Marlon Hernandez, Adam Shehata, Kevyn Girao, Alexis Selgado

The Very Hungry Caterpillar Final Project Report

The Very Hungry Caterpillar Game is a Java Swing arcade-style project. The game features a caterpillar that moves continuously across a 600x600 grid using the arrow keys. The player's goal is to eat randomly spawning apples, grow in length, and increase their score while avoiding collisions with the walls or the caterpillar's own body. When a collision occurs, the game displays a Game Over message along with the final score. A Reset button has been added to allow players to restart the game instantly without relaunching the application.

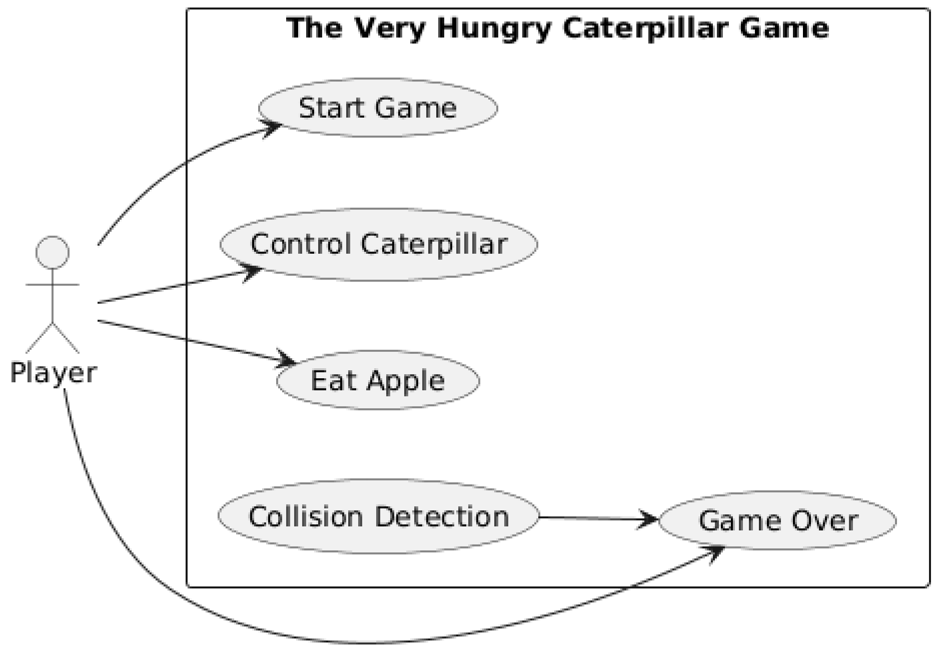
The game uses Java, Swing, and AWT, and includes a GameFrame for the window, GamePanel for all rendering and logic, and a main launcher class. The movement system reads arrow key input via a KeyAdapter, updates the caterpillar's direction, and advances its position every timer tick. Apple generation is fully randomized within the grid bounds, and collision logic checks both wall and self-body contact. Eating an apple increases the caterpillar's body length and increments the applesEaten counter. When the game ends, the timer stops, the caterpillar freezes, and the GamePanel displays both the message and score.

Core features include continuous movement, grid-based rendering, random apple spawning, dynamic body growth, score tracking, collision detection, and a reset mechanism. Use cases covered in the original project specification include starting the game, controlling the caterpillar, eating apples, detecting collisions, and ending the game. The system handles all rendering, movement updates, and event detection, while the player interacts only through the keyboard. Key concepts in the glossary include caterpillar segments, apples, the game board, collisions, score, game loop, UNIT_SIZE, SCREEN_WIDTH/HEIGHT, and all major methods such as startGame, draw, newApple, move, checkApple, checkCollisions, gameOver, and the MyKeyAdapter input handler.

To run the game, clone the repository from GitHub, open it in NetBeans or any Java IDE, and run the CaterpillarGame.java main file. The game window will open automatically, displaying the initial caterpillar and apple, and the player can begin controlling the movement immediately. The project is intended for academic use and demonstrates interactive programming, event handling, graphics rendering, and basic game loop logic using Java Swing.

