

**โครงงาน**

**Project mini game(Ninja Adventure)**

**จัดทำโดย**

**6504062620221 อรรถพร ศึกสพ**

**อาจารย์ผู้สอน**

**ผู้ช่วยศาสตราจารย์ ดร.สถิต ประสมพันธ์**

**โครงงานนี้เป็นส่วนหนึ่งของวิชา Object Oriented Programming**

**ภาควิชาวิทยาการคอมพิวเตอร์และสาระสนเทศ คณะวิทยาศาสตร์ประยุกต์**

**มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ**

**ภาคเรียนที่ 1 / 2567**

**บทที่ 1**

**บทนำ**

**1.1 ที่มาและความสำคัญ**

โครงงานนี้จัดขึ้นเพื่อวัดระดับผลการเรียนในรายวิชา Object Oriented Programming(OOP) โดยการนำเนื้อหาในบทเรียนมาปรับใช้ สร้างผลงานในรูปแบบของเกม ซึ่งผู้จัดทำได้ออกแบบเกมในรูปแบบการเล่น solo player เพราะการเล่นคนเดียวทำให้เราสมาธิจดจ่อกับเกม จึงสามารถดึงทักษะเฉพาะตัวของผู้เล่นออกมาใช้ได้ดี

**1.2 ประเภทของโครงงาน**

เกมผจญภัยรูปแบบ 2 มิติ ผู้เล่น 1 คน

**1.3 ประโยชน์ที่ได้รับจากโครงงาน**

1.3.1 สามารถนำเนื้อหาที่เรียนมาใช้สร้างสรรค์ผลงานในรูปแบบของเกมได้

1.3.2 มีความรู้ด้านการเขียนโปรแกรมเชิงวัตถุ(OOP) ติดตัวไปใช้ในการทำงานในอนาคต

**1.4 ขอบเขตของโครงงาน**

1.4.1 แผนการดำเนินงาน

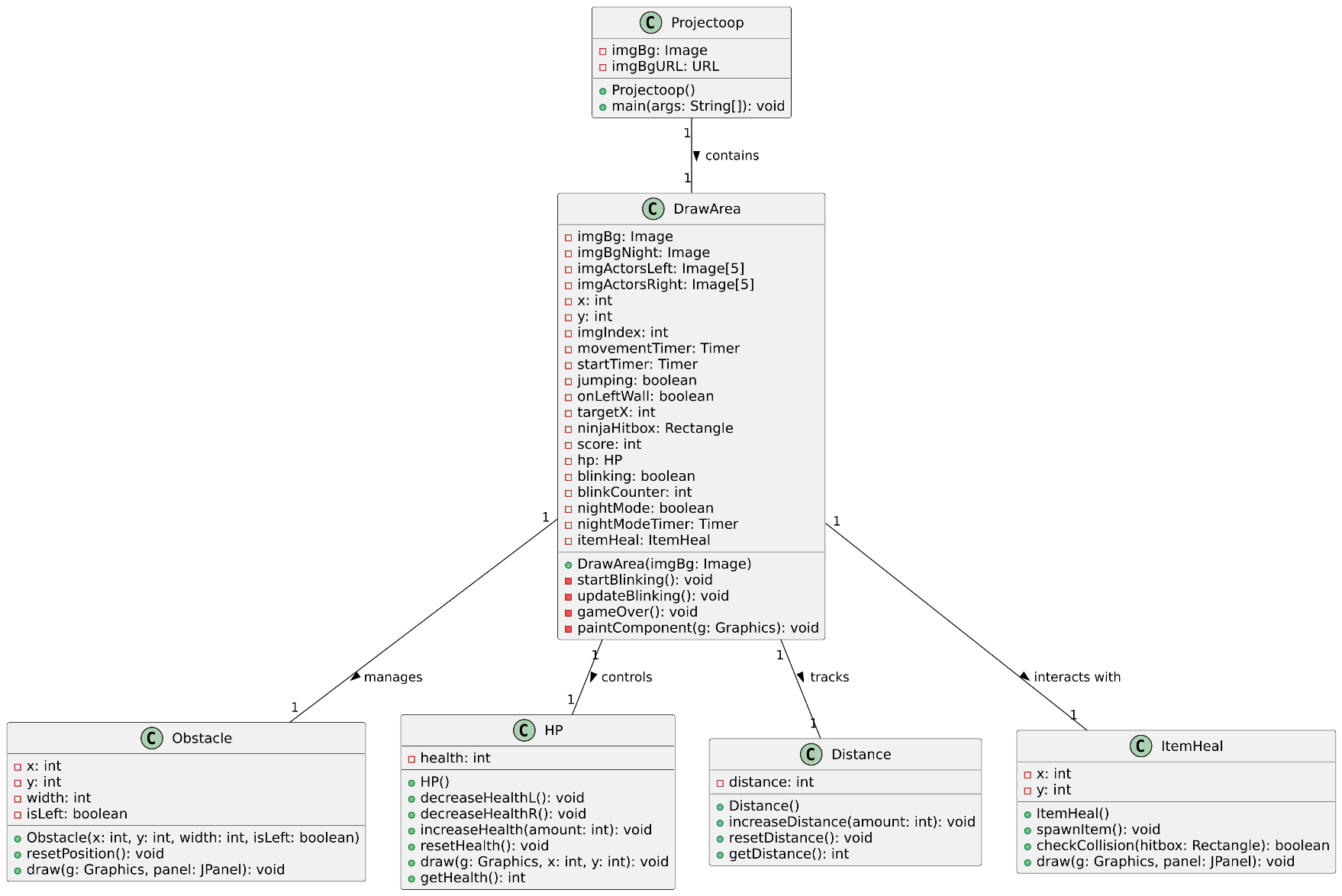
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ลำดับ** | **รายการ** | **1 - 7** | **8 - 24** | **25 - 31** |
| 1 | หารูปตัวละครและทำกราฟิกต่างๆ |  |  |  |
| 2 | ศึกษาเอกสารและข้อมูลที่เกี่ยวข้อง |  |  |  |
| 3 | ลงมือเขียนโปรแกรม |  |  |  |
| 4 | จัดทำเอกสาร |  |  |  |
| 5 | ตรวจสอบและแก้ไขข้อผิดพลาด |  |  |  |

