Mr. Bean Finding Teddy

ชื่อเกม : Mr. Bean: Cat Escape for Teddy

ประเภทเกม : เกมวิ่งไม่รู้จบ (Endless Runner)

บทที่ 1

บทนำ

- **ที่มาและความสำคัญของโปรเจ็ค**

โครงงานนี้จัดทำเพื่อวัดผลความเข้าใจและความสามารถในการประยุกต์ เนื้อหาในรายวิชา**** โดยนำเนื้อหาจากบทเรียนมาออกแบบโปรเจคของเกมส์ ให้สอดคล้องกับแต่ละหัวข้อ

- **ประเภทของโครงการ**

1. ใช้เครื่องมือในการพัฒนา : Java
2. ใช้หลักการแบบ OOP

- **ประโยชน์ที่ได้รับ**

1. พัฒนาทักษะการตอบสนองและการตัดสินใจ ผู้เล่นต้องหลบหลีกสิ่งกีดขวาง
2. ฝึกความอดทนในการเอาชนะ
3. สร้างความเพลิดเพลินและเสียง จะสร้างความบันเทิงให้ผู้เล่น
4.

- **ขอบเขตของโครงการ**

Project Game OOP

วัตถุประสงค์

1. เพื่อนำเนื้อหาที่เรียนมาประยุกต์ใช้
2. เพื่อเพิ่มประสบการณ์ และ ความชำนาญ

บทที่ 2

ส่วนการพัฒนา

- **รายละเอียดเกม**

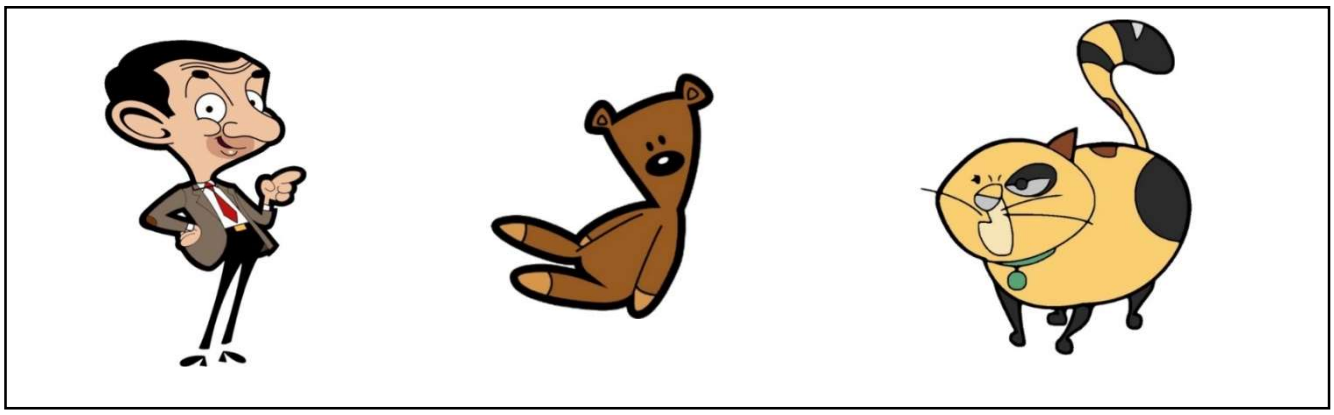
Teddy ของ Mr. Bean ได้หายไปในขณะที่เขากำลังอยู่ในช่วงเทศกาลวันหยุด แมวตัวร้ายหลายตัวกำลังไล่ตาม Mr. Bean เพื่อขัดขวางเขาในการตามหาตุ๊กตา Teddy

- **วิธีการเล่น**

- ผู้เล่นควบคุม Mr. Bean วิ่ง กระโดด และหลบสิ่งกีดขวาง
- Mr. Bean จะต้องวิ่งผ่านหลายด่านพร้อมเก็บ Teddy ที่อยู่ในแต่ละด่าน

