

**รายงาน**

**เรื่อง** **Running Dino!**

**สมาชิกกลุ่ม**

**นายกิตติวัชร เอี่ยมกิจการ รหัสนักศึกษา 61070017**

**นางสาวชญานี คำเจริญ รหัสนักศึกษา 61070030**

**นางสาวชรินรัตน์ บูรณะพิสิฐ รหัสนักศึกษา 61070037**

**นายภัทรนันท์ เรืองชนา รหัสนักศึกษา 61070156**

**นางสาวภาวินี ทองบ่อ รหัสนักศึกษา 61070163**

**นักศึกษาชั้นปีที่ 2**

**เสนอ**

**ดร. สุพัณณดา โชติพันธ์**

**ดร. ธราวิเชษฐ์ ธิติจรูญโรจน์**

**รายงานนี้เป็นส่วนหนึ่งของวิชา Object Oriented Programming รหัสวิชา 06016317**

**ภาคเรียนที่ 1 ปีการศึกษา 2562**

**คณะเทคโนโลยีสารสนเทศ สาขาวิชาเทคโนโลยีสารสนเทศ (ภาคปกติ)**

**สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง**

**คำนำ**

รายงานเล่มนี้เป็นส่วนหนึ่งของวิชา object oriented programming รหัสวิชา 06016317 คณะเทคโนโลยีสารสนเทศ สาขาวิชาเทคโนโลยีสารสนเทศ (ภาคปกติ) โดยมีจุดประสงค์เพื่อบอกรายละเอียดของเกม Running Dino! ที่พัฒนามาจากภาษา Java ในรูปแบบของ object oriented programming โดยในรายงานนั้นประกอบด้วยบทคัดย่อ, วัตถุประสงค์และประโยชน์ที่ได้รับ, ไดอะแกรมทั้งหมดของเกม และรายละเอียดเกี่ยวกับการทำงานใน Class ต่าง ๆ

คณะผู้จัดทำหวังว่ารายงานเล่มนี้จะเป็นประโยชน์กับผู้อ่าน หากมีข้อแนะนำหรือข้อผิดพลาดประการใด คณะผู้จัดทำขอน้อมรับไว้และขออภัยมา ณ ที่นี้ด้วย

คณะผู้จัดทำ

22 พฤศจิกายน 2562

**สารบัญ**

**เรื่อง หน้า**

คำนำ ก

สารบัญ ข

บทคัดย่อ 1

วัตถุประสงค์ 1

ประโยชน์ที่ได้รับ 2

คลาสไดอาแกรม 3

**คำอธิบาย Attribute และ Method**

หน้าที่ของสมาชิกภายในกลุ่ม

บรรณานุกรม ค

**บทคัดย่อ**

ในปัจจุบันนั้นมีเกมเพิ่มขึ้นมากมายหลากหลายประเภท ซึ่งผู้เล่นสามารถเลือกเล่นได้ตามความต้องการ ทั้งเกมในรูปแบบออนไลน์ที่สามารถเล่นร่วมกับผู้อื่น และเกมในรูปแบบออฟไลน์ที่ไม่ต้องใช้อินเทอร์เน็ต นอกจากนี้ เกมนั้นเริ่มมีบทบาทในชีวิตประจำวันมากขึ้น ยกตัวอย่างเช่น เกมไดโนเสาร์กระโดดผ่านสิ่งกีดขวาง ที่มักปรากฏขึ้นบนเบราว์เซอร์เมื่อไม่มีสัญญาณอินเทอร์เน็ต แต่เมื่อสัญญาณอินเทอร์เน็ตกลับมาใช้งานได้ปกติแล้ว ผู้ใช้งานก็ยังคงเล่นเกมนี้อยู่ เพราะเกมไดโนเสาร์กระโดดผ่านสิ่งกีดขวางนั้นเป็นเกมที่ให้ความเพลิดเพลินและฝึกสมาธิได้เป็นอย่างดี

