Java Pixel Extraction and Bitwise Operations

1. Storing Binary and Hexadecimal Values in Java

In Java, numerical values can be represented in different formats, including decimal, binary, and hexadecimal.

Binary Representation:
Java allows binary numbers using the prefix '0b' (since Java 7). Each digit is either '0' or '1'.

```
int binaryNumber = 0b1010; // Represents 10 in decimal
System.out.println(binaryNumber); // Output: 10
```

Hexadecimal Representation:
Hexadecimal numbers are prefixed with '0x' and use digits 0-9 and letters A-F.

```
int hexNumber = 0x1A; // Represents 26 in decimal
System.out.println(hexNumber); // Output: 26
```

2. Why Pixel Values Use Bitwise Right Shift (>>)

Each pixel in an image is stored as a 32-bit integer (4 bytes), with separate 8-bit values for Alpha, Red, Green, and Blue.

To extract these values, we use bitwise shifting (>>) and bit-masking (& 0xFF).

```
int red = (pixel >> 16) & 0xFF;
int green = (pixel >> 8) & 0xFF;
int blue = pixel & 0xFF;
```

3. Java Code for Extracting Pixels in Binary Format

```
import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;

public class ExtractPixelsBinary {
    public static void main(String[] args) {
        try {
            File file = new File("image.jpg");
            BufferedImage image = ImageIO.read(file);

            int width = image.getWidth();
            int height = image.getHeight();

            for (int y = 0; y < height; y++) {
                for (int x = 0; x < width; x++) {
                    int pixel = image.getRGB(x, y);
                    int red = (pixel >> 16) & 0xFF;
                    int green = (pixel >> 8) & 0xFF;
                    int blue = pixel & 0xFF;
```

Java Pixel Extraction and Bitwise Operations

```
                    String redBinary = String.format("%8s", Integer.toBinaryString(red)).replace(" ", "0");
                    String greenBinary = String.format("%8s", Integer.toBinaryString(green)).replace(" ", "0");
                    String blueBinary = String.format("%8s", Integer.toBinaryString(blue)).replace(" ", "0");

                     System.out.println("Pixel at (" + x + ", " + y + "): R=" + redBinary + " G=" + greenBinary
+ " B=" + blueBinary);
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Conclusion

Java allows storing binary (0b) and hexadecimal (0x) values efficiently.
Pixels are stored as 32-bit integers, and RGB components are extracted using bitwise shifting (>>) and bit-masking (& 0xFF).
The provided Java program extracts and displays RGB values in binary format, demonstrating how color components are stored in images.