- Story Board

ตัวละคร



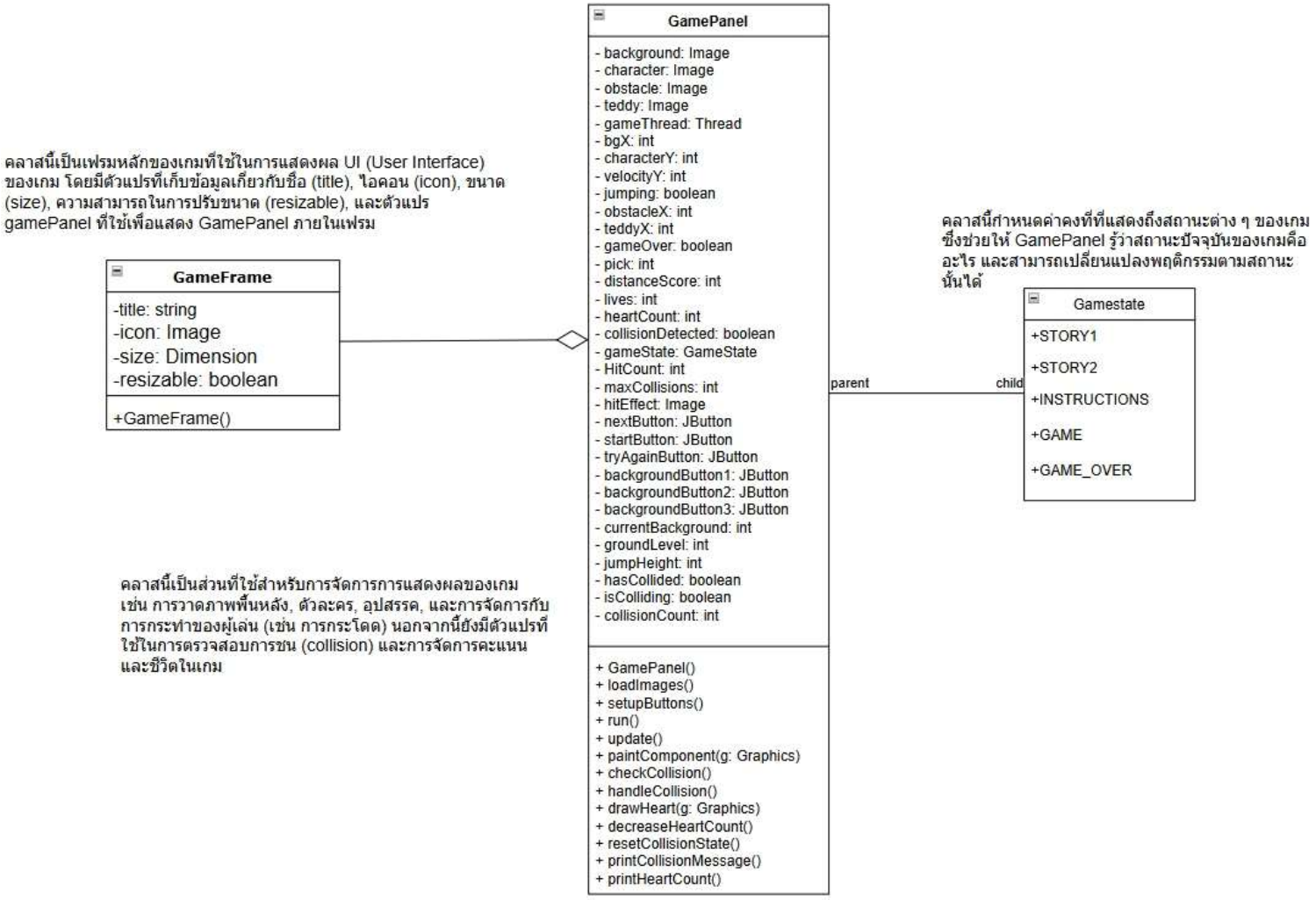
Mr.Bean Teddy Cat

- Background

- ตารางแผนการทำงานเภายใน1ดือน

ลำดับ	งานที่จะทำ	สัปดาห์ที่ 1	สัปดาห์ที่ 2	สัปดาห์ที่3	สัปดาห์ที่4
1	ออกแบบแนวเกม				
2	จัดหาภาพ และข้อมูล				
3	ศึกษาข้อมูล และ พัฒนา				
4	เขียนโปรแกรม				
5	ทดสอบและแก้ไข				