ดังนั้นกลุ่มของพวกเราจึงมีแนวคิดที่อยากจะพัฒนาเกมไดโนเสาร์กระโดดนี้ให้มีความน่าสนใจมากยิ่งขึ้น โดยมีชื่อเกมว่า “Running Dino!” ซึ่งตัวเกมนั้นพัฒนาด้วยภาษา Java ในรูปแบบของ Object Oriented Programming และตัวเกมอยู่ในรูปแบบของ Endless Game หรือเกมที่ไม่มีจุดสิ้นสุดในการเล่น นอกจากนี้ยังมีดีไซน์ที่น่ารักสดใส มีฟังก์ชันที่ใช้งานง่าย สามารถเล่นได้ทุกเพศทุกวัย ให้ความเพลิดเพลินในการเล่น และเป็นเกมในแบบออฟไลน์ โดยในตัวเกมจะให้เราสวมบทบาทเป็นไดโนเสาร์ที่กำลังวิ่งหนีอุกกาบาตจากนอกโลกและหลบสิ่งกีดขวางข้างหน้าไปด้วย มีฉากที่เปลี่ยนไปตามคะแนนที่กำหนด มี HP ที่เมื่อโดนสิ่งกีดขวางก็จะลดลงไปเรื่อย ๆ และมี Score ไว้เก็บคะแนนของผู้เล่นที่เล่นได้มากที่สุด ดังนั้นผู้เล่นจึงสามารถเล่นร่วมกับผู้อื่นเพื่อแข่งกันว่าใครมีคะแนนมากกว่ากันได้ และเป็นเกมที่ฝึกสมาธิได้เป็นอย่างดี

**วัตถุประสงค์**

1. เพื่อสร้างความสนุกสนานและความเพลิดเพลินให้แก่ผู้เล่น
2. เพื่อใช้เวลาว่างให้เกิดประโยชน์สูงสุด
3. เพื่อฝึกหลักการเขียน Object Oriented Programming
4. เพื่อฝึกพื้นฐานการเขียนภาษา Java

**ประโยชน์ที่ได้รับ**

1. ได้รับความสนุกสนานและความเพลิดเพลิน
2. ได้ฝึกสมาธิมากยิ่งขึ้น ทำให้มีความจดจ่อกับสิ่งที่ทำอยู่ได้มากยิ่งขึ้น
3. ได้เรียนรู้วิธีการสร้างและพัฒนาเกมโดยใช้ Object Oriented Programming ด้วยภาษา Java
4. ได้ฝึกการทำงานร่วมกับผู้อื่น
5. ได้ใช้เวลาว่างให้เกิดประโยชน์สูงสุด

**คลาสไดอาแกรม**

(รูปอยู่ในไฟล์ชื่อ ไดอาแกรมค้าบ แนะนำว่าทำแต่ละคลาสให้เสร็จก่อนแล้วครอปเป็นรูปไป เพราะถ้าเอาตารางมาใส่เลยมันจะทับกันแล้วจะมีปัญหามาก ๆ)

**คำอธิบาย Attribute และ Method**

**Class : StartWindow**

**Attribute :**

* SCREEN\_WIDTH : ตัวแปรเก็บขนาดความกว้างหน้าจอของหน้าใส่ชื่อ
* name : ตัวแปรเก็บชื่อที่กรอกเข้ามา
* gameWindow : Object JFrame ของเกม
* sp : JPanel สำหรับรองรับปุ่มบนหน้าจอ
* lb : JLabel ใส่ข้อความให้ user รู้ว่าต้องใส่ชื่อ
* n : JTextField ช่องใส่ชื่อ
* start : JButton ปุ่มเข้าหน้าเริ่มเกม
* screenSize : ตัวแปรสำหรับเรียกดูขนาดของหน้าจอ (ความสูง ความกว้าง หน้าจอcomputer)