**บทที่ 2**

**การพัฒนา**

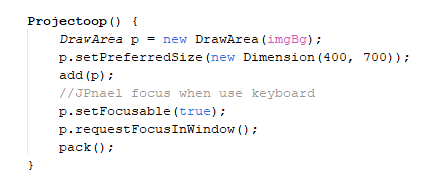
**2.1 เนื้อเรื่องย่อและวิธีการเล่น**

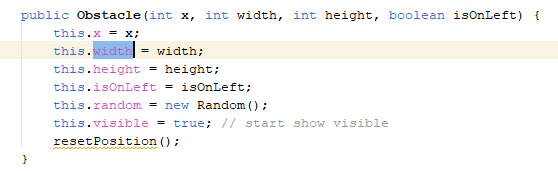
ทัตสึ เบนจิโร่ เป็นนินจาฝึกหัดในหมู่บ้านไคเซน แต่เขาเป็นคนที่มีทักษะนินจาน้อยที่สุดในหมู่บ้าน จึงไม่ค่อยมีเพื่อน ทำให้ทัตสึ เบนจิโร่ จึงต้องออกเดินทางเพื่อฝึกหนักกับวิชานินจา ในการเล่นเกมนินจาจะวิ่งขึ้นไปบนหุบเขา ใช้ปุ่ม space bar ในการหลบสิ่งกีดขวางเพื่อเก็บคะแนนจากระยะทางการเดินทาง