Class diagram



รูปแบบการทำงาน Application / Applet

โปรเจ็คนี้พัฒนาเป็น Application โดยใช้ Java สำหรับการพัฒนา โดยมีการสร้างเกมที่มีกราฟิกและมีการตอบสนองต่อผู้เล่นผ่านอินเทอร์เฟซที่ใช้ Swing (GUI) ในการสร้างหน้าจอเกม

อธิบายส่วนของโปรแกรม

Constructor

public GameFrame() { setTitle("Mr. Bean Find Teddy"); Image icon = new ImageIcon(getClass().getResource("/resources/teddy.png")).getImage(); setIconImage(icon); setSize(800, 600); setResizable(false); GamePanel gamePanel = new GamePanel(); add(gamePanel); }	Constructor นี้จะตั้งชื่อของหน้าต่างเกม ตั้งค่าไอคอน ขนาด และเพิ่ม GamePanel เข้าไปใน GameFrame
public GamePanel() { loadImages(); setLayout(null); setFocusable(true); setupButtons(); gameThread = new Thread(this); gameThread.start(); ... }	Constructor นี้จะโหลดภาพ ตั้งค่าเลย์เอาต์ เริ่มต้นเธรดเกม และตั้งค่าปุ่มต่าง ๆ

Encapsulation

private int bgX = 0; private int characterY = 400; private int velocityY = 0; private boolean jumping = false;	การใช้ private ทำให้ตัวแปรเหล่านี้ไม่สามารถเข้าถึงได้จากภายนอกคลาส GamePanel ซึ่งช่วยป้องกันการเข้าถึงที่ไม่เหมาะสม

Composition

public class GamePanel extends JPanel { private Image backgroundImage; private List<GameObject> gameObjects; public GamePanel() { this.backgroundImage = new ImageIcon("path/to/image.png").getImage(); this.gameObjects = new ArrayList<>(); } public void addGameObject(GameObject obj) { gameObjects.add(obj); } @Override protected void paintComponent(Graphics g) { super.paintComponent(g); g.drawImage(backgroundImage, 0, 0, null); for (GameObject obj : gameObjects) { obj.draw(g); } } }	ในตัวอย่างนี้GamePanelมีbackgroundImageและรายการGameObjectอินสแตนซ์ แสดงองค์ประกอบโดยรวมวัตถุเหล่านี้เป็นส่วนหนึ่งของสถานะ

Polymorphism

@Override protected void paintComponent(Graphics g) { super.paintComponent(g); switch (gameState) { case STORY1, STORY2, INSTRUCTIONS, GAME_OVER -> paintScreens(g); case GAME -> paintGame(g); } }	ความสามารถของเมธอดในการทำงานแตกต่างกันตามประเภทของอ็อบเจ็กต์ที่เรียกใช้ เมธอด paintComponent(Graphics g) มีการใช้ switch เพื่อจัดการกับสถานะของเกม มธอดนี้จะแสดงผลแตกต่างกันตามค่า gameState ซึ่งแสดงถึงการใช้ polymorphism

Abstract

public abstract class GameObject { protected int x, y; public GameObject(int x, int y) { this.x = x; this.y = y; } public abstract void draw(Graphics g); } public class Player extends GameObject { public Player(int x, int y) { super(x, y); } @Override public void draw(Graphics g) { g.fillRect(x, y, 50, 50); // Draw player as a rectangle } }	

Inheritance

public class GamePanel extends JPanel implements Runnable {	GamePanel สืบทอดจาก JPanel และ จะมีคุณสมบัติและเมธอดของ JPanel เช่น การวาดกราฟิกและการจัดการกับเหตุการณ์

