**Design Document:**

The project is organized amongst two java files, Assignment13.java, which contains all the actual game code, and ExampleProjects.java, which contains the big bang statement and some tests.

We are using/importing the following referenced libraries:

* geometry.\*;
* impsoundworld.\*;
* tunes.\*;
* java.util.\*;
* java.awt.Color;
* jvav.awt.Rectangle;

The game can be started by ensuring the run configuration’s main class has been set to ExampleProjects, so that it can enter the public static void main method and run the bigbang statement.

World

GameObjects

Frog

FrogWorld

The code of this Frogger game is organized into three classes: Frog, GameObjects, and FrogWorld. The following is a brief description of how each class contributes to the game’s overall flow and logic:

The **Frog** class represents the object that is the main character of the game and player controlled; the frog. Each frog holds information about it’s position on the game canvas, it’s radius, and even the filename of the image it loads. Furthermore, the instance field that actually contains the frog’s png image (“this.image”), also stores the image’s exact width and height, measures which prove quite useful later on.

* FrogImage( ) – This method produces a WorldImage of the current instance’s frog object. All it needs is the filename of the png image and the center Posn object, which will become the x and y coordinates of the frog’s location. This method is essentially used every time the frog is drawn and redrawn after updating, in a future method called makeImage( ).
* moveFrog( ) – Every time the frog is moved, either through user interaction or because of the river current, this method is called. It passes in one argument, the String ke, which contains one of the four possible string representations of the directional keys. However, before actually mutating the frog’s center posn, the method decides whether or not there is a collision in the location the frog is trying to move to. If that passes, the frog is moved using simple arithmetic mutations to the center posn’s x and y coordinates.
* leftWallCollision( ), rightWallCollision( ), topWallCollision( ), lowerWallCollision ( ) – All of these methods, which end up being called in the method moveFrog( ), make sure that the frog the user is controlling stays within the bounds/constraints of the game board. Returns a Boolean whether or not a collision will happen.
* GameObjectsCollision( ) – This method is passed one argument, a GameObjects g, which is checked for collision with the frog object. The way collisions are detected is by drawing rectangles around both images and seeing if they intersect. This method returns a Boolean depending on whether or not a collision has occurred. In the context of the entire game, this method is called on each vehicle currently active on the game board during onTick( ).
* logCollision( ) – Same as the GameObjectsCollision( ), but the collision checking code is more specific towards log objects. Takes and returns the same arguments as GameObjectsCollision( ).
* onLog( ) – Given an ArrayList of an ArrayList of GameObjects, in this case logs, onLog returns a Boolean corresponding to whether or not the frog object is on the log.
* frogInRiver( ) – Returns a Boolean that tells us whether or not the frog is inside the river area.
* frogWin( ) – Returns a Boolean that determines whether the frog is located in the winning area.

The **GameObjects** class represents all the objects in the game that interact with the frog and the rest of the game world, specifically those objects that are automated in their actions and movements. A GameObject has a lot of instance fields that are exactly similar to those in the Frog class, such as the string that holds the filename, the center Posn that holds the current location of the GameObject, and the image of the GameObject. It also, however, has fields that are specific to the GameObjects class, such as starting positions from the left and right, a string that represents direction/orientation and the speed for the automated movement.

The GameObjects class has the following two methods:

* objectImage( ) – Produces a worldImage of the GameObject at its current location, a.k.a the Posn center. Similar to the FrogImage method in the Frog class, the objectImage method is used in the later method makeImage.
* move( ) – This method updates the automatic movement based on the direction of this GameObject. If the GameObject gets to the end of the left or right screen, its movement is restarted from the opposition position.

The **FrogWorld** class, which extends the World class, is the core of the game that runs and displays our frog and GameObjects, automating what’s necessary and taking keyboard inputs for user inputs. The FrogWorld class contains numerous instance fields, such as the width and height of the gameboard, the number of lives the frog currently has, all the gameObjects currently in play, and even the notes to the frogger theme (just to name a few). So as to make this document as concise as possible, lets go over the implemented methods from the World class briefly before going further:

* The onKeyEvent method simply takes an input key and then either ends the game if the “q” key has been hit, or sends the input key to the moveFrog method otherwise.
* The onTick method is what controls all of the automated parts of the game, as well as checking for collisions, playing the frogger theme song, and checking whether the game has been won yet.
* The worldEnds overrides the worldEnds method in the World class. This method is automatically called ever tick in the onTick method. It simply checks to see if the game should be ended. If the game is over, it calls the method lastImage to display a game over or congratulations screen.
* lastImage overrides the lastImage method in the World class. It returns a black screen with the given message overlaid on top.
* The makeImage method brings runs through the game objects, frog, and background, overlaying them so they display properly every frame.

Here are methods that have not been implemented from the World class:

* initGameObjects( ) – Initializes the instance field arrays with all of the logs and cars/trucks.
* accumulateWinners( ) – If the frog has reached the end of the obstacle course, it is added to the arraylist of winning frogs in this method.
* frogOnWater( ) – This method determines if the frog is in the river area and whether or not it is situated on a log. If the frog is not on a log, the frog is removed, the lives are decremented, and a new frog is created. Otherwise, the frog is moved in the same direction and speed as the log it is sitting on.
* moveObjectsRiver( ) – Calls the move method on all the river GameObjects.
* moveObjectsRoad( ) – Calls the move method on all the road GameObjects, but firsts checks if there is a collision.