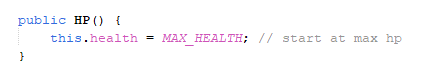
**2.2 Class Diagram**

**2.4 การทำงานในส่วนต่างๆ**

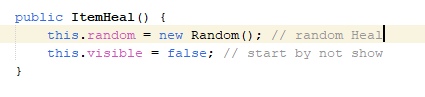
**2.4.1 Constructor**



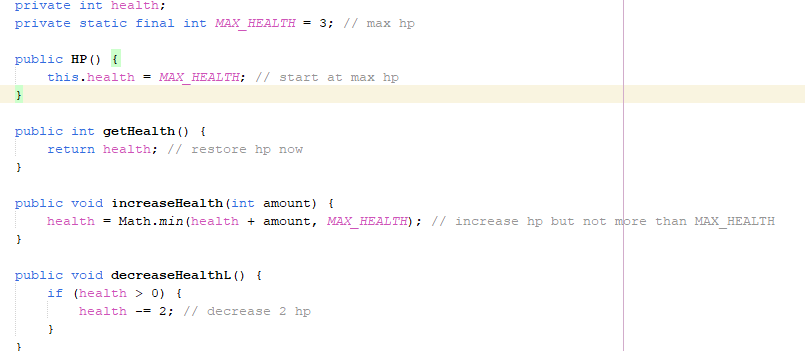
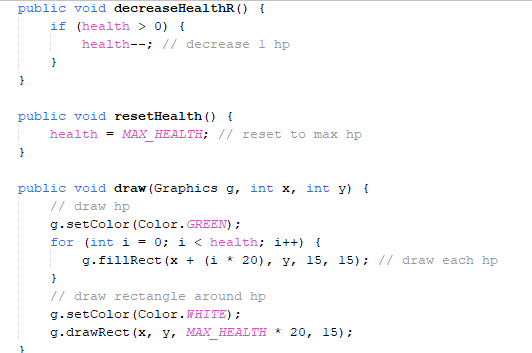
ในคลาส Projectoop constructor ใช้ในการสร้างหน้าต่างเกมและจัดการการตั้งค่าเบื้องต้นของการแสดงผล โดยจะมีการสร้างอ็อบเจ็กต์ DrawArea และตั้งค่าขนาดต่างๆ และสุดท้ายจะทำการตั้งค่าให้กับ JFrame

Constructor นี้ใช้ในการสร้างอ็อบเจ็กต์ Obstacle และกำหนดค่าตำแหน่ง ขนาด และสถานะการแสดงผลของอุปสรรค เช่น x, width, height, isOnLeft และ visible และเรียกใช้ resetPosition() เพื่อรีเซ็ตตำแหน่งของอุปสรรค

Constructor นี้ใช้ในการสร้างอ็อบเจ็กต์ HP และกำหนดค่าของ health ให้เริ่มต้นที่ MAX\_HEALTH ซึ่งเป็นค่าสูงสุดของพลังชีวิต



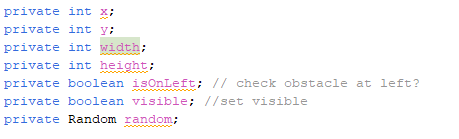
Constructor นี้ใช้ในการสร้างอ็อบเจ็กต์ ItemHeal และตั้งค่าเริ่มต้นของตัวแปร random และ visible ให้เป็น false ซึ่งหมายความว่าไอเทม Heal จะไม่แสดงผลจนกว่าจะมีการเรียกให้มันแสดงขึ้นมา

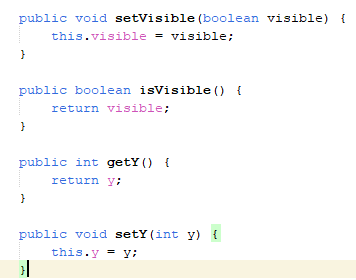
**** **2.4.2 Encapsulation**

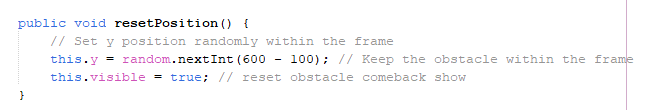
-ตัวแปร health ถูกกำหนดเป็น **private** ซึ่งหมายความว่าไม่สามารถเข้าถึงได้โดยตรงจากภายนอกคลาส

-การเปลี่ยนแปลงค่าของ health ต้องทำผ่านเมธอด เช่น increaseHealth(), decreaseHealthL(), decreaseHealthR() และ resetHealth()

-นอกจากนี้ยังมีเมธอด getHealth() สำหรับดึงค่าของ health ออกมาใช้ภายนอกคลาส

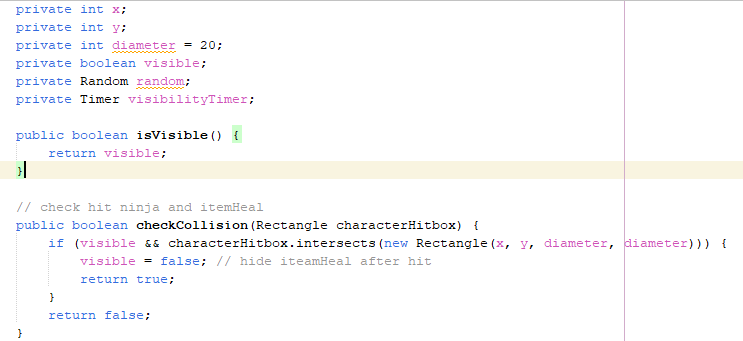






-ตัวแปร x, y, width, height, isOnLeft, และ visible ถูกกำหนดเป็น **private** ทำให้ไม่สามารถเข้าถึงจากภายนอกคลาสได้โดยตรง

-การเข้าถึงและเปลี่ยนแปลงค่าเหล่านี้ต้องทำผ่านเมธอด **getter/setter** เช่น setVisible(), isVisible(), getY(), และ setY()

