รายงาน

เรื่อง Brick - Breaker

จัดทำโดย

1.นภัสสร	**พิสุทธิ์ธาราชัย**	**6010613476**
2.ณัฐฑริกา	**บุญมี**	**6010613484**
3.กณวรรธน์	**เหลืองอรุณโรจน์**	**6010613583**

นักศึกษาชั้นปีที่ 4 คณะวิศวกรรมศาสตร์ สาขาวิชาวิศวกรรมคอมพิวเตอร์

มหาวิทยาลัยธรรมศาสตร์

นำเสนอโดย

ผศ.ดร.พิศาล แก้วประภา

อาจารย์ประจำวิชา วพ.341 การวิเคราะห์และออกแบบโปรแกรมเชิงวัตถุ

ภาคการศึกษาที่ 2 ปีการศึกษา 2563

มหาวิทยาลัยธรรมศาสตร์ ศูนย์รังสิต ปทุมธานี 12120

คำนำ

รายงานเล่มนี้เป็นส่วนหนึ่งของวิชา วพ.341 การวิเคราะห์และออกแบบโปรแกรมเชิงวัตถุ ซึ่งมีเนื้อหาเกี่ยวกับการจัดทำเกมโดยใช้องค์ความรู้ที่ได้ศึกษาจากเนื้อหาวิชา วพ.341 คณะผู้จัดทำมีความสนใจในด้านการนำความรู้ต่าง ๆ ที่ได้ศึกษา มาพัฒนาและต่อยอด ซึ่งคณะผู้จัดทำได้พัฒนาเกม Brick - Breaker ซึ่งเป็นเกมที่เกี่ยวกับการใช้ลูกบอลเพื่อทำลายวัตถุ โดยคณะผู้จัดทำมีความต้องการที่จะพัฒนาเกมให้มีความยากและท้าทายเพื่อเพิ่มความสนุกให้กับผู้เล่นทั้งนี้ คณะผู้จัดทำต้องขอบคุณ ผศ.ดร.พิศาล แก้วประภา และเพื่อนๆ ที่คอยให้คำปรึกษาและส่งเสริมจนสามารถพัฒนางานให้สำเร็จลุล่วงไปด้วยดี หากมีข้อผิดพลาดประการใด ต้องขออภัยไว้ ณ ที่นี้ด้วย

คณะผู้จัดทำ

Brick - Breaker

Problem Statement

เนื่องจาก Brick - Breaker เป็นที่รู้จักกันอย่างกว้างขวางและรู้จักกันอย่างดี ในด้านความสนุก ไม่ว่าผู้ใหญ่หรือเด็กก็รู้จักและสามารถเล่นได้อย่างง่ายดาย เนื่องจากเกมมีความสนุกและน่าสนใจ สามารถพัฒนาได้หลายรูปแบบ ผู้พัฒนาจึงสนใจที่จะสร้างความสนุกที่มากขึ้น โดยการสร้าง Brick – Breaker ที่จะสามารถทำให้ผู้เล่นรู้สึกสนุกสนานและเพลิดเพลินกับเกมนี้ได้

Objective

เพื่อพัฒนา Brick – Breaker ให้มีความหลากหลาย โดยการเพิ่มความสนุกสนานและความท้าทายให้กับผู้เล่นมากขึ้น

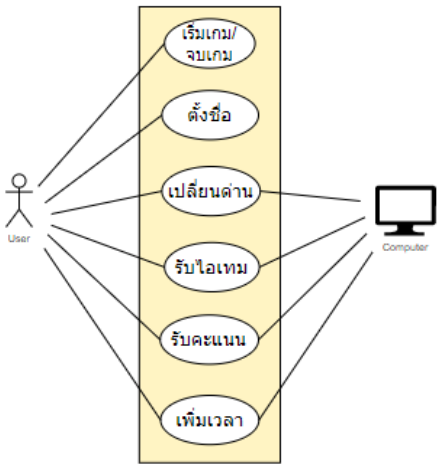
Function requirement

1. User เลือกเริ่มเล่นเกมได้
2. User สามารถตั้งชื่อผู้เล่นได้
3. User สามารถเปลี่ยนด่านได้
4. User สามารถรับไอเทมระหว่างเล่นเกมได้
5. เมื่อ User ชนะจะได้รับคะแนน
6. User สามารถใช้เวลาที่เหลือในด่านก่อนหน้า มาเพิ่มให้กับด่านถัดไปได้