**Method :**

* **StartWindow() :**

สร้างหน้าจอJFrame ของหน้าใส่ชื่อ หลังจากนั้น add ActionListener ให้ปุ่ม start และ add KeyListener ให้ JTextField เพื่อรับการกด spacebar , enter ให้ปิดหน้าจอนี้ และ เปิดหน้าจอเริ่มเกม

* **startPage() :**

เปิดการมองเห็นของหน้าจอนี้

* **main(String ares[]) :**

สร้าง Object ของclass นี้และสั่งใช้ startPage()

* **actionPerformed(ActionEvent e) :**

รับการทำงานจากปุ่ม start เพื่อปิดหน้านี้ และ เปิดหน้าเริ่มเกม(GameWindow) พร้อมสั่งmethod startGame ของ GameWindowให้ทำงาน

**Class : GameWindow**

**Attribute :**

* SCREEN\_WIDTH : ตัวแปรเก็บขนาดความกว้างหน้าจอของหน้าใส่ชื่อ
* gameScreen : Object JPanel ของเกม
* screenSize : ตัวแปรสำหรับเรียกดูขนาดของหน้าจอ (ความสูง ความกว้าง หน้าจอcomputer)

**Method :**

* **GameWindow() :**

สร้างหน้าจอ JFrame ของเกม add keyListener ให้ gameScreen และ add gameScreen ลง JFrame

* **startGame(String n) :** เปิดการมองเห็นหน้าจอนี้ หลังจากนั้นนำค่า n ที่นำเข้ามาใส่ลง method getNameFirst ของ GameScreen จากนั้นสั่งให้ method startGame ของ GameScreen ทำงาน

**Class : HighScoreStorage**

* **Attribute :**

- name : เก็บข้อมูลชื่อที่อ่านจากที่ได้บันทึกไว้

- highscore : เก็บข้อมูลคะแนนสูงสุดที่อ่านจากที่ได้บันทึกไว้

* **Method :**

- **getHighscore()** : อ่านข้อมูลคะแนนสูงสุด (highscore) ที่ได้บันทึกไว้ และตรวจสอบว่ามีไฟล์ดังกล่าวอยู่หรือไม่ หากมีไฟล์ Highscore.dat ให้ทำการเปิดไฟล์ เพื่ออ่านค่าคะแนนสูงสุด (highscore) ออกมาแสดง หากไม่มีไฟล์ Highscore.dat ให้ค่า highscore มีค่าเป็น 0 จากนั้น return ส่งค่า highscore กลับไป

- **getName()**  : อ่านข้อมูลชื่อที่ได้คะแนนสูงสุด (name) ที่ได้บันทึกไว้ และตรวจสอบว่ามีไฟล์ดังกล่าวอยู่หรือไม่ หากมีไฟล์ Name.dat ให้ทำการเปิดไฟล์ เพื่ออ่านค่าString (name) ออกมาแสดง หากไม่มีไฟล์ Name.dat ให้ค่า Name มีค่าเป็น “(none)” จากนั้น return ส่งค่า name กลับไป

- **saveHighscore(int hs)** : เก็บค่าคะแนนสูงสุดลงในไฟล์ Highscore.dat

- **saveName(String n)** : เก็บชื่อที่ได้คะแนนสูงสุดลงในไฟล์ Name.dat

**Class : GameScreen**

* **Attribute :**

enemyAndLandCount :

blackBox :

countStage :

name : ชื่อที่กรอกเข้ามาในตอนเข้าเกม

nameHs : เก็บชื่อที่ได้คะแนนสูงสุด

hightScore : เก็บค่าคะแนนสูงสุด

jumpCount : นับจำนวนการกระโดดของ mainCharacter ไม่ให้มีค่ามากกว่า 2

bg :

backgroundPoint :

START\_GAME\_STATE :

GANE\_PLAYING\_STATE :

GAME\_OVER\_STATE :