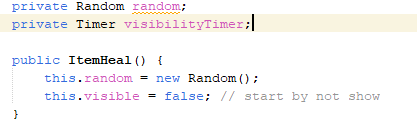


-ตัวแปร x, y, diameter, visible, random, และ visibilityTimer ถูกกำหนดเป็น **private**

-การเข้าถึงค่า visible จะทำได้ผ่าน **getter method** คือ isVisible()

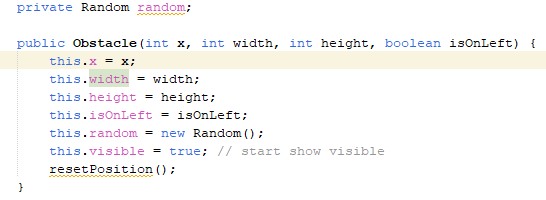
-เมธอด checkCollision() ใช้ในการตรวจสอบการชนระหว่าง ItemHeal กับ ninja และซ่อน ItemHeal เมื่อมีการชน

**2.4.3 Composition**

****

-ในคลาส ItemHeal จะเห็นว่า Random และ Timer ถูกสร้างขึ้นภายในคลาส ItemHeal ซึ่งเป็นการใช้ **Composition** โดยที่ ItemHeal จะประกอบไปด้วย Random (สำหรับการสุ่มตำแหน่งของไอเทม) และ Timer (สำหรับการกำหนดเวลาการแสดงของไอเทม)

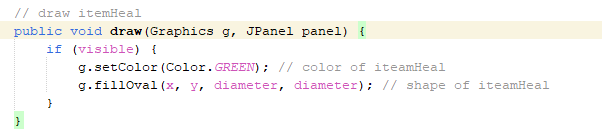
-การใช้ Random และ Timer ในคลาส ItemHeal เป็นการสร้างความสัมพันธ์ระหว่างคลาส ItemHeal กับคลาสเหล่านี้ในลักษณะของการรวมเป็นส่วนประกอบ (composition)



ในคลาส Obstacle การสร้างออบเจ็กต์ของ Random ถือเป็นการใช้ **Composition** ในที่นี้ Obstacle ประกอบด้วย Random ซึ่งจะช่วยในการสุ่มค่าตำแหน่ง y ในการวางอุปสรรคใหม่

**** **2.4.4 polymorphism**

เมธอด draw ในคลาส Obstacle จะใช้ Graphics ในการวาดรูปสี่เหลี่ยม โดยที่มันจะเปลี่ยนสีขึ้นอยู่กับค่าของ isOnLeft ซึ่งเป็นการแสดงถึงการทำงานที่แตกต่างกันตามชนิดของออบเจ็กต์ที่เรียกใช้



เมธอด draw ในคลาส ItemHeal ใช้ Graphics เพื่อวาดรูปวงกลมที่มีสีเขียว ซึ่งแตกต่างจากการวาดสี่เหลี่ยมในคลาส Obstacle



ในเมธอด paintComponent ของคลาส DrawArea มีการใช้ drawImage เพื่อวาดภาพของตัวละครนินจา (เป็นการเรียกใช้งาน polymorphism จากการเลือกภาพของนินจาจากอาร์เรย์ที่แตกต่างกันตามทิศทางการเคลื่อนไหว)

**2.4.5 Abstract**

ในโปรแกรมนี้ยังไม่มีการใช้คลาสหรือเมธอดนามธรรม (Abstract) แต่สามารถใช้เพื่อสร้างโครงสร้างที่สามารถกำหนดในคลาสลูกได้ ตัวอย่างเช่น การสร้าง abstract class GameObject เพื่อกำหนดลักษณะพื้นฐานของวัตถุในเกม เช่น Obstacle และ ItemHeal ที่อาจมีฟังก์ชัน draw() หรือ checkCollision() เป็นนามธรรม เพื่อให้คลาสลูกต้องกำหนดการทำงานของเมธอดเหล่านี้ตามรายละเอียดของแต่ละคลาส

**2.4.6 Inheritance**

-ในโปรแกรมนี้ยังไม่มีการใช้ Inheritance แต่สามารถใช้ได้หากต้องการขยายความสามารถของโปรแกรม เช่น การสร้างคลาส Ninja ที่สืบทอดมาจากคลาสพื้นฐาน Character เพื่อใช้คุณสมบัติทั่วไป เช่น การเคลื่อนที่ การชน