Nonfunction requirement

1. มีการกำหนดเวลาในการเล่นในแต่ละด่าน
2. มีเอฟเฟกต์ประกอบฉากระหว่างเล่นเกมในแต่ละด่าน
3. มีเพลงประกอบฉากระหว่างเล่นเกมในแต่ละด่าน
4. มีการแสดงคะแนนปัจจุบันระหว่างเล่นเกม

Use Case Diagram



Use Case Description

Name: เปลี่ยนด่าน

Goal: User สามารถเปลี่ยนด่านได้ เมื่อชนะด่านปัจจุบัน

Summary: User ที่ตั้งชื่อแล้วสามารถเล่นเกม และ ผ่านด่านปัจจุบันได้ก่อนหมดเวลา จะสามารถเปลี่ยนด่านได้

Actors: User และ Computer

Precondition:

- User ต้องตั้งชื่อเพื่อเข้าเล่นเกม

Trigger:

- User เล่นเกมในด่านปัจจุบัน และ ชนะด่านก่อนหมดเวลา

Primary sequences:

1. Computer แสดงด่านแรก
2. User เล่นชนะในด่านนี้ก่อนหมดเวลา
3. Computer แสดงคะแนนที่ User สามารถทำได้ และ ให้เลือกเล่น ด่านถัดไป
4. User กดเลือก ด่านถัดไป
5. Computer ทำการเปลี่ยนด่าน และ แสดงด่านถัดไป

Alternative sequences:

1. User ไม่ชนะด่านปัจจุบันในเวลาที่กำหนด
   1.1 Computer แสดงหน้า “เล่นอีกครั้ง” / “ออกจากเกม”

Postconditions:

- User ได้เล่นด่านถัดไป

Name: รับไอเทม

Goal: User สามารถเลือกรับไอเทมระหว่างเล่นเกมได้

Summary: User ที่ตั้งชื่อแล้ว ระหว่างเล่นเกมสามารถรับไอเทมที่ Computer สุ่มให้ได้

Actors: User และ Computer

Precondition:

- User ต้องตั้งชื่อเพื่อเข้าเล่นเกม

Trigger:

- User ต้องรับไอเทม

Primary sequences:

1. User อยู่ในระหว่างการเล่นด่านปัจจุบัน
2. User ยิงลูกบอลไปโดนบล็อกที่มีไอเทมเสริมซ่อนอยู่
3. ไอเทมเสริมตกลงมา
4. User รับไอเทมที่ตกลงมา
5. Computer สุ่มตัวช่วย/ขัดขวางเกมให้กับ User

Alternative sequences:

1. User พลาดการรับไอเทม

    1.1 ไอเทมออกนอกหน้าจอ User ไม่ได้รับผลของไอเทม

Postconditions:

- User สามารถรับไอเทมเสริมระหว่างเล่นเกมเพื่อเพิ่มโอกาสชนะให้มากขึ้นได้
- User สามารถรับไอเทมขัดขวางระหว่างเล่นเกมเพื่อลดโอกาสในการชนะได้

Name: รับคะแนน

Goal: User ได้รับคะแนนที่แตกต่างกัน

Summary: User ที่ตั้งชื่อแล้ว สามารถเล่นเกม และ ผ่านด่านปัจจุบันได้ก่อนหมดเวลา จะได้รับคะแนนของด่านปัจจุบัน

Actors: User และ Computer

Precondition:

- User ต้องตั้งชื่อเพื่อเข้าเล่นเกม

Trigger:

- User เล่นเกมในด่านปัจจุบัน และชนะด่านก่อนหมดเวลา

Primary sequences:

1. User เล่นเกมในด่านปัจจุบัน
2. User ชนะเกมในด่านปัจจุบัน
3. Computer คำนวณคะแนนจากการเล่น
4. Computer แสดงคะแนนให้ User ทราบ

Alternative sequences:

1. User ไม่ชนะด่านปัจจุบันในเวลาที่กำหนด
   1.1 Computer ไม่คำนวนคะแนนในด่านปัจจุบันให้
   1.2 Computer แสดงหน้า “เล่นอีกครั้ง” / “ออกจากเกม”

Postconditions:

- User จะได้รับคะแนนที่ Computer คำนวนให้

Design Pattern

1. Factory Method Pattern

หลักการทำงาน

1.1) Factory Method จะมี abstract method 1 ตัว เอาไว้สร้าง object

1.2) เมื่อ client ต้องการใช้ object ก็จะมาเรียก abstract method ตัวนั้น เพื่อเอา object ไปใช้

1.3) Subclass เป็นคนกำหนดเองว่าจะสร้าง object จาก class ตัวไหน

2. Abstract Factory Pattern

หลักการทำงาน

2.1) Abstract Factory จะกำหนด interface ที่ใช้ในการสร้าง product แต่ละประเภท และมี abstract method ที่ใช้ในการสร้าง product ต่างๆ

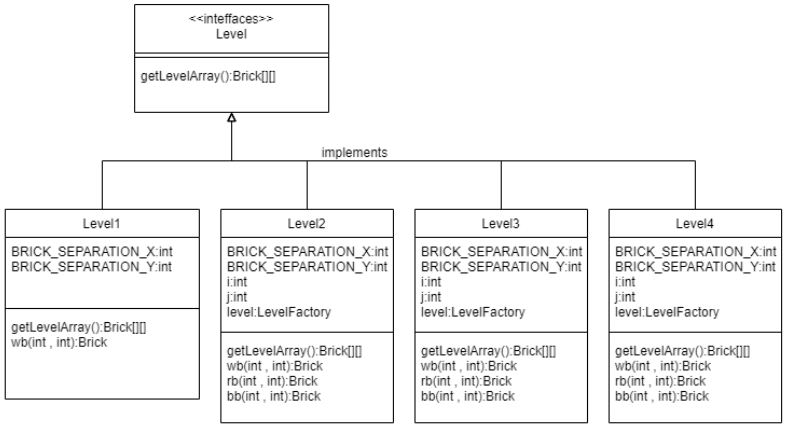
2.2) Subclass เป็นคนกำหนดเองว่า product ที่จะสร้างแต่ละตัว จะถูกสร้างจาก class ตัวไหน ซึ่งจะต้องเป็น class ที่อยู่ในกลุ่มเดียวกัน

2.3) เมื่อ client ต้องการ product ก็จะมาเรียก Subclass ให้ไปสร้าง product ต่างๆให้

ความแตกต่างระหว่าง Factory Method Pattern และ Abstract Factory Pattern

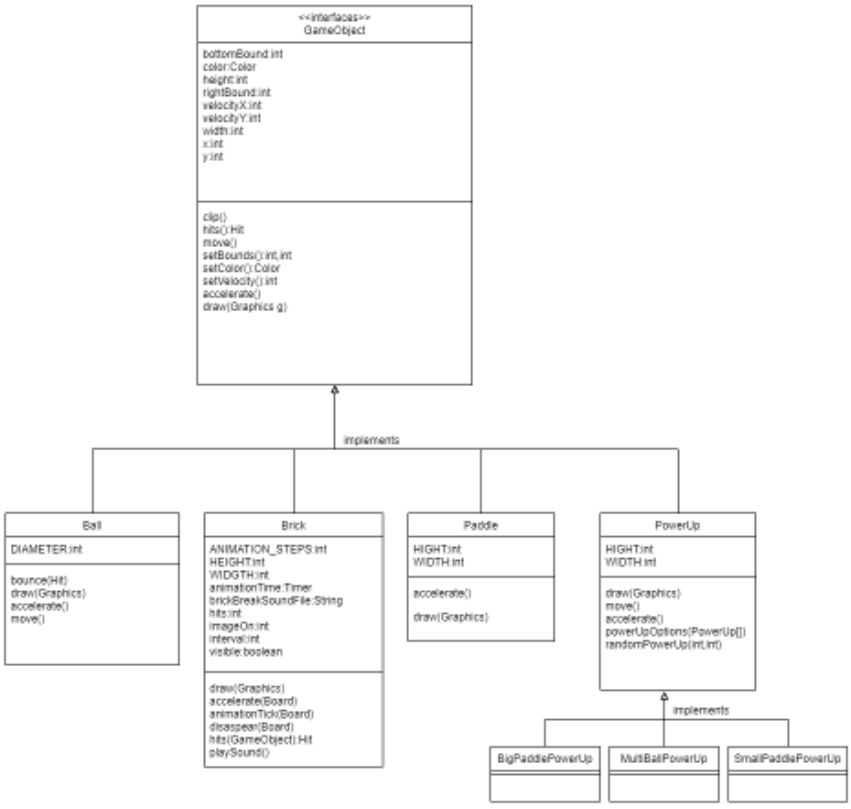
- Factory Method Pattern จะแก้ปัญหาผ่าน Inheritance โดยให้ Sub class เป็นคนจัดการ
- Factory Method Pattern จะสร้าง object โดยไม่ได้มีเรื่องเซตมาเกี่ยวข้อง
- Abstract Factory Pattern จะแก้ปัญหาผ่าน Composition โดยใช้คลาสนั้นๆไป implement เอาเอง
- Abstract Factory Pattern จะสร้าง object โดยคำนึงถึงเรื่องเซตด้วยเสมอ และ มันจะต้องสามารถสร้าง product อื่นๆที่อยู่ภายในเซตเหล่านั้นได้ด้วย