Manager :

mainCharacter : Object ของ MainCharacter ที่ใช้ภายในคลาสนี้

clouds :

thread :

score :

runScore :

isKeyPressed : ตรวจสอบว่ามีการกดปุ่มหรือไม่

enemyAndLandCount :

gameState :

replayButtonimage :

gameOverButtonImage :

speedGameM :

speedGameN :

* **Method :**

GameScreen () :

getNameFirst ( String n ) : กำหนดให้ name มีค่าเท่ากับ n

startGame () : สร้าง Object thread และ runScore ให้เป็นแบบ Thread จากนั้นเรื่มการทำงานของ Thread

gameUpdate () : ทำงานทุกครั้งเมื่อ run เพื่อให้ตัวเกมมีการอัพเดตสถานะ อย่างเช่น score, gameState, hp เป็นต้น

paint ( Graphics g ) : สร้าง graphic และ drawstring บน JPanel ตามค่าของ gameState

run () :

getCountState () : คืนค่า countStage

keyPressed ( KeyEvent e ) : รับการทำงานจากแป้นพิมพ์ ตรวจสอบเงื่อนไขหาก isKeyPressed มีค่าเป็น true จะสามรถทำงานได้ และค่าของตัวแปร gameState แต่ละค่าจะมีการทำงานที่แตกต่างกัน

START\_GAME\_STATE :

e.getKeyCode() มีค่าเท่ากับ KeyEvent.VK\_SPACE หมายถึงสั่งให้ method start ของ runScore ทำงาน และเปลี่ยนค่า gameStete เป็น GAME\_PLAYING\_STATE

GAME\_PLAYING\_STATE :

e.getKeyCode() มีค่าเท่ากับ KeyEvent.VK\_SPACE หมายถึงสั่งให้ method jump ของ mainCharacter ทำงาน หรือให้ mainCharacter กระโดดนั่นเอง

e.getKeyCode() มีค่าเท่ากับ KeyEvent. VK\_DOWN หมายถึงสั่งให้ method down ของ mainCharacter ทำงาน และใส่ค่า true เข้าไป

ใน GAME\_OVER\_STATE :

e.getKeyCode() มีค่าเท่ากับ KeyEvent.VK\_SPACE หมายถึงเริ่มเล่นเกมใหม่อีกครั้ง โดยมีทำการ reset แต่ละค่าที่กำหนดไว้

keyRelesrd ( KeyEvent e ) : กำหนดค่า isKeyPressed เป็น false และเช็คค่า gameState เมื่อเป็น GAME\_PLAYING\_STATE หากปุ่มที่กดก่อนหน้าคือปุ่ม Down จะสั่งให้ method down ของ mainCharacter ทำงาน และใส่ค่า false เข้าไป

resetGame () : สั่งให้ method reset ของ maneger, method dead ของ mainCharacter โดยใส่ค่า false เข้าไป, และ method reset ของ mainCharacter ให้ทำงาน

setSpeed ( int d ) :

**Class : RunnerScore**

* **Attribute :**

upSpeedGame : เก็บจำนวนที่ไว้ใช้ลดdelayของตัวเกม

* **Method :**

run() : วนการลดdelayของเกมทุก 1 วินาที

**Class : DinoCharacter**

**Attribute :**

LAND\_POSY : เก็บตำแหน่งความสูงของพื้น

GRAVITY : ความเร็วในการกลับลงพื้น

NORMAL\_RUN : บอกสถานะว่าวิ่งแบบปกติอยู่

JUMPING : บอกสถานะว่าให้กระโดด

DOWN\_RUN : บอกสถานะว่ากำลังหมอบ

DEATH : บอกสถานะว่า hp หมดแล้ว

posY : ตำแหน่งแกน Y บนหน้าจอเกม

posX : ตำแหน่งแกน x บนหน้าจอเกม

speedX : ความเร็วในการเคลื่อนที่ ( ที่ให้วัตถุอื่นขยับเข้ามาหา DinoCharacter )

