**CLASS Game :**

(1) java.awt.image.BufferStrategy: Là một lớp trong gói java.awt.image được sử dụng để hiển thị đồ họa trên một JFrame hoặc một cửa sổ trên hệ thống. Nó được sử dụng để tạo một bộ đệm đồ họa để tránh các vấn đề liên quan đến việc vẽ trực tiếp lên màn hình. Bộ đệm đồ họa cho phép ứng dụng vẽ các hình ảnh trước, sau đó hiển thị chúng một cách liên tục và mượt mà trên màn hình :

Sử dụng java.awt.image.BufferStrategy để quản lý double buffering trong game hoặc ứng dụng đồ họa. Ví dụ:

// Khởi tạo đối tượng BufferStrategy với 3 buffer

canvas.createBufferStrategy(3);

// Lấy đối tượng BufferStrategy từ canvas

BufferStrategy bs = canvas.getBufferStrategy();

// Lấy đối tượng Graphics để vẽ lên buffer

Graphics g = bs.getDrawGraphics();

// Vẽ đối tượng Graphics

g.fillRect(0, 0, canvas.getWidth(), canvas.getHeight());

// Hiển thị buffer lên màn hình

bs.show();

// Giải phóng đối tượng Graphics

g.dispose(); s

(2) java.awt.image.BufferedImage: Là một lớp trong gói java.awt.image được sử dụng để xử lý hình ảnh trong Java. Nó cho phép tạo, đọc và ghi hình ảnh dưới nhiều định dạng khác nhau, bao gồm JPEG, PNG, GIF và BMP. BufferedImage cung cấp các phương thức để lấy và đặt các pixel của một hình ảnh, cũng như các phương thức để thay đổi kích thước và kiểu hình ảnh.

Sử dụng java.awt.image.BufferedImage để tạo một hình ảnh trong Java. Ví dụ:

BufferedImage image = new BufferedImage(100, 100, BufferedImage.TYPE\_INT\_RGB);

Graphics2D g2d = image.createGraphics();

g2d.setColor(Color.BLUE);

g2d.fillRect(0, 0, 100, 100);

(3) java.awt.image.DataBufferInt: Là một lớp trong gói java.awt.image được sử dụng để lưu trữ dữ liệu pixel của hình ảnh dưới dạng mảng số nguyên. Nó cho phép truy cập nhanh chóng đến dữ liệu pixel của một hình ảnh và cho phép xử lý nhanh chóng của nó. Các phương thức của lớp DataBufferInt cho phép truy cập các pixel của hình ảnh theo hàng và cột, cũng như truy cập tất cả các giá trị pixel dưới dạng một mảng 1 chiều

Ví dụ sau sử dụng DataBufferInt để lấy mảng pixels của một hình ảnh BufferedImage:

import java.awt.image.BufferedImage;

import java.awt.image.DataBufferInt;

public class ImageExample {

public static void main(String[] args) {

// Tạo một BufferedImage

BufferedImage image = new BufferedImage(100, 100, BufferedImage.TYPE\_INT\_RGB);

// Lấy mảng pixels của BufferedImage

int[] pixels = ((DataBufferInt) image.getRaster().getDataBuffer()).getData();

// Thay đổi giá trị của một số pixels

for (int i = 0; i < pixels.length; i++) {

pixels[i] = 0xFFFF0000; // Đặt giá trị màu của pixel là đỏ (ARGB)

}

}

}

(4) + Hàm System.nanoTime() trả về một giá trị thời gian hiện tại, tính bằng nanosecond (1/1,000,000,000 giây) kể từ một thời điểm cố định trên hệ thống.

+ Hàm System.currentTimeMillis() trả về một giá trị thời gian hiện tại, tính bằng millisecond (1/1,000 giây) kể từ thời điểm 1/1/1970 00:00:00 GMT. Vì vậy, biến "timer" sẽ lưu trữ thời gian hiện tại tính bằng millisecond.

**CLASS FRAME:**

JFrame.EXIT\_ON\_CLOSE, điều này cho phép ứng dụng kết thúc toàn bộ quá trình của nó khi cửa sổ JFrame được đóng. Nó tương đương với việc gọi phương thức System.exit(0), nghĩa là thoát khỏi chương trình với mã thoát 0 (nghĩa là thoát một cách bình thường).  
  **CLASS KEYBOARD:**

Thư viện java.awt.event.KeyEvent được sử dụng **để định nghĩa** các sự kiện liên quan đến phím trên bàn phím, chẳng hạn như phím được nhấn, phím được giải nén và phím đã được giữ.

Thư viện java.awt.event.KeyListener được sử dụng **để lắng nghe** các sự kiện liên quan đến bàn phím, chẳng hạn như phím được nhấn, phím được giải nén và phím đã được giữ.

**CLASS GAMEPANEL**

(1) setLayout(new BorderLayout()) được sử dụng để thiết lập kiểu bố trí của GamePanel thành BorderLayout. Điều này có nghĩa là các thành phần con của GamePanel sẽ được sắp xếp theo 5 vị trí chính: North, South, East, West và Center. Đối với GamePanel, ta không chỉ định rõ vị trí sẽ sắp xếp các thành phần con trong panel nên chúng sẽ được đặt vào vị trí Center mặc định của BorderLayout . Vùng North, South, East, West nằm ở viền của cửa sổ và thường dùng để đặt các control nhỏ như button, label, ...

(2) setPreferredSize(new Dimension(Game.WIDTH \* Game.SCALE, Game.HEIGHT \* Game.SCALE));, đặt kích thước ưu tiên của GamePanel thành kích thước của cảnh game. Game.WIDTH và Game.HEIGHT là kích thước của cảnh game trong đơn vị ô vuông. Game.SCALE là hệ số tỷ lệ, cho phép người chơi điều chỉnh kích thước hiển thị của game. Với kích thước ưu tiên này, GamePanel sẽ có kích thước đủ lớn để hiển thị cảnh game một cách đầy đủ và rõ ràng.

**CLASS INFOPANEL**

(1) setLayout(new GridLayout()) được sử dụng để thiết lập trình bố cục của các thành phần trên panel bằng GridLayout. GridLayout là một kiểu bố cục cho phép các thành phần được sắp xếp thành lưới có kích thước cố định, trong đó các ô có kích thước bằng nhau.

Ví dụ: Nếu muốn thiết kế một panel chứa 6 nút bấm sắp xếp thành 2 hàng và 3 cột, ta có thể sử dụng mã nguồn sau:

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(2, 3)); // Số 2 và 3 là số hàng và số cột tương ứng

panel.add(new JButton("Button 1"));

panel.add(new JButton("Button 2"));

panel.add(new JButton("Button 3"));

panel.add(new JButton("Button 4"));

panel.add(new JButton("Button 5"));

panel.add(new JButton("Button 6"));

Trong class InfoPanel : setLayout(new GridLayout()) được sử dụng để thiết lập bố cục hiển thị các thành phần timeLabel và pointsLabel trong JPanel theo lưới với một hàng và hai cột.