การใช้งาน Design Pattern ร่วมกับ Brick – Breaker



1. การใช้งาน Factory Method Pattern ร่วมกับ LevelFactory Class

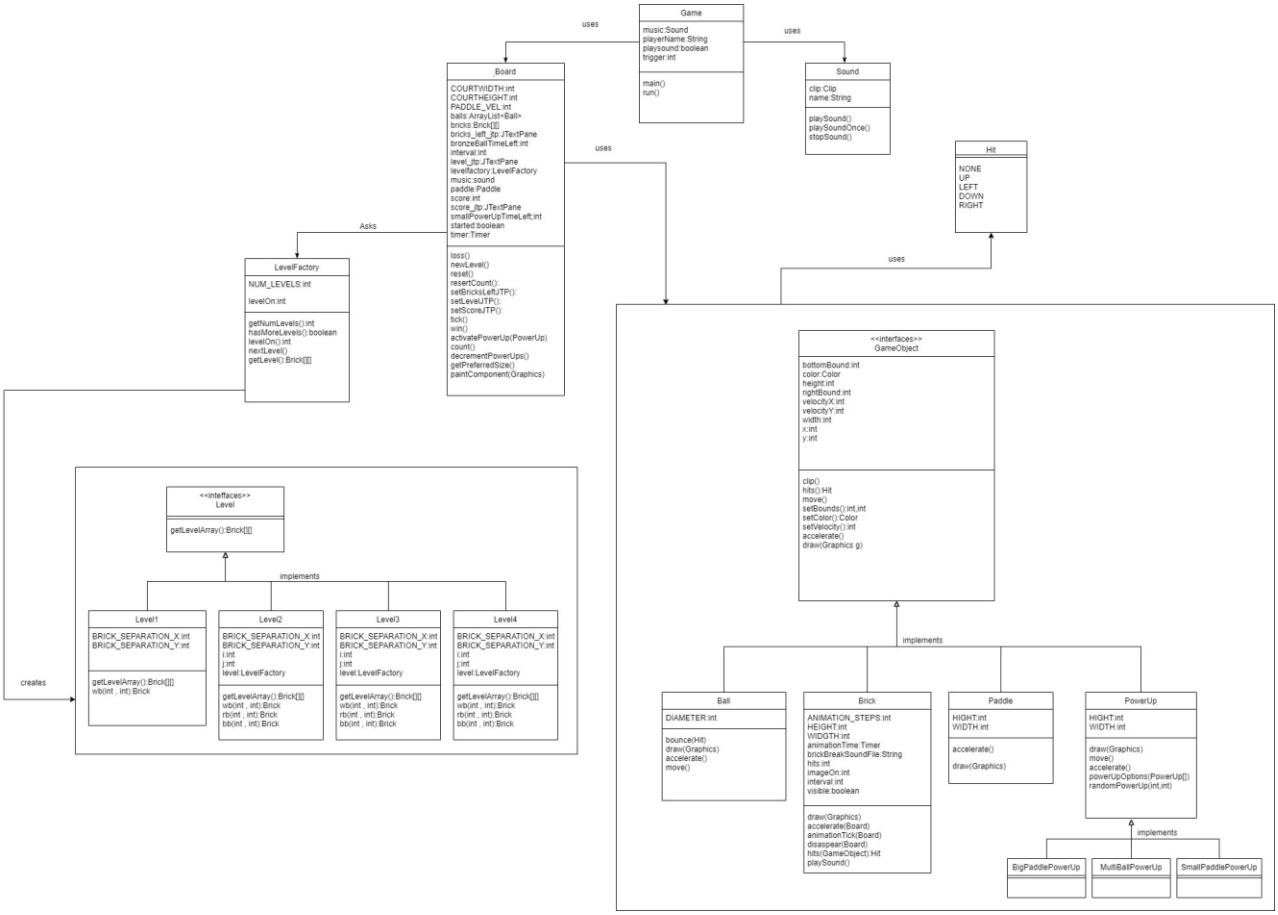
LevelFactory Class ทำหน้าที่ สร้างด่าน ซึ่งใน Brick – Breaker มีด่านหลายด่านจึงมีการประยุกต์ใช้ Factory Method Pattern โดยการสร้าง Interface ชื่อว่า Level แล้วมี class ต่างๆ ที่ implement Interface นี้ ซึ่ง Class ต่างๆ คือ ด่านที่อยู่ใน Brick – Breaker เราจะมองว่า Class เหล่านี้ทำหน้าที่เหมือนกันคือเป็นด่าน แต่เก็บรูปแบบของด่านที่แตกต่างกันไว้ ดังนั้นเราจึงเลือกใช้ Factory Method Pattern ที่ถูกมองว่าเป็นโรงงานที่ผลิตสิ่งต่างๆที่ทำหน้าที่เหมือนกันแต่ภายในไม่เหมือนกัน เช่น การผลิต ไอศกรีม ที่มีรสชาติแตกต่างกัน ซึ่งในกรณีนี้คือการผลิตด่านที่มีรูปแบบแตกต่างกัน โดยผลิตภัณฑ์ของโรงงานนี้คือ ด่านที่ 1(Level 1), ด่านที่ 2(Level 2),ด่านที่ 3(Level 3) และ ด่านที่ 4(Level 4)



