## FroggerModel FroggerController - CAR START Y: int - INDEX INCREMENT : int - MAX NUM SCORES: int - SCORE INCREMENT : int

- MAX LEVELS: int - OFFSCREEN LEFT : int - STEP SIZE LR: double - OFFSCREEN\_RIGHT: int - STEP\_SIZE\_UD : double - PANE\_WIDTH: int

- RIVER\_START\_Y: int

+ generateCarPaths(): Road[]

+ generateLilyPads() : LilyPad[]

+ setGameMode( mode : int )

- STARTING X POS: int

- fileName : String

- height: String

- width : double

+ restartFrog()

- HEIGHT: double

- gameMode : int

- thePath: Path

+ getThePath(): Path

+ moveFroggerObject()

+ setWidth( width : int )

- mediaPlayer : MediaPlayer

- theModel: FroggerModel

- theView : FroggerView

+ FroggerMain()

+ init ()

- theController : FroggerController

- theMainMenu : FroggerMainMenu

int)

STARTING Y POS: int

- isOnWaterObject : boolean

+ setIsOnWaterObjectFalse()

+ setIsOnWaterObjectTrue()

- pathTransition : PathTransition

+ generateWaterObjectPaths(): River[]

Frog

+ setXTranslation( Xtranslate : double )

MovingObject

+ MovingObject( fileName : String, startX :

int, startY: int, endX: int, gameMode: int)

- createPathTransition( startX : int, endX :

FroggerMain

- createPath( startX : int, endX : int)

- currentLevel : int

- gameMode: int

+ FroggerModel()

+ levelUp(): boolean

+ resetLevels()

- carCollisionsTask : CarCollisionsTask - controlledByKeys: boolean

- frogIndex : int - gameOver : boolean - highScores : HighScores

- lilyPadCollisionsTask : LilyPadCollisionsTask

- maxFrogIndex : int - maxScore : int - minFrogIndex : int - numLives: int

- ridingWaterObjectTask : RideTheWaterObjectTask - score : int

- theMainMenu: FroggerMainMenu - theModel : FroggerModel - theMoveCarTask : MoveCarsTask

- sleepTimeFactor : int

- theMoveWaterObjectsTask : MoveWaterObjectsTask

- theView : FroggerView

- waterObjectCollisionsTask : WaterObjectCollisionTask

+ Frog() + getHeight(): double + FroggerController( theView : FroggerView, theModel : + getWidth(): double FroggerModel, theMainMenu: FroggerMainMenu) + getIsOnWaterObject(): boolean

+ checkBottomBound(): boolean

+ checkCarCollisions()

+ checkLilyCollisions() + checkRightBound(): boolean

+ checkTopBound(): boolean

+ checkWaterObjectCollision()

+ getControlledByKeys(): boolean

+ getNumLives(): int + isGameOver(): boolean

+ restartFrogIndex()

+ setGameOver( gameover : boolean )

+ startTheWaterObjects ( delay : int )

+ updateFrogDownPosition()

+ updateFrogLeftPosition()

CarCollisionsTask :: Task <Integer>

- cars : MovingObject[] stopTask : boolean

+ call(): Integer

RideTheWaterObjectTask :: Task <Integer>

- theObject : MovingObject - isPaused : boolean

+ RideTheWaterObjectTask( theObj : MovingObject )

+ call(): Integer + stopTask()

+ resumeTask()

+ pauseTask()

HighScores

- adjustScore( i : int )

+ checkLeftBound(): boolean

+ endGame()

+ getMaxScore(): int

+ removeLife()

+ setControlledByKeysFalse()

+ startTheCars( delay : int )

+ stopRidingWaterObjectTask()

+ updateFrogRightPosition()

+ updateFrogUpPosition()

+ CarCollisionsTask( cars : MovingObject [])

+ stopTask()

- cars : MovingObject[] - baseDelay: int

+ MoveCarsTask( cars : MovingObject [], delay :