-การใช้ Inheritance ทำให้สามารถเพิ่มคลาสใหม่ ๆ ที่มีความสามารถสืบทอดจากคลาสแม่ได้อย่างมีประสิทธิภาพ โดยไม่ต้องเขียนโค้ดใหม่

**2.5 GUI**



**1.Background Image (ภาพพื้นหลัง)**:

ภาพพื้นหลังเป็นฉากภูเขาและบ้านที่อยู่ด้านซ้ายและขวาของหน้าจอ ซึ่งถูกวาดโดยใช้ imgBg หรือ imgBgNight ที่เปลี่ยนไปตามโหมดกลางวันหรือกลางคืน ซึ่งโค้ดจะสลับภาพพื้นหลังเป็นกลางคืนทุก ๆ 500 เมตรในเกมโดยใช้ตัวแปร nightMode และ nightModeTimer

2.**Distance Display (แสดงระยะทาง)**:

ข้อความที่แสดงระยะทางในหน่วย "เมตร" อยู่ที่มุมบนซ้ายของหน้าจอ ซึ่งวาดขึ้นโดยใช้ g.drawString("Distance: " + distance.getDistance() + " m", 10, 20); ในฟังก์ชัน paintComponent ของคลาส DrawArea

3.**Health Bar (แถบพลังชีวิต)**:

แถบพลังชีวิตของตัวละครอยู่ที่มุมบนซ้ายของหน้าจอ วาดขึ้นในสีเขียวโดยฟังก์ชัน draw ของคลาส HP ซึ่งจะแสดงจำนวนช่องพลังชีวิตตามค่าของตัวแปร health

4.**Obstacles (สิ่งกีดขวาง)**:

สิ่งกีดขวางมีสองฝั่ง คือฝั่งซ้าย (สีแดง) และฝั่งขวา (สีเหลือง) ซึ่งจะปรากฏเป็นสี่เหลี่ยมในตำแหน่งต่าง ๆ บนหน้าจอ ถูกสร้างและควบคุมการแสดงผลโดยคลาส Obstacle และถูกเรียกใช้ใน draw ภายใน DrawArea

5.**Ninja Character (ตัวละครนินจา)**:

ตัวละครนินจาอยู่ตรงกลางหน้าจอ สามารถกระโดดจากฝั่งซ้ายไปฝั่งขวาได้ โดยภาพของตัวละครจะถูกเลือกจาก imgActorsLeft หรือ imgActorsRight ตามตำแหน่งของตัวละครและวาดขึ้นที่ตำแหน่ง (x, y) ผ่าน g.drawImage(currentImages[imgIndex], x, y, 200, 140, this);



1.**Background Image (ภาพพื้นหลัง)**

ภาพพื้นหลังได้เปลี่ยนเป็นโหมดกลางคืน มีพระจันทร์เต็มดวงและท้องฟ้าที่มีเมฆและดวงดาว ซึ่งเกิดจากการสลับ imgBgNight เข้ามาแสดงเมื่อระยะทางเกิน 500 เมตร โดยมีการควบคุมผ่านตัวแปร nightMode ในโค้ด

2.**Distance Display (แสดงระยะทาง)**

ตัวเลขระยะทางได้เปลี่ยนไปเป็น "517 m" ซึ่งแสดงถึงการเดินทางของตัวละครและระยะทางที่สะสมได้จากการเล่นเกม

3.**Health Bar (แถบพลังชีวิต)**

แถบพลังชีวิตของตัวละคร (สีเขียว) ยังคงอยู่ในตำแหน่งเดิมที่มุมซ้ายบนของหน้าจอ และมีการแสดงผลที่เต็ม (สามช่อง) ซึ่งบ่งบอกว่าตัวละครมีพลังชีวิตเต็มอยู่ในขณะนี้

4.**Obstacles (สิ่งกีดขวาง)**

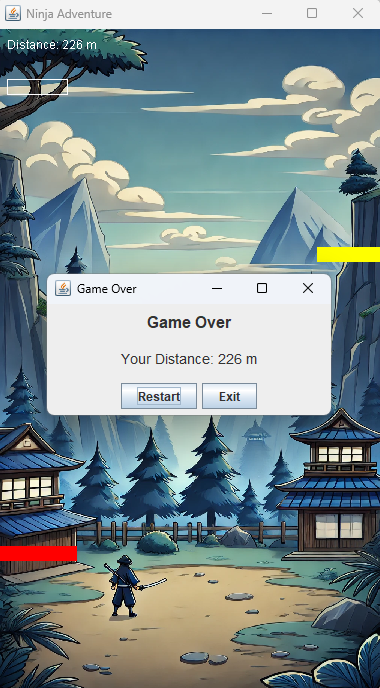
สิ่งกีดขวางยังคงมีอยู่ทั้งฝั่งซ้าย (สีแดง) และฝั่งขวา (สีเหลือง) โดยปรากฏในตำแหน่งที่แตกต่างกันจากภาพก่อนหน้า ซึ่งบ่งบอกว่าการวางตำแหน่งสิ่งกีดขวางนั้นสุ่มเปลี่ยนที่ตามการตั้งค่าในคลาส Obstacle

