Comments:
- World() will read sprites from ArrayList and depending on the type of the Sprite (there will be if statements that identify the type of the sprite),it will initialize the sprite differently in the constructor. It will also render remaining lives as according to livesLeft, each DISTANCE_DIFFERENCE away from each other. Extra life is created in World() only if ExtraLife class' getIsExtraLifeVisible() is false and readyToCreateExtraLife() is true. Then extra life's coordinates are retrieved, and the sprite is added to the Sprites ArrayList.

- readFirstCSV() will read first line, store that line in readCSV, and process its content. The method will split the String after each comma. The first String (after the split) will indicate the type of sprite, and the rest will be parsed into Integers to store that sprite's coordinates. I will also create variables to hold those integers and the sprite's name (those same variables will be used for each sprite until the CSV file ends). The same logic will be applied to readSecondCSV().

- blackHolesX — I have decided to deal with black holes in a lazy way. Basically, the frogs in blackholes are created from the very beginning, but they are originally offscreen. However, when the player enters a black hole, the frog of that black hole's x coordinates will change and a frog will be rendered at the black hole's coordinates instead.

- setBlackHoleX() verifies which x coordinate did the frog enter. That is also how it will know which hole did the player enter and at which hole should the frog be rendered.

- livesLeft was made public so that other classes could update it whenever the player dies. It also will determine how many RemainingLives will be rendered.

- livesLeft ArrayList will essentially indicate how many lives will be crated and rendered (by World() of course) with the help of livesLeftY to indicate the y coordinate. The arrayList will story the x coordinates of each lives left. So whenever a life is added then ArrayList will add a new element, which will be the sum of the last element + 32 (and simultaneously add a new sprite to Sprite ArrayList). Deletion of the life will result in deleting the last element of the ArrayList and elimination of that sprite from Sprite ArrayList.

- setLivesLeft() ensures that if livesLeft drop below 0 then the game terminates, and allows outside methods to up the value of livesLeft. Moreover, setLivesLeft() will also terminate the game when livesLeft have a value of 1 and the setLivesLeft() wants to reduce livesLeft by 1. If there is more than 1 lives left setLivesLeft() will go through the sprites ArrayList and delete the last remaining lives sprite (so that the live sprite from the left disappears last) and delete the last element of livesLeft ArrayList too.

In addition, each time the value given to setLivesLeft() is negative (so the player died), then the player's icon will
be reset to its offset position by calling Sprite's resetToOffsetPosition()

- readSecondCVS() will be activated only if player succeeds (emptyHolesLeft reach 0 while livesLeft never fall below 0).The method will also reset livesLeft to 3. Moreover, Extra Life will be set to false if it was true when the player leveled up. Basically, this method will reset the ArrayList and fill it anew with the CVS with level 2. It will also set setExtraLifeVisible() to false if getExtraLifeVisible() is true (so that there is no extra life immediately upon starting level 2).

- pickRandomLog() will go through Sprite ArrayList, count how many log and long logs there are, pick a random value (let's call it n) between 1 and the sum of log and long logs, and finally pick nth log/longlog. It will retrieve the coordinates and provide it to the extra life's coordinates (extraLifeX and extraLifeY)

- timeUntilNextExtraLife() calculates how much time have elapsed and updates timeElapsed. If the timeElapsed equals currentRandomValue then new extra life is created by calling readyToCreateExtraLife(), which also appears in the World(). getRandomog() is called and extra life is placed on that one (and simualtnously the new frog is added to the Sprite Array List). generateRandomValue() will be called from the ExtraLife class to generate another value between 25 and 35 and timeUntilNextExtraLife() is reset to the new new random value and cycle repeats.

- deleteExtraLife() deletes the extra life from Sprites Array List.

ATTRIBUTION: THE CODE WAS BASED ON THE SAMPLE CODE RECEIVED FROM THE TUTOR AND NOT ON MY OWN CODE.
I DO NOT OWN THE BASE CODE. I ONLY MADE MODIFICATIONS TO IT.

Comments:
- Essentially it's the same code as the sample code with a few modifications:

- onCollision() checks if the player collided with a lethal object. If the player did so, then the method checks how many lives does player have by calling getLivesLeft(). If the player has any lives left, the x and y will be reset to player's offset positions.

- resetToOffsetPosition() sets x and y to playerXOffset and playerYOffset.

Comments:
- isPlayerOnLog() detects where the Player's Y coordinate is. After all, water spans within the range of y-coordinates 336 to 48, so if the player enters that area and intersects a log (for this, onCollisionSafe() will check the tag. If there is no HAZARD, then it will return true) then it's fine.