2. การใช้งาน Abstract Factory Pattern ร่วมกับ Board Class

Board Class ทำหน้าที่ รับองค์ประกอบของเกมต่างๆเพื่อใช้ในการแสดงผลให้กับผู้เล่น โดย จะมีการแสดงในส่วนของด่านต่างๆ และองค์ประกอบอื่นๆ เช่น ลูกบอล(Ball), คาน(Paddle) ซึ่งองค์ประกอบอื่นๆ นั้น เราได้ทำการประยุกต์ใช้ Abstract Factory Pattern ในการทำงานร่วมกัน โดยการสร้าง Interface ชื่อว่า GameObject ที่ทำหน้าที่รวบรวม Class ที่สร้างองค์ประกอบที่แตกต่างกัน ได้แก่ ลูกบอล(Ball), คาน (Paddle), อิฐ(Brick) และ Class ที่ทำหน้าที่สร้างตัวช่วยหรือตัวขัดขวางให้กับผู้เล่น คือ PowerUp Class จะเห็นได้ว่า การใช้งาน Abstract Factory Pattern ช่วยในการรวบรวมองค์ประกอบที่แตกต่างกันแต่มีการใช้งานร่วมกัน ซึ่งสิ่งที่ Class ที่ implement Interface ที่ชื่อว่า GameObject นี้ สร้างจะถูกนำไปใช้ใน Board Class เพื่อทำให้การแสดงด่านให้กับผู้เล่นสมบูรณ์

UML Diagram



Class Game

- ทำหน้าที่เป็น Main Class โดยจะรวบรวมทุกส่วนของเกม เช่น Board sound เป็นต้น ให้มาแสดงผลในหน้าเดียวให้กับผู้เล่นได้เริ่มเกม และ ทำหน้าที่เปิดเสียงเพลง เสียงประกอบให้กับเกม

Class Sound

- ทำหน้าที่ดูแลเสียงประกอบ และ เสียงเพลงของเกม

Class Board

- ทำหน้าที่แสดงด่านต่างๆให้กับผู้เล่น รวมถึงแสดงส่วนประกอบของแต่ละด่าน ได้แก่ เวลาที่ใช้ในด่านนั้นๆ คะแนนในด่านปัจจุบัน ชื่อของผู้เล่น จำนวนของอิฐที่เหลือ รวมทั้งควบคุมการแพ้ชนะของผู้เล่น และองค์ประกอบอื่นๆ ได้แก่ ลูกบอล คาน อิฐ ตัวช่วยหรือตัวขัดขวาง