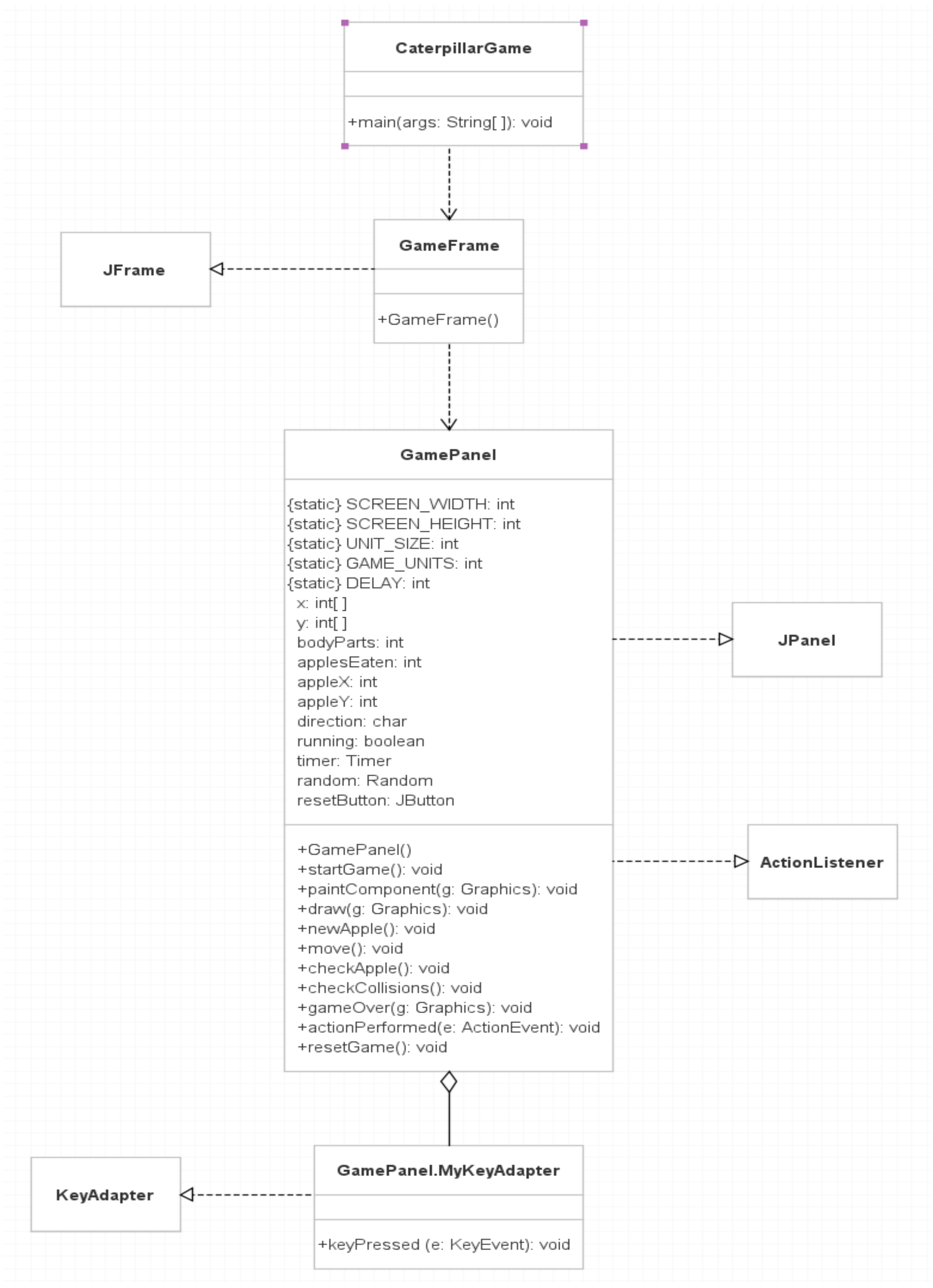
Marlon Hernandez, Adam Shehata, Kevyn Girao, Alexis Selgado

UML Diagrams (1 of 2)



Marlon Hernandez, Adam Shehata, Kevyn Girao, Alexis Selgado

UML Diagrams (2 of 2)