- isPlayerWithBulldozer() works similarly as the method above. The bulldozers are allocated on specific Y coordinates, so if the player enters that Y coordinate, then there are no buses for sure and the player will be pushed. However, here if the player attempts to enter on top of the bulldozer, the dx and dy will be set to 0 (update() will calculate if the next move is going to cause a collision between the player and a bulldozer when the player plans to go up)

- IsOnScreen() checks if the player is on the screen, regardless if the player is riding an object or is pushed by a bulldozer.

- moveIfOkay() will move the player depending what isPlayerOnLog and isPlayerWithBulldozer()reply. If the player is on the log, the method will access the log's coordinates and velocity, and will move the player unless the player makes a new move. If the player on the Bulldozer, then the player will be pushed.

- update() and Bulldozer — update() will also check Player's new coordinates. There will be an if statement that will check the player's future y coordinate and compare it to trees' bottom row's coordinate, specifically y = 48. If the player's next move is to push the frog to the trees (the method checks it by calling getY() from Sprite class and adds dy(difference in y coordinate) to it; if the sum is equal to 48, then that means that the player is planning to step into the trees), then the method checks the future x coordinates.

The if statements that guard player's frog to stay on screen will also have another variable — isPlayerWithBulldozer. If the player is on the same y coordinate as bulldozers, the frog WILL be able to go off screen, and the player will die (World's setLivesLeft() will be called to decrease the player's lives left and Sprite's resetToOffsetPosition will reset the player back to his origins).

If new x coordinates (a sum of the value returned from Sprite's getX() and dx (dx = difference in x coordinate)) is equal to that of trees (I will iterate through treesX list and compare each element with x) dx and dy will be set to 0 (so, no change in player's position). Finally, the method will also check if the x coordinate match any of the blackHoleX (any among 5) — this is to prevent the player from accessing the hole the player had already accessed before.

If the player had been at that empty hole before, World's setLivesLeft() will be called to decrease the number of lives left by 1 and Sprite's resetToOffsetPosition() will be called too to reset the whole game.

Otherwise, the player will access the black hole. Once the player enters the black hole, then the World's enteredBlackHole() will be set to true and black hole's x coordinates will be sent in the World. Simultaneously Sprite's resetToOffsetPosition() will be called to sent the player back to the start position. This way, I do not allow the player to touch the upper level of trees.

- update() and Turtles — if isPlayerIsOnTurtles is true and Turtle's getAreTurtlesVisible() is true, then player's behavior will resemble that of player's position on logs. Otherwise ifPlayerIsOnTurtles() is true, but the turtles are not visible then the player dies (Sprite's resetToOffsetPosition is called and setLivesLeft() will be called to decrease the number of lives left).

Comments:
- timePassed holds time until the turtles appear or disappear.

- update() will either render the turtles or not, depending on whether areTurtlesVisible is true.

- turtleElapsedTime() updates the turtleTimeUnderWater and sevenSeconds. If sevenSeconds is more or equal to 7, then it flips the value of the areTurtlesVisible to false. If timePassed is equal to 2 the turtles resurface again by turning areTurtlesVisible to true.

- sevenSeconds counts whether seven seconds have passed since the turtles emerged. turtleTimeUnderWater measures how long has it been since the turtles had been under water.

- controlTime() ensures that once sevenSeconds start, turtleTimeUnderWater does not start as well. It is until sevenSeconds reach 7, when turtleTimeUnderWater starts counting. It tracks when each of them turns on and ensures they both don't run at the same time. It also changes sets the values when the time comes.

World
- sprites[*] : ArrayList<Sprite>
- readCSV[*] = String
- livesLeft[*] : ArrayList<Int> = [48, 96, 144]
- emptyHolesLeft : int = 5
- timeElapsed : float
- currentRandomValue : int
- livesLeft : int 3
- blackHoleX1 : int = -48
- blackHoleX2 : int = -48
- blackHoleX3 : int = -48
- blackHoleX4 : int = -48
- blackHoleX5 : int = -48
- blackHoleY : int = 96
- livesLeftY : int = 768
- extraLifeX : int
- extraLifeY : int
+ TILE_SIZE : int = 48
+ FIRST_REMAING_LIFE_X : int = 24
+ REMAING_LIVES_Y : int = 744
+ DISTANCE_DIFFERENCE : int = 32