Class LevelFactory

- ทำหน้าที่สร้างด่านที่แตกต่างกันให้กับผู้เล่น โดยจะสร้างด่านต่างๆได้แก่ ด่านที่1 ด่านที่2 ด่านที่3 และด่านที่4 ผ่านการเรียกใช้ Class ที่ implement Interface ที่ชื่อว่า Level

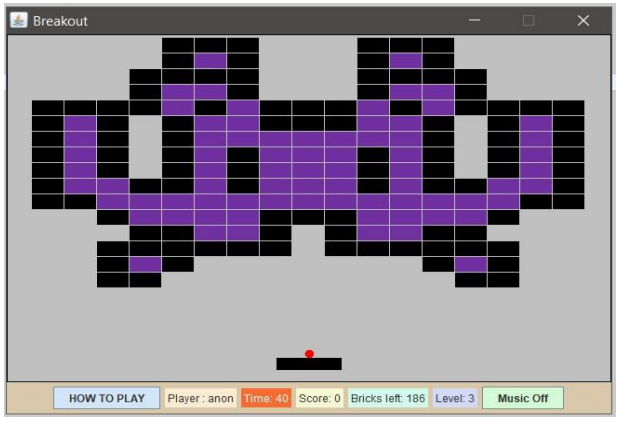
Class Interface Level

- ทำหน้าที่เป็นต้นแบบของด่านต่างๆที่ใช้ในการเล่นเกม โดย Class ต่อไปนี้ทำหน้าที่ implement Interface Level ได้แก่

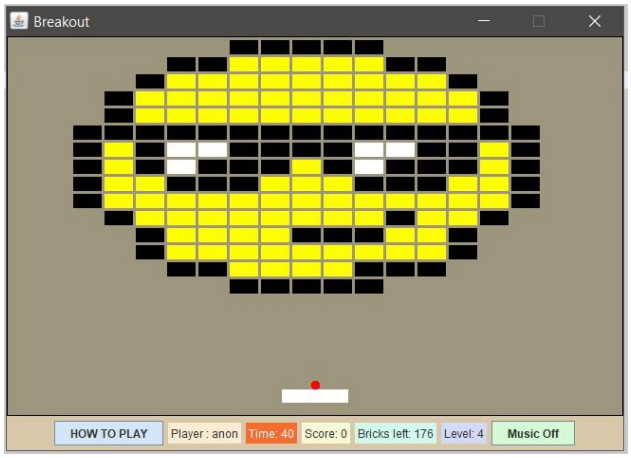
1. **Class Level1** ทำหน้าที่สร้างรูปแบบของอิฐของด่านที่1

2. **Class Level2** ทำหน้าที่สร้างรูปแบบของอิฐของด่านที่2

3. **Class Level3** ทำหน้าที่สร้างรูปแบบของอิฐของด่านที่3



4. **Class Level4** ทำหน้าที่สร้างรูปแบบของอิฐของด่านที่4



Class Interface GameObject

- ทำหน้าที่เป็นต้นแบบขององค์ประกอบในเกม โดย Class ต่อไปนี้ทำหน้าที่ implement Interface GameObject ได้แก่

1. **Class Ball** ทำหน้าที่สร้างลูกบอลให้กับด่าน
2. **Class Brick** ทำหน้าที่สร้างอิฐให้กับด่าน
3. **Class Paddle** ทำหน้าที่สร้างคานให้กับด่าน
4. **Class PowerUp** ทำหน้าที่ในการสร้างตัวช่วยหรือตัวขัดขวางให้กับด่าน โดยมี Class ของตัวช่วยหรือตัวขวาง implement ได้แก่

4.1 **Class BigPaddlePowerUp** ทำหน้าที่สร้างตัวช่วยที่ทำให้คานมีขนาดใหญ่ขึ้น

4.2 **Class MultiBallPowerUp** ทำหน้าที่สร้างตัวช่วยที่ทำให้ลูกบอลสามารถแตกออกเป็นหลายลูก