5.**Ninja Character (ตัวละครนินจา)**

ตัวละครนินจาในโหมดกลางคืนจะอยู่ตรงกลางหน้าจอ มีเงาที่ชัดเจนซึ่งสะท้อนถึงภาพลักษณ์ที่เข้ากับบรรยากาศกลางคืน

6.**Item Heal (ไอเทมเพิ่มพลังชีวิต)**

ไอเทมเพิ่มพลังชีวิตปรากฏเป็นวงกลมสีเขียวด้านขวาของหน้าจอ แสดงให้เห็นว่ามันสุ่มเกิดขึ้นเพื่อให้ผู้เล่นเก็บ ซึ่งจะหายไปหลังจาก 10 วินาทีหรือเมื่อผู้เล่นเก็บมันได้ (โดยตรวจสอบการชนกับตัวละครในโค้ด ItemHeal)



1.**Game Over Popup (หน้าต่างแสดงผลเกมจบ)**

1.มีหน้าต่างข้อความขนาดเล็กแสดงอยู่ตรงกลางของหน้าจอ โดยในหน้าต่างนี้มีข้อความ "**Game Over**" อยู่ที่ส่วนบน ซึ่งเป็นการแจ้งเตือนว่าผู้เล่นจบเกมแล้ว

2.มีการแสดงข้อความ "**Your Distance: 226 m**" เพื่อแสดงระยะทางที่ผู้เล่นได้เดินทางก่อนที่จะพ่ายแพ้

3.มีปุ่มให้เลือกสองปุ่ม คือ **Restart** และ **Exit**

3.1ปุ่ม **Restart**: เพื่อเริ่มเกมใหม่ ซึ่งจะเรียกใช้เมธอดที่รีเซ็ตค่าสถานะทั้งหมดของเกม (รีเซ็ตระยะทาง พลังชีวิต และตำแหน่งของตัวละคร)

3.2ปุ่ม **Exit**: เพื่อออกจากเกม

2.**Background (ภาพพื้นหลัง)**

ภาพพื้นหลังในโหมดกลางวันยังคงแสดงอยู่ ซึ่งเป็นการกลับมาที่ภาพพื้นหลังเดิมตามโค้ดที่กำหนดไว้ โดยหน้าต่าง Game Over เป็นเพียงหน้าต่างเล็ก ๆ ที่ซ้อนอยู่ด้านบนของ GUI หลัก

3.**Distance Display (แสดงระยะทาง)**

มีการแสดงผลระยะทางอยู่ที่มุมซ้ายบนของหน้าจอเช่นเดิม แสดงระยะทางสะสมของผู้เล่นที่ 226 เมตร ซึ่งสอดคล้องกับระยะทางที่แสดงในหน้าต่าง Game Over

4.**Health Bar (แถบพลังชีวิต)**

แถบพลังชีวิตของตัวละครเป็นสีขาวทั้งหมด ซึ่งบ่งบอกว่าตัวละครสูญเสียพลังชีวิตจนหมด ส่งผลให้เกิดสถานะ Game Over

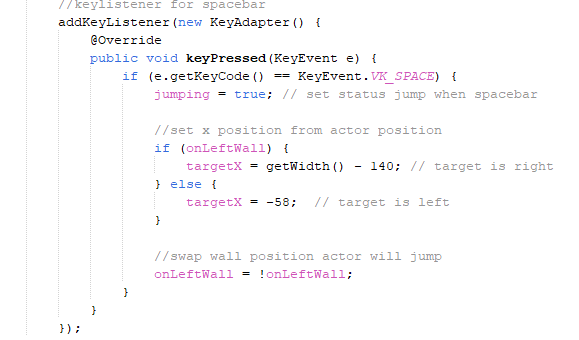
5.**Obstacles and Ninja Character (สิ่งกีดขวางและตัวละครนินจา)**