- GUI

โดยหน้านี้จะเล่าเรื่องของตัวละคร กดปุ่ม next เพื่อไปประโยคถัดไป

หน้าอธิบายวิธีการเล่น จะมีปุ่ม start กดเพื่อเริ่มเกมส์

หน้าเริ่มต้นจะขึ้นเฉพาะตอน กด run เข้าเกมส์ตอนแรก เท่านั้น

เริ่มต้นเกมส์ เกมส์จะเลื่อนไปเรื่อย

-แถบของหัวใจ3อัน

-แถบจับเวลานับระยะทางที่วิ่ง

-เลขจำนวน teddy

กด spacebar เพื่อกระโดดข้าม

ต้องไล่เก็บ teddy เพื่อ

เมื่อปะทะกับแมวจะเกิด effectชน แล้วหัวใจก็ลด 1 ดวง

เมื่อเกิดการปะทะจนหัวใจหมด จะขึ้นหน้า lose game

จะขึ้นจำนวนระยะทางล่าสุดที่ทำได้

จะขึ้นจำนวนteddy ที่เก็บได้ มีปุ่ม try again เพื่อเริ่มเกมส์ใหม่

เลือก **Button** 3 อัน ด้านบน เพื่อเปลี่ยนฉากพื้นหลัง

Event handling

ActionListener

```
private void setupButtons() {
        //-------->NEXT
        nextButton = new JButton("Next");
        nextButton.setBounds(300, 500, 100, 50);
        nextButton.addActionListener(e -> {
            if (gameState == GameState.STORY1) {
                gameState = GameState.STORY2;
                repaint();
            } else if (gameState == GameState.STORY2) {
                gameState = GameState.INSTRUCTIONS;
                remove(nextButton);
                add(startButton);
                repaint();
            }
        });
        //-------->START
        startButton = new JButton("Start Game");
        startButton.setBounds(300, 500, 120, 50);
        startButton.addActionListener(e -> {
            if (gameState == GameState.INSTRUCTIONS) {
                gameState = GameState.GAME;
                remove(startButton);
                add(backgroundButton1);
                add(backgroundButton2);
                add(backgroundButton3);
                requestFocusInWindow();
                repaint();
            }
```

nextButton:

- ปุ่ม "Next" ที่เมื่อกดจะเปลี่ยนสถานะของเกม จาก STORY1 ไป STORY2 และ จาก STORY2 ไป INSTRUCTIONS โดยจะทำการลบปุ่มและเพิ่มปุ่ม startButton เมื่อถึง INSTRUCTIONS.

startButton:

- ปุ่ม "Start Game" ที่จะเริ่มเกมเมื่อกดในสถานะ INSTRUCTIONS โดยจะลบปุ่มนี้และเพิ่มปุ่มพื้นหลัง (background buttons).

```
});

//-------->TRY AGAIN
tryAgainButton = new JButton("Try Again");
tryAgainButton.setBounds(300, 500, 120, 50);
tryAgainButton.addActionListener(e -> {
    if (gameState == GameState.GAME_OVER) {
        restartGameWithoutStory();
        remove(tryAgainButton);
        add(backgroundButton1);
        add(backgroundButton2);
        add(backgroundButton3);
        requestFocusInWindow();
        repaint();
    }
});

//-------->BACKGROUND BUTTONS
backgroundButton1 = new JButton(resizeIcon(buttonbg, 80, 80));
backgroundButton1.setBounds(500, 10, 80, 80);
backgroundButton1.setBorderPainted(false);
backgroundButton1.setContentAreaFilled(false);
backgroundButton1.setFocusPainted(false);
backgroundButton1.setFocusable(false);
backgroundButton1.addActionListener(e -> {
    currentBackground = 0;
    requestFocusInWindow();
    repaint();
});
backgroundButton2 = new JButton(resizeIcon(buttonbg2, 80, 80));
backgroundButton2.setBounds(600, 10, 80, 80);
backgroundButton2.setBorderPainted(false);
backgroundButton2.setContentAreaFilled(false);
backgroundButton2.setFocusPainted(false);
backgroundButton2.setFocusable(false);
backgroundButton2.addActionListener(e -> {
    currentBackground = 1;
    requestFocusInWindow();
    repaint();
});
backgroundButton3 = new JButton(resizeIcon(buttonbg3, 80, 80));
backgroundButton3.setBounds(700, 10, 80, 80);
backgroundButton3.setBorderPainted(false);
backgroundButton3.setContentAreaFilled(false);
backgroundButton3.setFocusPainted(false);
backgroundButton3.setFocusable(false);
backgroundButton3.addActionListener(e -> {
    currentBackground = 2;
    requestFocusInWindow();
    repaint();
});
```

tryAgainButton:

- ปุ่ม "Try Again" ที่จะเริ่มเกมใหม่เมื่อกดในสถานะ GAME_OVER. จะลบปุ่มนี้และเพิ่มปุ่มพื้นหลังเช่นเดียวกัน.

backgroundButtons:

- ปุ่มที่ใช้เปลี่ยนพื้นหลังของเกม โดยจะมีสามปุ่ม ที่แต่ละปุ่มจะเปลี่ยน
  ค่า currentBackground และทำการรีเฟรชหน้าจอ.

KeyListener

```
        add(nextButton);
        setFocusable(true);

        addKeyListener(new KeyAdapter() {
            @Override
            public void keyPressed(KeyEvent e) {
                if (e.getKeyCode() == KeyEvent.VK_SPACE) {
                    if (gameState == GameState.GAME && !jumping) {
                        jumping = true;
                        velocityY = -20;
                    }
                }
            }
        });
    }
```

ฟังการกดปุ่ม Spacebar ซึ่งจะทำให้ตัวละครกระโดดในระหว่างที่อยู่ในสถานะ GAME และไม่อยู่ในระหว่างการกระโดด (jumping)

อธิบายอัลกอริทึม

```
    private boolean isColliding = false;
        private int collisionCount = 0;

        public void checkCollision() {
            if (isColliding) {

                return;
            }

            if (playerCollidesWithObstacle()) {

                collisionCount++;
                isColliding = true;

                System.out.println("Collision Count: " + collisionCount);
            }

        }
    public void update() {

            if (!playerCollidesWithObstacle()) {
                isColliding = false;
            }
        }


        private boolean checkCollisionWithObstacle() {
```

checkCollision()

- ฟังก์ชันนี้ใช้ในการตรวจสอบการชนกัน (collision) กับอุปสรรค (obstacle).
- ถ้ามีการชนกัน (isColliding เป็น true), จะไม่ทำการตรวจสอบอีก.
- ถ้ามีการชนกันกับอุปสรรค ฟังก์ชันจะเพิ่ม collisionCount ขึ้น 1 และพิมพ์จำนวนการชนกัน.

update()

- ฟังก์ชันนี้จะอัปเดตสถานะการชนกัน โดยถ้าตัวละครไม่ชนกับอุปสรรค (playerCollidesWithObstacle() คืนค่า false), จะรีเซ็ต isColliding เป็น false.

Rectangle characterBounds = new Rectangle(100, characterY, character.getWidth(null), character.getHeight(null)); int obstacleWidth = obstacle.getWidth(null) / 2; int obstacleHeight = obstacle.getHeight(null); Rectangle obstacleBounds = new Rectangle(obstacleX + (obstacle.getWidth(null) / 4), 430, obstacleWidth, obstacleHeight); return characterBounds.intersects(obstacleBounds); }	
private boolean checkCollisionWithObstacle() { Rectangle characterBounds = new Rectangle(100, characterY, character.getWidth(null), character.getHeight(null)); int obstacleWidth = obstacle.getWidth(null) / 2; int obstacleHeight = obstacle.getHeight(null); Rectangle obstacleBounds = new Rectangle(obstacleX + (obstacle.getWidth(null) / 4), 430, obstacleWidth, obstacleHeight); return characterBounds.intersects(obstacleBounds); }	ฟังก์ชันนี้สร้าง Rectangle สำหรับตัวละครและอุปสรรค และตรวจสอบว่ามีการชนกันระหว่างกันหรือไม่ โดยใช้ intersects().
private boolean checkCollisionWithTeddy() { Rectangle characterBounds = new Rectangle(100, characterY, character.getWidth(null), character.getHeight(null)); Rectangle teddyBounds = new Rectangle(teddyX, 350, teddy.getWidth(null), teddy.getHeight(null)); return characterBounds.intersects(teddyBounds); }	ฟังก์ชันนี้สร้าง Rectangle สำหรับสิ่งของตรวจสอบว่ามีการชนกันระหว่างกันหรือไม่ โดยใช้ intersects().
private void handleCollision() { if (checkCollisionWithObstacle() && !hasCollided) { hasCollided = true; decreaseHeartCount(); HitCount++; printCollisionMessage(); printHeartCount(); if (HitCount == maxCollisions || heartCount == 0) { gameState = GameState.GAME_OVER; gameOver = true; remove(backgroundButton1); remove(backgroundButton2); remove(backgroundButton3); add(tryAgainButton); }	ฟังก์ชันนี้จัดการกับการชนกันที่เกิดขึ้น: • ถ้ามีการชนกันกับอุปสรรค, จะลดจำนวนหัวใจ (heartCount) และเพิ่มจำนวนการชน (HitCount). • ถ้าจำนวนการชนถึงค่าที่กำหนด (maxCollisions) หรือหัวใจหมด, เกมจะเข้าสู่สถานะ GAME_OVER. • หากมีการชนกับ teddy, จะเพิ่มคะแนน (pick) และเคลื่อนที่ teddy ออกไปนอกหน้าจอ.