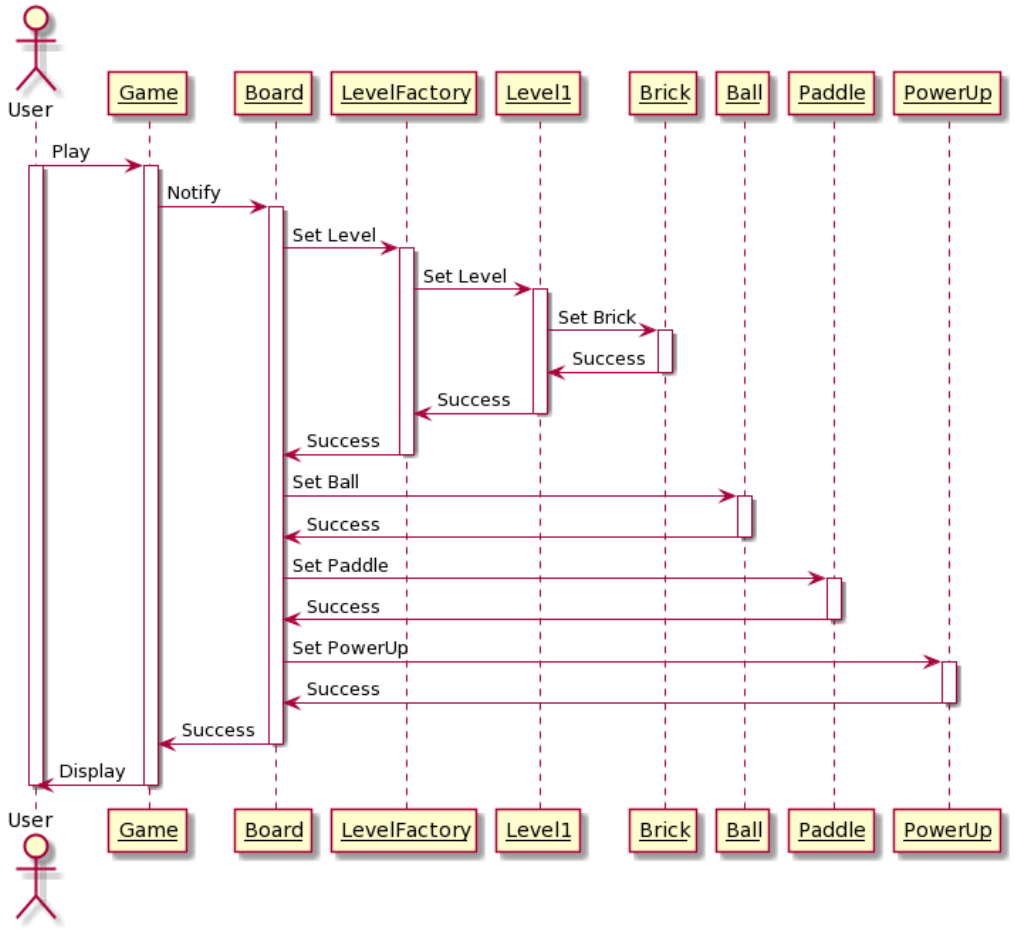
4.3 **Class SmallPaddlePowerUp** ทำหน้าที่สร้างตัวขัดขวางที่ทำให้คานมีขนาดเล็กลง

Class Hit

- ทำหน้าที่ควบคุมทิศทางของลูกบอลเมื่อเกิดการชนกับอิฐ

Sequence Diagram

การเล่นเกมในด่านที่ 1



เมื่อผู้เล่นทำการกดเล่นเกม Class Game จะถูกเรียกใช้ ซึ่ง Class Game จะทำการเรียก Class Board เพื่อทำการสร้างหน้าของเกมให้กับผู้เล่น โดยที่ Class Board จะทำหน้าที่เรียกใช้ Class LevelFactory เพื่อให้ทำการสร้างด่านที่ 1 (Class Level1) และส่งกลับมายัง Class Board เมื่อ Class Board ได้รับแล้วจะทำการสร้างองค์ประกอบอื่นๆคือ ลูกบอล(Ball), คาน(Paddle) โดยทำการเรียกใช้ Class Ball และ Class Paddle ตามลำดับ หลังจากนั้น Class Board จะมีการสร้างตัวช่วยเหลือหรือตัวขัดขวางให้กับผู้เล่นโดนการเรียก Class PowerUp และ Class PowerUp จะทำการส่งข้อมูลไปให้กับ Class Board เมื่อ Class Board ได้รับข้อมูลที่สร้างเสร็จเรียบร้อยแล้วจะทำการส่งไปให้กับ Class Game เพื่อทำการแสดงให้กับผู้เล่นได้เห็นและทำการเล่นต่อไป