5.1ตัวละครนินจายังคงอยู่ในตำแหน่งเดิมที่ด้านล่างของหน้าจอ

5.2สิ่งกีดขวางยังคงแสดงอยู่ทั้งฝั่งซ้าย (สีแดง) และฝั่งขวา (สีเหลือง) ซึ่งเป็นสถานะหยุดค้างตามตำแหน่งสุดท้ายก่อนเกิด Game Over

**2.6 Event handling**

1. การจัดการการกดปุ่ม (KeyEvent) ใน DrawArea

****เมื่อผู้ใช้กดปุ่ม Spacebar (เพื่อกระโดด) เราจะใช้ KeyListener เพื่อจับเหตุการณ์ที่เกิดขึ้นจากการกดปุ่มบนคีย์บอร์ด ซึ่งจะกระตุ้นการกระทำต่างๆ เช่น การกระโดดของตัวละคร

-เราใช้ addKeyListener เพื่อเพิ่มตัวจับเหตุการณ์การกดปุ่ม (KeyEvent) ในคอมโพเนนต์ DrawArea ที่เป็น JPanel

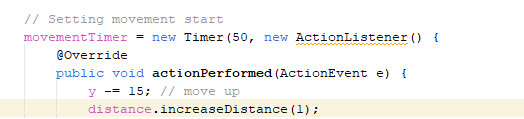
-ฟังก์ชัน keyPressed ถูกเรียกใช้เมื่อผู้ใช้กดปุ่ม ซึ่งจะตรวจสอบว่าเป็นปุ่ม **Spacebar** (VK\_SPACE) หรือไม่ ถ้าใช่จะตั้งค่าตัวแปร jumping เป็น true และกำหนดตำแหน่งที่ตัวละครจะกระโดดไป (ใช้ targetX และ onLeftWall เพื่อกำหนดทิศทางการกระโดด)

-requestFocusInWindow() ใน constructor ของ DrawArea ช่วยให้ JPanel สามารถรับโฟกัสและรองรับการจับเหตุการณ์คีย์

2. การจัดการ Timer (สำหรับการเคลื่อนไหวของตัวละคร และการตั้งเวลาในเกม):

โปรแกรมใช้ Timer เพื่อควบคุมการเคลื่อนไหวของตัวละครและอุปสรรค รวมถึงการตั้งเวลาให้ไอเทม ItemHeal หายไปหลังจากที่แสดงในหน้าจอเป็นเวลา 10 วินาที

ตัวอย่างการใช้ Timer เพื่อเคลื่อนที่ของตัวละคร:



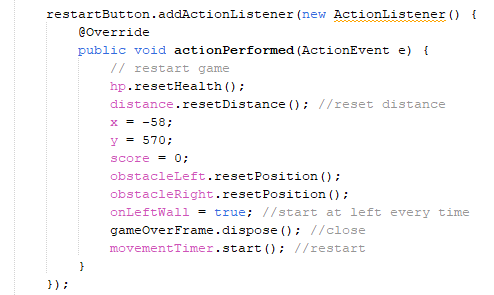
-Timer จะทำงานทุกๆ 50 มิลลิวินาที (ตามพารามิเตอร์แรก) โดยที่ ActionListener จะทำงานทุกครั้งที่ Timer ถูกกระตุ้น

-ในที่นี้ y -= 15 จะทำให้ตัวละครเคลื่อนที่ขึ้นไป (ลดค่า y ของตำแหน่ง) และเพิ่มระยะทางที่ตัวละครเดินไปในตัวแปร distance

-หลังจากแต่ละการเคลื่อนไหว repaint() จะถูกเรียกเพื่อให้หน้าจออัปเดตและวาดภาพใหม่(คำสั่งนี้อยู่ด้านล่างสุดเพราะโปรแกรมต้องรันเหตุการณ์ทั้งหมดให้เสร็จสิ้นจึงจะทำการวาดภาพใหม่ตามเงื่อนไขที่กำหนด)

3. การจัดการการคลิกปุ่มในหน้าต่าง "Game Over":

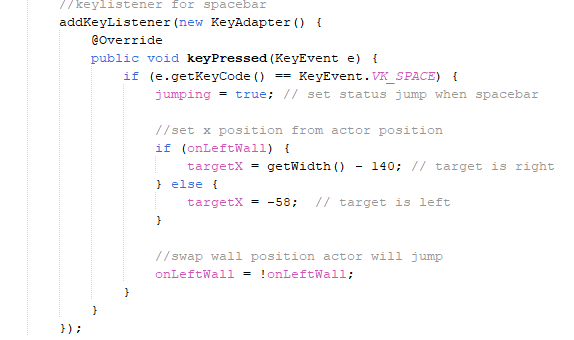
เมื่อเกมจบลง (เช่น ตัวละครพลังชีวิตหมด) หน้าต่าง "Game Over" จะปรากฏขึ้น ซึ่งจะมีปุ่ม Restart และ Exit ที่ให้ผู้เล่นเลือกเพื่อรีสตาร์ทเกมหรือออกจากโปรแกรม



