.July 14, 2018 (Michael Manila)

Class: Game.java

* Created and worked on runGame() method.
  + Initializes all cars and the frog, so that the game starts with all the objects in their starting positions.
  + Prints a 10x10, that prints an F for the frog, an arrow for the cars, and empty space for the rest of the game space.
  + Receives user input so by running moveFrog() method from the Frog class and updates the game accordingly(first moves the frog according to the input, and then updates the car to move forward).
  + If user gets to the top of the board, stops runGame method.

Class: Frog.java

* Frog class that is a representation of the frog in game(the user).
* The frog class has a getter for the frogX and frogY, instance variables of type int that stores the frog’s X and Y position.
* Created a moveFrog() method.
  + Gets user to input w for up , a for left, s for down, or d for right followed by enter. Otherwise, continues to ask for input until a w, a, s, or d is inputted.
  + Method then alters the frogX and frogY based on input.

Class: Menus.java

* Menus class which displays the game menus.
* displayMenu() method prints menu and takes user input, play, instructions, or quit.

July 16, 2018 (Nathan Moton)

Class: Car.java

* Car class that is a representation of the cars in the game.
* The car class has a getter for the carX and carY, instance variables of type int that stores the car’s X and Y position.
* Created moveCarLeft() and moveCarRight() methods.
  + moveCarLeft, moves the car to the left 1 space. If the car reaches the left boundary, it’s X position is moved to the furthest right so we can use the same car to move left, without creating a new instance of car.
  + moveCarRight, moves the car to the right 1 space. If the car reaches the right boundary, it’s X position is moved to the furthest left so we can use the same car to move right, without creating a new instance of car.

July 18, 2018 (Justin Flores)

Class: Game.java

* Added the collisionDetection() method
  + Detects collision by comparing the the frogs X and Y position to the cars X and Y position.
  + Changed some of the runGame() method into instance variables of the class so that the collisionDetection() method can access the objects
  + Added two booleans: hit, and hit2. One for checking if frog jumps onto a car and the other for checking if the user jumps into the immediate path of any of the cars.

July 20, 2018 (Michael)

Class: Frog.java

* Edited moveFrog() method so that the frog will stay inside the boundaries.
  + Detects if the frog is next to the edge of the game space, and if the user tries to go further in that direction, the frog will maintain its position but the move will stay count (cars will advance one position).

July 21, 2018(Michael Manila)

Class: Frogger.java

* Frogger class is the class with the main method, which incorporates all the above classes into a basic game format.

DEMO\_2

Date: July 13, 2018 (Nathan Moton)

Class: Frogger.java

* Creates the game window.
  + Uses Java’s Swing library
  + Creates a perfect 640x480 game window that is centered and on the screen
  + Creates a game engine object that will essentially run and manage the entire game.

Date: July 14, 2018 (Nathan Moton)

Class: Frog.java

* Frog class that is the representation of the frog in the game
  + Has instance variables that shows the frog’s x and y positions
  + The instance variables each have their own setter and getter methods
  + Added move<Direction>() methods which can move the frog right, left, up, or down by adding or subtracting 32 to the frogs x and y position

Date: July 14, 2018 - July 19, 2018 (Michael Manila, Justin Flores)

Class: Log.java, Vehicle.java, Turtle.java

* Obstacle classes that is the representation of the obstacles that the frog must avoid or jump in to.
  + Has instance variables of type double that represent the object’s x position, y position, acceleration in the x direction
  + Has instance variable of type int, which represents the object’s length
  + Has a boolean type instance variables, asking if the object is going right
    - If the boolean is true, then the right moves in the right direction, otherwise, it is going to move left
  + The class has setters and getters for the object’s x and y position
  + Has a getter for the log length and setter for log direction
  + move<Object><Direction>() method moves the object right or left by subtracting its x position by the object’s acceleration

Date: July 16, 2018 - Present (Nathan Moton)

Class: Map.java

* This class creates a map by using instance variables of type int to determine the maps end boundaries and the water zone boundaries.
* It also uses instance variables of type double to determine the frog/user’s starting x and y position.
* It creates a new Frog object using the two type double instance variables as arguments for the new Frog operator.
* The obstacles in the map are stored in 2D arrays:
  + Changing the arguments for the objects or adding more objects in the arrays will allows us to create different maps.