speedY : ความเร็วเร็วในการตกถึงพื้น

rectBound : พื้นที่ และ ตำแหน่งในการชนกับวัตถุอื่นของ DinoCharacter

countJump : เก็บจำนวนการกระโดดที่ต่อเนื่องกัน และ กำหนดค่า = 0เมื่อถึงพื้น

hp : เก็บเลือดของ DinoCharacter

state : เก็บสถานะต่างๆของ DinoCharacter

normalRunAnim : เก็บ frame รูปภาพที่ใช้ทำanimate ของการวิ่งแบบปกติ

jumping : เก็บรูปภาพตอนกระโดดของ DinoCharacter

downRunAnim : : เก็บ frame รูปภาพที่ใช้ทำanimate ของการวิ่งแบบหมอบ

deathImage : เก็บรูปภาพตอนเลือดหมดของ DinoCharacter

jumpSound : เก็บAudioเสียงตอนกระโดด

deadSound : เก็บAudioเสียงตอนhpหมด

scoreUpSound : เก็บAudioเสียงทุกที่จะดังทุก ๆ ช่วง 100 คะแนน

normalState : เก็บรูปภาพตอนวิ่งปกติ

downState : เก็บรูปภาพตอนวิ่งแบบหมอบ

**Method :**

DinoCharacter () : กำหนดเสียงที่จะใช้ให้ตัวแปรเก็บเสียง กำหนดรูปภาพให้ตัวแปร abimation และ กำหนดแกน x , y ให้ DinoCharacter

getSpeedX () : คือค่าความเร็วในการวิ่ง

setSpeedX ( double speedX ) : กำหนดความเร็วของ DinoCharacter

draw ( Graphics g ) : วาดรูปภาพของ DinoCharacter ตาม Frame รูปภาพที่เปลี่ยนไป และ เปลี่ยนชุดรูปภาพตาม state ปัจจุบัน

update () : ขยับ frame ภาพของ normalRunAnim , downRunAnim และ ตรวจสอบว่า DinoCharacter อย่ตำแหน่งเดียวกันกับพื้นหรือไม่ ภ้าไม่อยู่ก็จะเพื่อค่าแกน y ตามค่า GRAVITY จนกว่า DinoCharacter จะไม่อยู่สูงกว่าพื้น

jump () : ลดค่าในแกน y เพื่อให้ DinoCharacter ลอยจากพื้น

down ( boolean isDown ) :

getBound () : คำนวณพื้นที่ และ ตำแหน่งที่ใช้ในการตรวจสอบการชนกันของวัตถุนี้กับวัตถุอื่น และ คืนค่าออกมาเป็นตัวแปร Rectangle

dead ( Boolean isDeath ) : ตรวจสอบว่าเกิดการชนกันกับวัตถุอื่นหรือไม่ ถ้าชนจะคืนค่า True ถ้าไม่จะคืนค่า False

reset () : กำหนดตำแหน่งแกน y บนหน้าจอของ DinoCharacter ให้เท่ากับ LAND\_POSY และ กำหนดเลือดให้กลับมาเต็ม

playDeadSound () : สั่งให้ deadSound เล่น

playScoreSound () : สั่งให้ scoreUpSound เล่น

setLAND\_POSY ( int g ) : กำหนดแกน y ของเป็นตำแหน่งพื้นที่ DinoCharacter ใช้ในการอ้างอิง

getLAND\_POSY () : คืนค่าแกน y ของเป็นตำแหน่งพื้นที่ DinoCharacter ใช้ในการอ้างอิง

