entered (1 or more).
- In a **count** loop:
    - Simultaneously invoke the **popWheelie** method for both cars, to make both cars pop the specified number of wheelies, at the same time.

Add code to the **doScene2** method to:

- Create a variable and store the distance from the ***left*** car to the Prop object in the variable.
- Create a second variable and store the distance from the ***right*** car to the Prop object in the variable.
- Have the each car say how far it is from the Prop object.
- Create a variable **moveLeftCarDistance** to hold the distance the ***left*** car will move each time, and set it to be ***one tenth*** of the distance from the left car to the Prop object.
- Create a variable to hold a ***move factor*** for the ***right*** car.
- Have the ***right*** car ask the user what factor to use to move it, and explain that it must be a double between 5 and 15. Store the user input into the ***move factor*** variable.
- Within a **while** loop
    - Test to see whether the ***move factor*** the user entered is invalid (i.e. ***between 5 and 15***)
    - If the value entered is ***invalid***
        - Have the right car say why the factor entered is incorrect
        - Prompt for and read another value into the ***move factor***.

      Loop until a valid value is entered.
- Create a variable **moveRightCarDistance** to hold the distance the right car will move each time, and set it to the value of the distance from the right car to the Prop object ***divided by*** the ***move factor***.
- Invoke the **race** method to make the cars race to the Prop object.
    - Pass the two ***move distance*** variables as arguments to the **race** method.

Add code to the **race** method to:

- Take 2 parameters
    - The first parameter defines the amount that the left car will move each time.
    - The second parameter defines the amount that the right car will move each time.
- Create a Boolean variable that will indicate whether the race should continue, and set it to an initial value of ***true***.
- In a **while** loop that runs as long as the boolean is ***true***:
    - Move the left car forward by its parameter amount
    - Move the right car forward by its parameter amount.
    - Check to see if the left car reached (collided with) the Prop object (Hint: use built-in **isCollidingWith** method). If so:
        - Have that car say that it won the race.

        - Reset the Boolean flag so that the loop will exit

      Otherwise, do nothing and continue with the next check.

- Check to see if the right car reached (collided with) the Prop object. If so:
    - Check to see if the left car ***also*** reached the prop object (i.e. the boolean was reset already). If so:
        - Have that car say that the other car did not win, but they tied instead.

      Otherwise:
        - Have that car say it won the race.
        - Reset the Boolean flag so that the loop will exit

    Otherwise (right car has not reached Prop), do nothing and loop.

This loop should run until one (or both) car(s) reaches (collides with) the Prop object.

Note that you can test all possibilities for this loop by adjusting the ***move factor*** entered in Scene 2 to make each possibility occur.

Make sure the duration of each movement in your animation is long enough to view comfortably.
Run, test, and debug your Alice program, until it works correctly.

Extra credit (5 pts)

Optionally you may add one or more methods that will use a loop to cause the objects to perform some other behavior more than once.

NOTE: The new behavior cannot be a behavior implemented by any of the course materials.

The method(s) should be invoked from an additional **extraCreditScene** method.

Warning: Do not attempt to implement the extra credit unless you have first successfully implemented the normal part of the assignment. ***You will not receive any extra credit if the normal part of the assignment is not functioning properly.*** Be sure to save a copy of the normal part of the assignment, in case your efforts on the extra credit are unsuccessful.

Submission

This programming assignment is due by midnight of the date listed on the Assignments page.

Submit your program source code (the **.a3p** file) to the **Alice Prog Assn 5** dropbox (located under the **Dropbox** tab in the online course).

Before submitting your program file, you MUST re-name it as follows:

`LastnameAliceAssn5.a3p`
For example: **`SmithAliceAssn5.a3p`**

Grading

The rubric that will be used to grade your program is linked on the same assignment page that you downloaded this file from.

WARNING:
*Programs submitted more than 5 days past the due date will **not** be accepted, and will receive a grade of 0.*