Date: July 13, 2018 - Present (Nathan Moton)

Class: Gameplay.java

* The checkWater(parameters) method checks if the the frog is in the water by comparing the frog’s x and y position to the x and y position of the boundaries. If the frog is in the water, the game window will close.
* checkEndZone(parameters) method checks if the the frog is in the end by comparing the frog’s x and y position to the x and y position of the boundaries. If the frog is in the end zone, the game window will close.
* keyPressed(KeyEvent e) method moves the frog up, down, left, or right in the map when the user presses on the arrow keys. It also prevents the frog from moving out of the map.
* <obstacle>Boundary<Left or Right>Bound() methods resets the obstacle to its origin if it reaches a certain point outside the map. This allows us to have the objects to keep looping through its path, which means that we don't need to create new obstacle objects everytime it moves outside the map.
* vehicleCollision<Right or Left>Bound() method checks if the frog and a vehicle has collided. Both the frog’s x and y position and a vehicle’s x and y position are used to determine if a collision has occur.
* userOn<Turtle or Log><Length of obstacle><L or R>() methods puts the frog into the obstacle if the if has successfully jumped on the obstacle. Also, it keeps the frog in obstacle as it moves across the map. This is done by making sure that the frog’s position is the same as the obstacle position.
* checkLogs() method checks every log in the game
  + If the log has gone out of bounds, it resets it to its origin position by calling logBoundary<Left or Right>Bound() method.
  + It invokes the userOnLog<length><R or L>() method to keep the frog in the log or put the frog in the log when it jumps on it
* checkTurtles() method checks every turle in the game
  + If the turtle has gone out of bounds, it resets it to its origin position by calling turtleBoundary<Left or Right>Bound() method.
  + It invokes the useronTurtle<lentght><R or L>() method to keesp the frog in the turtle or put the frog in the turtle when it jumps on it.
* checkVehicles() method checks every vehicle in the game
  + If the turtle has gone out of bounds, it resets it to its origin position by calling turtleBoundary<Left or Right>Bound() method.
  + It also checks collision by using the vehicleCollision<Right or Left>Bound() method

Date: July 21, 2018 - Present (Nathan Moton)

Class: GameGraphics.java

* Utilizes four pass-by reference so that a map’s log array, turtle array, vehicle array, and frog avatar can be drawn onto the game window.
* Utilizes Swing’s BufferedImage library and Java’s FileIO to include the custom created sprites.

Date: July 23, 2018 (Nathan Moton)

* Used GIMP to re-create sprites based on the original Frogger game sprites.

Date: August 3, 2018 (Michael Manila, Nathan Moton)

Classes: Gameplay.java, Map.java

* Changed checkWater method in Gameplay so that it takes the Y location of the log and turtle arrays instead of water boundaries from the Map class. This will help water detection and will allow for more flexible map creations.
* Removed waterBoundaries variables from map.
* Modified checkLogs method and checkTurtles method so that if the user is not on a log or turtle then checkWater will check the row if the the user fell into the water by using the newly modified checkWater method.

Date: August 3, 2018(Michael Manila)

Class: Gameplay.java

* Added a frogInBounds method to ensure that user dies if the go out of the boundary (by riding logs or turtles).

Date: August 3, 2018(Nathan Moton)

Class: Gameplay.java

* Fixed bugs in the water zone.

Date: August 4, 2018(Nathan Moton)

* Used GIMP to add additional sprites for the frog turning in different directions.
* Added code to make the frog face different directions depending on which button was pressed.

Date: August 5, 2018(Nathan Moton)

* Used GIMP to design menus.

Class: Gameplay.java, GameGraphics.java

* Implemented menus.
* Modified vehicle collisions to be more lenient.
* Fixed fire truck vehicle collision bug.
* Implemented sounds.
* Fixed a bug to do with winning the game.

Date August 5, 2018(Michael Manila)

Classes: Gameplay.java, Map.java, LevelOne.java, LevelTwo.java, LevelThree.java

* Created LevelOne-Three class, classes that extend map and store arrays of the game objects(Logs, Turtles, Vehicles).
* Modified Map class so that level classes contain arrays. Added MapGraphics() method which passes the objects of the map to the GameGraphics class and which can be used by the subclasses to set the arrays.
* Modified Gameplay class so that it has instances of all three Levels.
* Changed arrays in LevelTwo to create a map layout consisting of only Logs.

Date August 5, 2018 (Justin Flores)

Class: Score.java

* An object that has a username and score
* Updates the score of the player
  + Increases the score by 10 if it reaches a y position higher than its highestPosY, and updates the highestPosY to the new highest y position reached.
  + Increases the score by 1000 if the player clears a level
* Ask the user for its username through keyboard inputs

Date August 5, 2018 (Justin Flores)

* Class: HighscoreManager.java & HighscoreManagerTest.java
* This class stores scores and names of players and stores them in an arrayList. It sorts the arrayList based on the player's score or who achieved the score first. it manages how scores are handled/sorted and where they should be stored.
* Sorts the arrayList in an descending or
* For the text-based version, it can display the top 5 scores and the players’ names

Date August 7, 2018(Justin Flores)

Class: LevelThree.java

* Updated the arrayList of obstacles to create a custom map
  + Added more objects to arrayList
  + Changed the y and x position of some objects
  + Changed the speed and length of the obstacles

Date: August 7, 2018(Nathan Moton)

Class: Gameplay.java

* Implemented instant death when user crashes into wall of the end-zone.
* Incorporated short pauses after death before the “Game over” screen appears.

Date: August 8, 2018(Nathan Moton)

* Used GIMP to create new textures.
* Created new map image files.

Class: Gameplay.java

* Fixed bug with game over screen not showing the actual map.