getPosX () : คืนค่าตำแหน่งแกน x บนหน้าจอเกมของ DinoCharacter

getHp () : คือค่า hp ที่ยังเหลืออยู่ของ DinoCharacter

setHp ( int hp ) : กำหนดค่า hp ของ DinoCharacter

**Class : ObjectGameManager**

* **Attribute :**

countEnemy : เก็บจำนวนสิ่งกีดขวางที่ถูกสร้างมาติดๆกัน

LAND\_POSY : เก็บตำแหน่งความสูงของพื้น

listLand : arraylist เก็บ Object ของพื้นที่ยังอยู่บนหน้าจอ

land[] : เก็บรูปของพื้นแต่ละstage

pitCount : เก็บจำนวนหลุมที่ออกมาติดกัน ถ้าถูกขั้นด้วยพื้นจะเริ่มนับใหม่

mainCharacter : Object ของDinosaurตัวหลัก

pitList[][] : เก็บหลุมของแต่ละstage และ แต่ละ stage เก็บหลายแบบ

rand : ตัวเก็บตัวเลขในการสุ่มค่าต่างๆ

listEnemies : arraylist เก็บ Object ของ Class ที่สืบทอดมาจาก Enemy

boxWidth : ขนาดความกว้างของboxพื้น

blankBox : Object ของจุดที่ไม่มีสิ่งกีดขวาง

pre : รับObject landBox ในการเช็คว่าเป็นหลุมแบบเดี่ยวหรือคู่

* **Method :**

ObjectGameManager( int width, DinoCharacter mainCharacter) :

createEnemy( int type, int posX) :

draw( Graphics g) : จะคอยวาดกราฟฟิก( รูปภาพ )ที่ถูกกำหนดขึ้นมาในตัวแปรประเภทBufferedImage

update() : ขยับObjectของพื้น และ สิ่งกีดขวางเข้ามาตามspeedXของmainCharacter ปรับLAND\_POSY ของ mainCharacterตามลักษณะพื้น(ถ้าเป็นหลุม LAND\_POSY จะมีค่าสูง)

และ จัดการเอาObjectที่เกินขอบหน้าจอออก แล้ว เพิ่มเข้าไปในlistใหม่

randomNumber(int limit) : แลนด้อมตัวเลขเป็นintออกมาในขอบเขตของ limitที่ใส่เข้าไป

isCollison() : คำนวณว่ามีการชนกันของวัตถุเกิดขึ้นไหม แล้ว ส่งTrue/False กลับออกมา

newStage() :

reset() : clearค่าในlistLand , listEnemyออก สร้างObject สิ่งกีดขวางแบบช่องว่าง และ พื้นใส่เข้าไปในlistให้ครบ 12

setImageLand( landBox imgLand, int type) : กำหนดรูปภาพให้imgLand ตามtypeที่สุ่มมาได้ ถ้าเป็นหลุมแต่เกินจำนวนสิ่งกีดขวางที่ติดกันได้จะสุ่มtypeใหม่จนกว่าจะไม่ใช่หลุม

**Class : landBox**

**Attribute :**

posX : เก็บตำแหน่งแกน x ของlandBoxบนหน้าจอ

image : เก็บรูปภาพของพื้น

numberOfPit : เก็บค่าว่าเป็นหลุมหรือเปล่า และ เป็นหลุมลำดับที่เท่าไหร่นับจากที่ติดกัน

**Method :**

-

**Class : Enemy\***

**Attribute :**

**-**

**Method :**

Update( double posX) :

Draw (Graphics g) :

getBound() :

isOutOfScreen() :

**Class : blankBox**

**Attribute :**

posX : เก็บตำแหน่งแกน xของObjectนั้นที่อยู่บนหน้าจอ

**Method :**

blankBox( DinoCharacter mainCharacter, int posX) :

update( double posX) : updateตำแหน่งของObject blankBox บนหน้าจอในแกน x

**Class : groundEnemy**

**Attribute :**

Y\_LAND : ตำแหน่งพื้นในแกน y ของหน้าจอเกม

posX : ตำแหน่งของObjectในแนวแกน x บนหน้าจอเกม

width : ความกว้างของรูปภาพ

image : รูปของObject ที่จะแสดงบนหน้าจอเกม

mainCharacter : Object ของ DinoCharacter ที่ใช้คำนวนความเร็วในclassนี้

rectBound : ตำแหน่งที่ใช้ในการตรวจสอบการชนกันของวัตถุนี้กับวัตถุอื่น

listCactus[] : เก็บรูปสิ่งกีดขวางบนพื้นแต่ละแบบของ แต่ละstage

**Method :**

groundEnemy (DinoCharacter mainCharacter, int posX ) :