+ readFirstCVS() : void
+ readSecondCVS() : void
+ timeUntilExtraLife() : float
+ generateRandomValue() : int
+ pickRandomLog() : void
+ World()
+ update(input: Input, delta: int): void
+ render(g: Graphics) : void
+ setLivesLeft() : void
+ getLivesLeft() : int
+ enteredBlackHole() : boolean
+ setBlackHoleX() : int
+ deleteExtraLife() : void
+ readyToCreateExtraLife() : boolean

BoundingBox
- left : float
- top : float
- width : float
- height : float
- FUZZ : float = 0.95f

+ BoundingBox(x: float, y: float, width: float, height: float)
+ BoundingBox(img: Image, x: float, y: float)
+ BoundingBox(bb: BoundingBox)
+ setX(x: float) : void
+ setY(y: float) : void
+ setWidth(w: float): void
+ setHeight(h: float) : void
+ getLeft(): float
+ getTop() : float
+ getRight() : float
+ getBottom() : float
+ getWidth() : float
+ getHeight() : float
+ intersects(other: BoundingBox) : boolean

Sprites
- bounds : BoundingBox
- image : Image
- x : float
- y : float
- playerXOffset : int = 512
- playerYOffset : int = 720
- tags[*] : String
- HAZARD : String = "hazard"

+ Sprite(imageSrc: String, x: float, y: float)
+ Sprite(imageSrc: String, x: float, y: float, tags: String[])
+ setupSprite(imageSrc: String, x: float, y: float)
+ setX(x: float) : void
+ setY(y: float) : void
+ getX() : float
+ getY() : float
+ move(dx: float, dy: float) : void
+ onScreen(x: float, y: float) : boolean
+ onScreen() : boolean (final)
+ collides(other: Sprite) : boolean
+ update(input: Input, delta: int) : void
+ onCollision(other: Sprite) : void
+ render() : void
+ hasTag(tag: String) : boolean
+ resetToOffsetPosition() : void

App
- world : World
+ SCREEN_WIDTH : int = 1024
+ SCREEN_HEIGHT : int = 768

+ App()
+ init(gc: GameContainer) : void
+ update(gc: GameContainer, delta: int) : vc
+ render(gc: GameContainer, g: Graphics) :
+ main (args: String[]) : void (static)

RemainingLives
- REMAING_LIFE_PATH : String = "assets/lives.png"

+ RemainingLives (x: float, y: float) : void

Comments:
- Here the class just creates the remaining lives.

Player
- PLAYER_PATH : String = "assets/frog.png"(static)
- treesX[12] : ArrayList<Int> = [so 0, 48, 192, 240, 384, 432, 576, 624, 768, 816, 960, or 1008]

+ Player(x: float, y: float)
+ isPlayerOnLog() : boolean
+ isPlayerWithBulldozer() : boolean
+ isPlayerOnTurtles : boolean
+ update(input: Input, delata: int) : void
+ onCollision(other: Sprite) : void
+ onCollisionSafe() : boolean
+ IsOnScreen() : boolean
+ moveIfOkay() : void

Tile
- GRASS_PATH : String = "assets/grass.png"
- WATER_PATH : String = "assets/water.png"
- TREE_PATH : String = "assets/tree.png"

+ createGrassTile(x: float, y: float) : new Tile(GRASS_PATH: String, x: float, y: float)
+ createWaterTile(x: float, y: float) : new Tile(WATER_PATH: String, x: float, y: float, new String[] { Sprite.HAZARD })
+ createTreeTile(x: float, y: float) : new Tile(TREE_PATH: String, x:float, y:float)
- Tile(imageSrc: String, x: float, y: float)
- Tile(imageSrc: String, x: float, y: float, tags: String[])

MovingObjects
- moveRight : boolean

- getInitialX() : float
+ Vehicle(x: float, y: float, moveRight: boolean)
+ update(input: Input, delta: int) : void
+ setMoveRight() : void
+ getMoveRight() : boolean

Turtles
- TURTLES_PATH : String = "assets/turtles.png"
- TURTLES_SPEED: float = 0.085f
- turtleTimeUnderWater : float
- turtleMoveRight : boolean
+ turtleCurrentX : int
+ turtleCurrentY : int
+ areTurtlesVisible : boolean
+ sevenSeconds : int

- getInitialX() : float (final)
+ Turtles(x: float, y: float, moveRight: boolean)
+ update(input: Input, delta: int) : void
+ turtleElapsedTime() : float
+ getAreTurtlesVisible() : boolean
+ controlTime() : void

Comments:
- Trees will be created in the same way as the rest of the tiles — that is why there are not that many changes to the sample code.