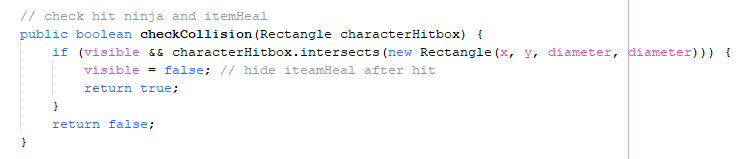
-เราสร้างปุ่ม **Restart** และเพิ่ม ActionListener ซึ่งจะเรียกฟังก์ชัน actionPerformed เมื่อปุ่มถูกคลิก

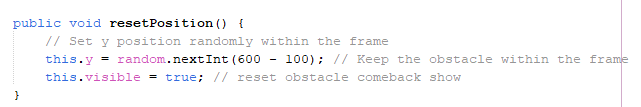
-เมื่อผู้เล่นกดปุ่ม **Restart** จะทำการรีเซ็ตค่าพลังชีวิต, ระยะทาง, และตำแหน่งต่างๆ เพื่อเริ่มเกมใหม่ และจะเริ่ม movementTimer เพื่อให้เกมกลับมาเล่นได้

-gameOverFrame.dispose() จะปิดหน้าต่าง "Game Over" หลังจากผู้เล่นเลือกรีสตาร์ท

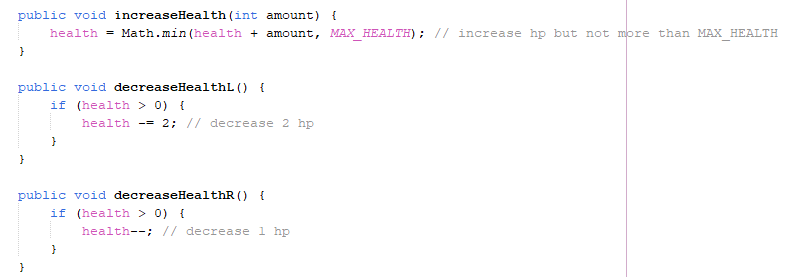
**2.7 Algorithm**

การเคลื่อนไหวของตัวละคร(Movement Algorithm)

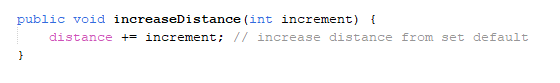
การตรวจจับการชน(Collision Detection Algorithm)



การเคลื่อนไหวของอุปสรรค(Obstacle Movement Algorithm)



การเพิ่ม/ลดพลังชีวิต(Health Management Algorithm)

การคำนวณระยะทาง(Distance Tracking Algorithm)

**บทที่ 3**

**สรุปผลและข้อเสนอแนะ**

**3.1 ปัญหาที่พบ**

1.บัคตอนกระโดดแล้วตัวละครหันผิด(เป็นในบางครั้งตอนที่กด Restart หลังจาก HP หมด)

2.เนื่องจากการบริหารจัดการเวลาของผู้สร้างไม่ดีเท่าที่ควร ทำให้อาจจะยังขาดบางฟีเจอร์สำคัญ เช่น ปุ่ม start และสิ่งที่อาจารย์ให้เพิ่มเติมก็ทำได้เพียง 1 อย่างคือเปลี่ยนด่านได้ แต่ไม่ได้ทำในส่วนของการจับเวลาในแต่ละด่านอย่างที่อาจารย์ได้เพิ่มมา

**3.2 จุดเด่นของโปรแกรมที่ไม่เหมือนใคร**

การเคลื่อนไหวของตัวละครที่สมจริงระดับนึงเพราะใช้ภาพที่ต่อกันหลายๆอัน

**3.3 ข้อเสนอแนะ**

ฝากถึงน้องปีต่อๆไปว่าอย่าย่อท้อต่อวิชานี้ Lab อาจจะเยอะบ้าง ฉะนั้นวิชานี้ควรเคลียร์ให้เสร็จแต่ละสัปดาห์ไปเลย จะได้ไม่มาท้อทีหลังตอนนั่งทำ Lab ส่งอาจารย์ สู้ๆนะเด็กๆ!!!