`` new Timer(1000, evt -> resetCollisionState()).start(); } if (checkCollisionWithTeddy()) { pick++; teddyX = -100; } } ```	
`` private void updateGame() { bgX -= 5; if (bgX <= -getWidth()) { bgX = 0; } distanceScore += 1; if (jumping) { if (characterY > jumpHeight) { characterY -= 10; } else { Timer holdTimer = new Timer(500, event -> { jumping = false; ((Timer) event.getSource()).stop(); }); holdTimer.setRepeats(false); holdTimer.start(); } } else if (characterY < groundLevel) { characterY += 10; // Falling down } obstacleX -= 5; if (obstacleX < -50) { obstacleX = getWidth(); } teddyX -= 5; if (teddyX < -64) { teddyX = getWidth() + 200; } } ```	• ฟังก์ชันนี้จะอัปเดตตำแหน่งของพื้นหลัง, ตัวละคร, อุปสรรค , และ teddy. • ถ้าตัวละครกำลังกระโดด, จะลดตำแหน่ง Y ของตัวละคร. ถ้าถึงจุดสูงสุด, จะเริ่ม Timer เพื่อรอให้ตัวละครกลับลง. • ถ้าตัวละครไม่อยู่ในช่วงการกระโดดและตำแหน่ง Y ของตัวละครต่ำกว่า groundLevel, จะทำการเพิ่มตำแหน่ง Y ของตัวละครเพื่อให้มันตกลง. • อัปเดตตำแหน่ง X ของอุปสรรคและ teddy ให้เคลื่อนที่ไปทางซ้าย.

บทที่ 3

สรุป

7. ปัญหาที่พบระหว่างการพัฒนา

- ปัญหาการจัดการงานและเวลา ที่ไม่สมดุล
- ยังหาความรู้ไม่เพียงพอต่อการเขียน
- การคำนวณ logic การชนที่ไม่แม่นยำ
- ปัญหาด้านประสิทธิภาพในการโหลดกราฟิกที่มีขนาดใหญ่ ทำให้เกมหน่วงบางครั้ง
- การจัดการกับกราฟิกและการเคลื่อนที่ของวัตถุในหลายๆ ด่าน

8. จุดเด่นของโปรแกรม

ความสวยงามและองค์ประกอบที่หลากหลาย ภายในเกมโดยการกระโดดและหลบสิ่งกีดขวาง

ระบบการเก็บคะแนนที่แสดงผลในแถบสถานะและระบบสะสมคะแนนเมื่อผ่านด่าน

9. คำแนะนำสำหรับผู้สอนหรือรุ่นต่อไป

สำหรับนักพัฒนารุ่นต่อไป อยากให้เรา ให้ความสำคัญกับการบริหารจัดการเวลาการทำงานให้ดี และ การประเมินขอบเขตของงานให้พอดีกับเวลาและความสามารถค่ะ

----------end----------