update ( double posX ) : ขยับแกน x ของวัตถุเข้าหา mainCharacter ตาม speedX ของ mainCharacter

draw ( Graphics g ) : คอยวาดรูปภาพของObjectลงบนหน้าจอทุกครั้งที่มีการ update

getBound () : คำนวณพื้นที่ และ ตำแหน่งที่ใช้ในการตรวจสอบการชนกันของวัตถุนี้กับวัตถุอื่น และ คืนค่าออกมาเป็นตัวแปร Rectangle

getPosX () : เรียกดูตำแหน่งแกน x ของObject

setPosX () : กำหนดตำแหน่งแกน x ของObject

getimageNum () : สุ่มตัวเลขในขอบเขตของจำนวนรูปภาพสิ่งกีดขวาง

**Class : AirEnemy**

**Attribute :**

Y\_LAND : ตำแหน่งพื้นในแกน y ของหน้าจอเกม

posX : ตำแหน่งของObjectในแนวแกน x บนหน้าจอเกม

width : ความกว้างของรูปภาพ

height : ความสูงของรูปภาพ

image : รูปของObject ที่จะแสดงบนหน้าจอเกม

mainCharacter : Object ของ DinoCharacter ที่ใช้คำนวนความเร็วในclassนี้

rectBound : ตำแหน่งที่ใช้ในการตรวจสอบการชนกันของวัตถุนี้กับวัตถุอื่น

listMons[][] : เก็บรูปสิ่งกีดขวางกลางอากาศแต่ละแบบของ แต่ละstage

flyAnim : เก็บรูปภาพของObject แต่ละFrame ไว้สำหรับการทำAnimation

isOutOfScreen () : ตรวจสอบว่าตำแหน่งของวัตถุออกจากหน้าจอจนสุดหรือยัง ถ้าสุดจะส่ง True ออกไป ถ้ายังไม่ออกจะส่ง falseออกไป

**Method :**

AirEnemy (DinoCharacter mainCharacter, int posX ) :

update ( double posX ) : ขยับแกน x ของวัตถุเข้าหา mainCharacter ตาม speedX ของ mainCharacter

draw ( Graphics g ) : คอยวาดรูปภาพของObjectลงบนหน้าจอทุกครั้งที่มีการ update

getBound () : คำนวณพื้นที่ และ ตำแหน่งที่ใช้ในการตรวจสอบการชนกันของวัตถุนี้กับวัตถุอื่น และ คืนค่าออกมาเป็นตัวแปร Rectangle

getPosX () : เรียกดูตำแหน่งแกน x ของObject

setPosX (int x) : กำหนดตำแหน่งแกน x ของObject

randomLandY () : สุ่มและกำหนดตำแหน่งแกน y ของวัตถุ

isOutOfScreen () : ตรวจสอบว่าตำแหน่งของวัตถุออกจากหน้าจอจนสุดหรือยัง ถ้าสุดจะส่ง True ออกไป ถ้ายังไม่ออกจะส่ง falseออกไป

**Class : Clouds**

**Attribute :**

listCloud : arraylist เก็บObject ของcloud ที่แสดงอยู่บนหน้าจอ

cloud[][] : เก็บรูปภาพก้อนเมฆแต่ละแบบของแต่ละ stage

mainCharacter : Object ของ DinoCharacter ที่ใช้คำนวนความเร็วในclassนี้

**Method :**

Clouds ( int width, DinoCharacter mainCharacter ) :

update () :

draw ( Graphics g ) :