The purpose of this assignment is to draw and manipulate a model of a frog on the canvas (a pop-up graphical window). The code you create could be used in a much larger game/simulation.

You are required to write a complete class and to use several provided classes to build a BlueJ project. The full mark for the assignment is 10, but there are Bonus/Challenge extensions that are worth up to 4 extra marks.

The assignment is to be submitted via WebCT before **noon** on **October 14**. Late assignments will be accepted until noon on October 17, but a 20% penalty will be assessed (and no bonus marks for late assignments). Assignments submitted later will *not* be graded.

The class you will write is called `Frog`. It has fields, two constructors, and methods described below. Three of the fields of `Frog` are objects of the `Triangle` and `Circle` classes. These are classes from the BlueJ `shapes` project accompanying Chapter 1 in the text which you should use as the basis of your project.

WARNINGS:
1. You must not modify the classes in the `shapes` project. You will hand in only your `Frog.java` file.

2. Your class **must** include properly formatted JavaDoc comments. Be sure to test the generation of your documentation.

3. If you have not yet done so, carefully read the Course Ethics section on the course web page. Note especially the remarks concerning **plagiarism**, which is a serious academic offence.

Do not *expect that small changes in a program (such as altering comments, changing variable names, or interchanging statements) will avoid detection. If you cannot do the work yourself, it is extremely unlikely that you will succeed in disguising someone else's work.* Students who ignore this warning have been detected and penalized.

Class Frog

A `Frog` object has a name, a head (triangle), two eyes (circles), an h-position and a v-position (horizontal and vertical). The h-position and a v-position refer to the top vertex (tip) of the triangle. When visible, the `Frog` is on the canvas (the pop-up window). This canvas is 300 pixels by 300 pixels. The top left of the canvas is at position (0,0) and the bottom right is at position (299,299). Several of the methods will require `if` statements, and the `jump` method will require the use of loops. Note that the default position (set by the constructor) for a `Triangle` object is (50,15), and for a `Circle` object it is (20,60).

Fields

`name` A `String`, for example `"Kermit"` or `"Greenie"`.

`head` A reference to a `Triangle` object.

`leftEye` A reference to a `Circle` object.

`rightEye` A reference to a `Circle` object.

`hPos, vPos` Two `int` values that hold the h- and v-positions of tip of the triangle object.

Constructors

There are two constructors for the `Frog` class. The first constructor takes no parameters and does the following: sets the frog's name to `"Kermit"`; creates a `Triangle` object referenced by `head`, and changes the triangle's width and height to 105 and 35 respectively; sets `hPos` to 150 and `vPos` to 100 and moves the triangle so that its tip is at the canvas location $(x, y) = (150, 100)$. Then it creates two `Circle` objects referenced by `leftEye` and `rightEye`, and changes their color to `"black"` and their size to 21; the eye circles are positioned so that their centers are 15 pixels down from and 20 pixels to the left and right of the tip of the triangle.

The second constructor takes *three* parameters: the first is a `String` indicating the frog's name, the second and third are integers giving the horizontal and vertical positions of the frog object (that is, the tip of the triangle). This constructor does what the first constructor does, except that the frog's name and initial positions are set using the parameters.

Note: For the second constructor, if the `vPos` or `hPos` locations passed via the last two parameters would cause *any part* of the frog to be off the canvas, use the default location $(x, y) = (150, 100)$ as in the first constructor.

Methods

The `Frog` class has six methods. The first, called `getName`, is an accessor method that simply returns the `String` holding the name of the frog. The methods `makeFrogVisible` and `makeFrogInvisible` do exactly what their names indicate. They are void methods without parameters. The `frogLeft` and `frogRight` are `void` methods that take a single `int` parameter which must be strictly positive. The methods move the frog horizontally left or right by that amount. If a non-positive parameter is supplied nothing is done. If the new position would place any part of the frog off the canvas, move the frog just to the edge of the canvas. Otherwise, move the specified number of pixels. The last method is `jump`. This method takes a single `int` parameter indicating a jump height (must be strictly positive! - if not do nothing). If the jump height would move any part the frog off the canvas then the jump height is reduced so that it makes the frog just touch the top of the canvas. The method moves the frog vertically upwards *one pixel at a time* and then downwards *one pixel at a time* to the original location.

Bonus/Challenge Problems

There are two directions you may pursue for extra credit. You may choose to work on only one or a combination of both. If you wish to submit work for bonus marks you should make a new

project called `FrogPlus`, starting again from the `shapes` project and containing a new class that you should name `FrogPlus`. Once again, you must not modify the other classes of the `shapes` project.

Bonus marks will be awarded on the basis of both the quality of your implementation (including documentation) and on its creativity. Remember that your class **must** include properly formatted JavaDoc comments.

The first challenge is to enhance the appearance of a `Frog` object. This can be done by adding, for example, a body, legs, facial features, a hat... This will require both that you add new fields and that you modify the constructors and methods of the `Frog` class. Note that you may also use the `Square` class from the `shapes` project to express some features. It is expected that the two constructors described above will again be available, but you may add new constructors with appropriate parameters. You *must* also ensure that *all* of the methods from the `Frog` class described above are implemented in `FrogPlus`.

The second challenge is to enhance the movements of the frog. There are two suggested methods and you may try others (be sure to fully document all of your methods!). The first method `slowDiagonal` takes takes two parameters, an `int` called `distance` and a `String` called `direction`. The valid values for `direction` are `"NE"`, `"SE"`, `"NW"` and `"SW"`. The method should move the frog slowly (one pixel at a time) in the indicated direction on the canvas by a diagonal distance of `distance` pixels but leave the frog at the wall if the distance would take it off the canvas. The second suggested method is called `diagonalBounce`. It takes the same parameters as `slowDiagonal`, but this time if the frog hits the wall it should reflect from the wall (for example hitting the wall in a `"SE"` direction means the frog will continue in the `"SW"` direction).

Submitting Your Assignment

You should place your results in a zipped file called `usrnmA1.zip` (containing only `Frog.java` or both `Frog.java` and `FrogPlus.java`) and submit it using WebCT.

Don't forget to include proper headers and Javadoc-style comments.