Marlon Hernandez, Adam Shehata, Kevyn Girao, Alexis Selgado

```
package Catepillar;

/**
 * The Very Hungry Caterpillar game
 * Marlon Hernandez, Kevyn Girao, Adam Shehata, Alexis Salgado
 */
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.util.Random;

import javax.swing.JPanel;

public class GamePanel extends JPanel implements ActionListener{
    // All functions needed in order to program, made by Alexis
    static final int SCREEN_WIDTH = 600; // Screen size
    static final int SCREEN_HEIGHT = 600; // Screen size
    static final int UNIT_SIZE = 25; // Caterpillar dimension
    static final int GAME_UNITS = (SCREEN_WIDTH*SCREEN_HEIGHT)/UNIT_SIZE;
    static final int DELAY = 120; // Speed of caterpillar
    final int x[] = new int[GAME_UNITS];
    final int y[] = new int[GAME_UNITS];
    int bodyParts = 6; // Starting caterpillar bodypart size
    int applesEaten; // Point marker
    int appleX; // Location of apples on x axis
    int appleY; // Location of apples on y axis
    char direction = 'R'; // Starting caterpillar direction
    boolean running = false; // Caterpillar moving automatically
    Timer timer;
    Random random;

    JButton resetButton; // << ADDED

    GamePanel(){
        // Background of the game, made by Alexis
        random = new Random();
        this.setPreferredSize(new Dimension(SCREEN_WIDTH, SCREEN_HEIGHT));
        this.setBackground(Color.BLACK);
        this.setFocusable(true);
        this.addKeyListener(new MyKeyAdapter());
        startGame();

        // Reset Button, made by Adam
        resetButton = new JButton("Reset");
        resetButton.setFocusable(false);
        resetButton.setVisible(false);
        resetButton.addActionListener(e -> resetGame());
        this.setLayout(null);
        resetButton.setBounds((SCREEN_WIDTH/2)-50, (SCREEN_HEIGHT/2)+100, 100, 40);
        this.add(resetButton);
    }
```

GamePanel.java Code 1

Marlon Hernandez, Adam Shehata, Kevyn Girao, Alexis Selgado

```
    // Made by Alexis
    public void startGame() {
        // When program starts running, game begins
        newApple();
        running = true;
        timer = new Timer(DELAY,this);
        timer.start();
    }

    // Made by Marlon
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        draw(g);
    }

    // Made by Marlon
    public void draw(Graphics g) {
        // Coloring the apple, catepillar, and score
        if(running) {
            for(int i=0;i<SCREEN_HEIGHT/UNIT_SIZE;i++) {
                g.drawLine(i*UNIT_SIZE, 0, i*UNIT_SIZE,
SCREEN_HEIGHT);
                g.drawLine(0, i*UNIT_SIZE, SCREEN_WIDTH,
i*UNIT_SIZE);
            }
            g.setColor(Color.red);
            g.fillOval(appleX, appleY, UNIT_SIZE,
UNIT_SIZE);

            for(int i =0; i< bodyParts;i++) {
                if(i == 0) {
                    g.setColor(Color.YELLOW);
                    g.fillRect(x[i], y[i], UNIT_SIZE,
UNIT_SIZE);
                }
                else {
                    g.setColor(new Color(255,222,89));
                    g.fillRect(x[i], y[i], UNIT_SIZE,
UNIT_SIZE);
                }
            }
            g.setColor(Color.red);
            g.setFont( new Font("Bookman Old
Style",Font.BOLD, 40));
            FontMetrics metrics =
getFontMetrics(g.getFont());
            g.drawString("Score: "+applesEaten,
(SCREEN_WIDTH - metrics.stringWidth("Score:
"+applesEaten))/2, g.getFont().getSize());
        }
        else {
            gameOver(g);
        }
    }
```

GamePanel.java Code 2

Marlon Hernandez, Adam Shehata, Kevyn Girao, Alexis Selgado

```java
// Made by Kevyn
    public void newApple() {
        // Generating a new apple when game begins
        appleX = random.nextInt((int)(SCREEN_WIDTH/UNIT_SIZE))*UNIT_SIZE;
        appleY = random.nextInt((int)(SCREEN_HEIGHT/UNIT_SIZE))*UNIT_SIZE;

    }

    // Made by Marlon
    public void move() {
        // Controlling the caterpillar
        for(int i = bodyParts;i>0;i--) {
            x[i] = x[i-1];
            y[i] = y[i-1];
        }

        switch(direction) {
            case 'U':
                y[0] = y[0] - UNIT_SIZE;
                break;
            case 'D':
                y[0] = y[0] + UNIT_SIZE;
                break;
            case 'L':
                x[0] = x[0] - UNIT_SIZE;
                break;
            case 'R':
                x[0] = x[0] + UNIT_SIZE;
                break;
        }
    }

    // Made by Kevyn
    public void checkApple() {
        // Checking to see if when catepillar eats an apple:
        // Caterpillar grows and a new apple generates.
        if((x[0] == appleX) && (y[0] == appleY)) {
            bodyParts++;
            applesEaten++;
            newApple();
        }

    }
```