Bus
- BUS_PATH : String = "assets/bus.png"
- BUS_SPEED : float = 0.15f

- getInitialX() : float
+ Bus(x: float, y: float, moveRight: boolean)
+ update(input: Input, delta: int) : void

Racecar
- RACECAR_PATH : String = "assets/racecar.p
- RACECAR_SPEED : float = 0.5f

- getInitialX() : float
+ Racecar(x: float, y: float, moveRight: boolean
+ update(input: Input, delta: int) : void

Longlog
- LONGLOG_PATH : String = "assets/longlog.png"
- LONGLOG_SPEED : float = 0.07f
- longLCurrentX : int
- longLCurrentY : int

- getInitialX() : float
+ Longlog(x: float, y: float, moveRight: boolean)
+ update(input: Input, delta: int) : void
+ getLongLCurrentX() : int
+ getLongLCurrentY() : int

Log
- LOG_PATH : String = "assets/log.png"
- LOG_SPEED : float = 0.1f
- logMoveRight : boolean
+ logCurrentX : int
+ logCurrentY : int

- getInitialX() : float (final)
+ Log(x: float, y: float, moveRight: boolean)
+ update(input: Input, delta: int) : void

Comments:
- update() moves the objects, but also updates theirCurrentCoordinates of each log.

- getters and setter allow current coordinates of the logs (e.g. longLCurrentX) to be accessed by the player when the player jumps onto a log/turtle.

Bike
- BIKE_PATH : String = "assets/bike.png"
- BIKE_SPEED : float = 0.2f

- getInitialX() : float
+ Bike(x: float, y: float, moveRight: boolean
+ update(input: Input, delta: int) : void
+ changeDirection() : void

Comments:
- To show a smooth transition of objects when they disappear and appear off screen, I added 48 to the boundaries on both sides of the screen so that they seemingly appear and disappear smoother.

Comments:
- changeDirection() checks calls getX() to check if the bike has reached either 24 or 1000. If it did, then the bike changes direction by calling setMoverRight()
- setMoveRight() first calls getMoveRight() and flips the value of the moveRight, depending on what value getMoveRight() returns.

Bulldozer
- BULLDOZER_PATH : String = "assets/bulldozer.pn
- BULLDOZER_SPEED : float = 0.05f
- bulldozerXDifference : int
- bulldozerYDifference : int
- bulldozerX : int
- bulldozerY : int

- getInitialX() : float
+ Bulldozer(x: float, y: float, moveRight: boolean)
+ update(input: Input, delta: int) : void
+ getBulldozerXDifference() : int
+ getBulldozerYDifference() : int

Comments:
- getBulldozerXDifference() and getBulldozerYDifference() are meant to change the player's position — it will look as if the player wa pushed by the bulldozer because player's change in position will be the same as that of bulldozer's.

Extralife
- EXTRALIFE_PATH : String = "assets/extralife.ong"
- timeLeftToMove : int = 2
- isExtraLifeVisible : boolean
- extraLifeTimePassed : float

+ getInitialX() : float
+ Extralife(x: float, y: float, moveRight: boolean)
+ isTimeToMove() : boolean
+ extraLifeTimeElapsed() : void
+ update(input: Input, delta: int) : void
+ onCollision() : void
+ getExtraLifeVisible() : boolean
+ setExtraLifeVisible() : void
+ setExtraLifeTimePassed() : void
+ isOnEdge() : boolean
+ timeElapsed() : void

Comments:
- isTimeToMove determines if 2 seconds have passed yet.

- timeElapsed updates TimeLeftToMove. Once TimeLeftToMove hits zero (or less, depending on delta), then moveRight will flip values (flipping will be done with an if statement)

Comments:
- Similar reasoning as turtles — similar variables track how time has passed.

- onCollision() will override the parent class definition. If the onCollision detects collision then the livesLeft in the World class will be upped by 1.

- extraLifeTimeElapsed checks whether 14 seconds have passed (using delta to determine how much time have passed). Once extraLifeTimePassed is 14, isExtraLifeVisible turns false. Moreover, it calls World's deleteExtraLife() to eliminate the extra life from the Sprites (so that it does not render anymore). Moreover, the method will also call World's generateRandomValue() to start countdown until next extra life.

- getter for extraLifeVisible is for World class to set Extra Life to false at the start of Level 2 and the World will also set extraLifeTimePassed to 0 (in the case where Extra Life still exists from Level 1)

- isOnEdge() determines if the extra life has reached the end of the log. If extra life has reached the edge (the method will return true), the setMoveRight() to the opposite value.

- timeElapsed() updates timeLeftToMove and extraLifeTimeElapsed by adding delta to their values.