MoveCarsTask :: Task <Integer>

FroggerView

- NUM\_LIVES: int - carGroup : Group

- root : Pane

- theFrog: Frog

- theRivers : River[]

- theRoads : Road[]

+ addFrog()

- addLives()

+ addPaths()

+ addScore()

- clearFrogs()

FroggerController)

+ getNum\_LIVES(): int

+ getRootNode(): Pane + getRootXMax(): int

+ getRootXMin(): int

+ getRootYMax(): int

+ getRootYMin(): int

+ getTheFrog(): Frog

+ launchNewFrog()

+ removeNextLife()

FroggerController)

- lilyPads : LilyPad []

- onPad : boolean

+ call(): Integer

- waterPath: River

+ call(): Integer

+ stopTask()

- stopTask : boolean

+ levelUp()

+ getTheLilyPads() : LilyPad[]

+restartGame( froggerController :

LilyPadCollisionsTask :: Task <Integer>

+ LilyPadCollisionsTask( lilypads : LilyPad [])

WaterObjectsCollisionsTask :: Task <Integer>

+ WaterObjectsCollisionsTask( path : River )

+ updateScores( score: int )

+ getTheRivers(): River[]

+ getTheRoads(): Road[]

- addLilyPads()

- waterGroup : Group

- lilyPadGroup: Group

- theLilyPads : LilyPad[]

- theModel : FroggerModel

- safeFrogs : ArrayList<Frog>

- score : SimpleIntegerProperty

- theLives : ArrayList<ImageView>

+ FroggerView(theModel: FroggerMovel)

+ endGame(finalScoer: int, scores:

ArrayList<Integer>, froggerController:

+ call(): Integer

MoveWaterObjectsTask :: Task <Integer>

- waterObjects : MovingObject[]

- baseDelay: int

+ MoveWaterObjectsTask( waterObjects: MovingObject [], delay: int)

+ call(): Integer

- root : StackPane

+ main ( args : String [])

+ start ( primaryStage : Stage )

+ handle ( event : KeyEvent )

FroggerMainMenu - LOADTIME : Integer

- beginner : RadioButton - exit : Button

- expert : RadioButton - levelGroup : ToggleGroup

- start : Button

 levelGroup : ToggleGroup - root : StackPane - start : Button

- startGame : boolean - subStage : Stage

- theModel : FroggerModel

- theVBox : VBox

- timeSeconds : IntegerProperty

- timeline : Timeline - title : Label

+ FroggerMainMenu( theModel :

FroggerModel)

- addButtons()

- addLevelOptions()

- addProgressBar()

- addTitle()

+ getGameDelay

+ getStartGame()

River

- NUM OBJECTS IN RIVER : int

- WIDTH : int - gameMode : int - logs : boolean

- theWaterObjects : MovingObejct []

+ River( startX : int, startY : int, endX : int, logs: boolean, gameMode: int) - addObjects( startX : int, startY : int, endX: int)

+ getTheObjects() : MovingObject []

- getWaterObjectType() : String

**HighScores** 

- scores : ArrayList<Integer>

+ HighScores()

+ pause lask()

+ getScores() : ArrayList<Integer>

+ insertScore( score : int ) : ArrayList<Integer>

+ printScores()

+ saveScores()

LilyPad

- HEIGHT : double

- WIDTH : double

- fileName : String - isOcuppied : boolean

- xLocation : double

- yLocation : double

+ LilyPad( xLocation : double )

+ getIsOccupied : boolean

+ getxLocation() : double

+ getyLocation(): double

+ setOccupiedFalse()

+ setOccupiedTrue()

setPosition()

MovingObject [], delay: int) + call(): Integer

Road

- NUM CARS ON ROAD : int

- WIDTH: int

- carFilesL : String [] - carFilesR : String [] - gameMode : int - faceRight : boolean - theCars : MovingObejct []

+ Road( startX : int, startY : int, endX : int, faceR :

boolean, gameMode: int)

- addCars( startX : int, startY : int, endX : int)

+ getTheCars(): MovingObject []

- getCarType() : String