GamePanel.java Code 3

Marlon Hernandez, Adam Shehata, Kevyn Girao, Alexis Selgado

```
   // Made by Kevyn
    public void checkCollisions() {
        // Checking if head collides with body of catepillar or borders of screen
        for(int i = bodyParts;i>0;i--) {
            if((x[0] == x[i])&& (y[0] == y[i])) {
                running = false;
            }
        }
        // checking if head touches left border
        if(x[0] < 0) {
            running = false;
        }
        // checking if head touches right border
        if(x[0] > SCREEN_WIDTH) {
            running = false;
        }
        // checking if head touches top border
        if(y[0] < 0) {
            running = false;
        }
        // checking if head touches bottom border
        if(y[0] > SCREEN_HEIGHT) {
            running = false;
        }

        if(!running) {
            timer.stop();
        }
    }

    // Made by Adam
    public void gameOver(Graphics g) {
        // When caterpiilar hits wall or itself, game over text appears along with score
        g.setColor(Color.red);
        g.setFont( new Font("Bookman Old Style",Font.BOLD, 40));
        FontMetrics metrics1 = getFontMetrics(g.getFont());
        g.drawString("Score: "+applesEaten, (SCREEN_WIDTH - metrics1.stringWidth("Score: 
"+applesEaten))/2, g.getFont().getSize());

        g.setColor(Color.red);
        g.setFont( new Font("Bookman Old Style",Font.BOLD, 75));
        FontMetrics metrics2 = getFontMetrics(g.getFont());
        g.drawString("Game Over", (SCREEN_WIDTH - metrics2.stringWidth("Gamer Over"))/2, 
SCREEN_HEIGHT/2);

        resetButton.setVisible(true); // << ADDED
    }

    @Override
    // Made by Adam
    public void actionPerformed(ActionEvent e) {
        if(running) {
            move();
            checkApple();
            checkCollisions();
        }
        repaint();
    }
```

GamePanel.java Code 4

```
// Reset game logic, made by Adam
    public void resetGame() {
        bodyParts = 6;
        applesEaten = 0;
        direction = 'R';

        // reset all body coordinates
        for(int i=0; i<bodyParts; i++){
            x[i] = 0;
            y[i] = 0;
        }

        newApple();
        running = true;
        resetButton.setVisible(false);
        timer.start();
    }

    public class MyKeyAdapter extends KeyAdapter{
        // Connecting the game with UP DOWN LEFT RIGHT keys
        @Override
        // Made by Marlon
        public void keyPressed(KeyEvent e) {
            switch(e.getKeyCode()) {
                case KeyEvent.VK_LEFT:
                    if(direction != 'R') {
                        direction = 'L';
                    }
                    break;
                case KeyEvent.VK_RIGHT:
                    if(direction != 'L') {
                        direction = 'R';
                    }
                    break;
                case KeyEvent.VK_UP:
                    if(direction != 'D') {
                        direction = 'U';
                    }
                    break;
                case KeyEvent.VK_DOWN:
                    if(direction != 'U') {
                        direction = 'D';
                    }
                    break;
            }
        }
    }
}
```

GamePanel.java Code 5

Marlon Hernandez, Adam Shehata, Kevyn Girao, Alexis Selgado

```
package Catepillar;

/**
 * The Very Hungry Caterpillar game
 * Marlon Hernandez, Kevyn Girao, Adam Shehata, Alexis
Salgado
 */
public class CaterpillarGame {

    public static void main(String[] args) {

        new GameFrame();

        }
    }
```

CaterpillarGame.java Code 1

```
package Catepillar;

/**
 * The Very Hungry Caterpillar game
 * Marlon Hernandez, Kevyn Girao, Adam Shehata, Alexis Salgado
 */
import javax.swing.JFrame;

public class GameFrame extends JFrame{

    GameFrame(){

        this.add(new GamePanel());
        this.setTitle("Caterpillar");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setResizable(false);
        this.pack();
        this.setVisible(true);
        this.setLocationRelativeTo(null);

    }
}
```

GameFrame.java Code 1