# CS 51 Homework Laboratory # 13
## Duck Shot

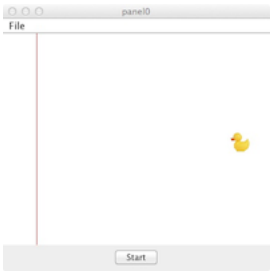
**Objective:** To gain experience with GUI & Threads in Java.

—

**The Problem** This time you will write a Java program from scratch (though you might notice there is a certain similarity of this project with the earlier BoxBall program).

As usual, start by dragging a copy of the starting folder from the server.

Below is a picture of Duck Shot. In the picture, Nibbles is the winding black ribbon that looks a bit like the top of a question mark. The small gray rectangle is the food. A user controls the movement of the snake with the arrow keys on the keyboard. *Important: You must click on the window displaying the snake before your program will respond to the arrow keys!* The snake constantly moves by extending its "head" in the direction indicated by the last arrow key the user presses. While it is hard to tell from the picture, when the picture was taken, the snake's head was the square just below and to the right of the food. It was moving to the left on the screen. The square at the bottom of the question mark is the end of the snake's tail. Normally, each time the head moves into a new cell, the last cell of the snake's tail is removed from the screen. If the snake manages to eat the food, it becomes a few squares longer. It does not grow all at once. Instead, for a few steps after it eats, the snake's head is allowed to move into new squares while none of the squares in the tail are removed, thus simulating the growth of the tail.




The duck and foul line are shown are the screen when the program starts. When the start button is pressed, the duck moves repeatedly up and down between the top and bottom of the screen until it is hit with a paintball. The user fires a paintball by clicking to the left of the foul line (the read line in the picture). The paintball moves horizontally to the right until it hits either the duck or the wall. When it hits either of those, it compresses to half its original width and then falls at a steady speed to the floor. When it hits the floor it compresses again to 1/4 of its original height and then remains on the floor for the rest of the game. The game is over when a paintball hits the duck. A message should be displayed when the duck is hit and the duck should stop moving. You do not have to stop the player from throwing paintballs when the duck is dead.

**Design** The DuckShot program consists of 3 classes:

- `PaintBallGame` - the window controller that initializes the display and handles the user input, namely responding to the start button and creating and starting the paintballs.
- `PaintBall` - a class used to create and animate the paint ball image. Because it creates an animation it will extend built-in class `Thread`.

- `Target` - a class used to create and animate the duck. Like `PaintBall`, it will extend `Thread`. It also supports a method that determines whether a filled oval overlaps with the duck.

To make your life simpler, we will provide you with the code for the class `Target`. You are responsible for implementing `PaintBallGame` and `PaintBall`.

**Implementation** Here are some hints on how to approach the problem of implementing this program.

Start by constructing by drawing the foul line and creating and installing the start button. Please put the start button in a `JPanel` at the SOUTH end (bottom) of the window. The `PaintBallGame` should be set to be the listener for the button.

Next you should create the target. The Target class constructor takes a parameter of type `Image`. This is a built-in Java class from the `java.awt` library. There is a method defined in all classes extending `WindowController` with the following signature:

```
public Image getImage(String imageFile) {...}
```

The duck image is in file `duck.jpeg`, so you put the string `"duck.jpeg"` in as the parameter to get the `Image`. The `Target` class will use that image to create a `VisibleImage` of the duck that will be installed on the canvas.

Add the `actionPerformed` method to `PaintBallGame`. When the user presses the start button the target duck should start moving.

Finally, you will need to write the `PaintBall` class. Each paint ball should have a color that is randomly generated. The objectdraw library contains a class `RandomIntGenerator` that can be used to create new random integers. For example, if you wanted to create random numbers between 1 and 6, you would write

```
RandomIntGenerator gen = new RandomIntGenerator(1,6)
```

Then each time you want a new random number you would evaluate `gen.nextValue()`. The idea is that you create only a single generator, but then use it as often as you like. In this case you will be choosing numbers between 0 and 255 (inclusive) for the red, green, and blue components of the color. A color is created in Java using the construction `new Color(r,g,b)` where `r`, `g`, and `b` are the components for red, green, and blue.

First see if you can get the ball to move right, hit the right edge of the canvas, and then slowly fall down the wall to the floor. Once that is working, work on getting it to ask the target if it has hit it each time it moves to the right.

If it hits the target, then the target should stop moving and it should display a message indicating that the game is over.

**Submitting Your Work** When your work is complete you should export the project and then deposit in the dropoff folder a copy of the entire `Lab13-DuckShot` folder. Before you do this, make sure the folder name includes the phrase "Lab 13" and your name. Also make sure to double check your work for correctness, organization and style. This assignment is due Wednesday, the last day of classes, at 4 PM.

Table 1: Grading Guidelines

| Value | Feature |
|---|---|
| | **Readability (5 pts total)** |
| 2 pts. | Descriptive comments |
| 1 pts. | Good names |
| 1 pts. | Good use of constants |
| 1 pt. | Appropriate formatting |
| | |
| | **Code Quality (6 pts total)** |
| 2 pt. | conditionals and loops |
| 2 pt. | General correctness/design/efficiency issues |
| 2 pts. | Parameters, variables, and scoping |
| | |
| | **Correctness (9 pts total)** |
| 1 pt. | Start button starts duck moving |
| 1 pt. | Ball starts when click behind foul line |
| 1 pt. | Randomly colored balls |
| 3 pt. | Ball moves & falls appropriately |
| 2 pt. | Duck stops when hit |
| 1 pt. | End of game message |