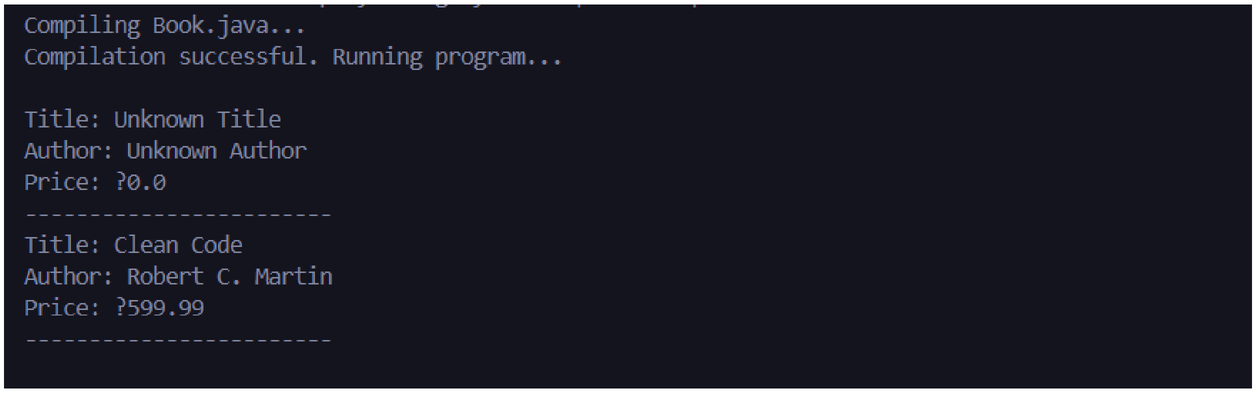
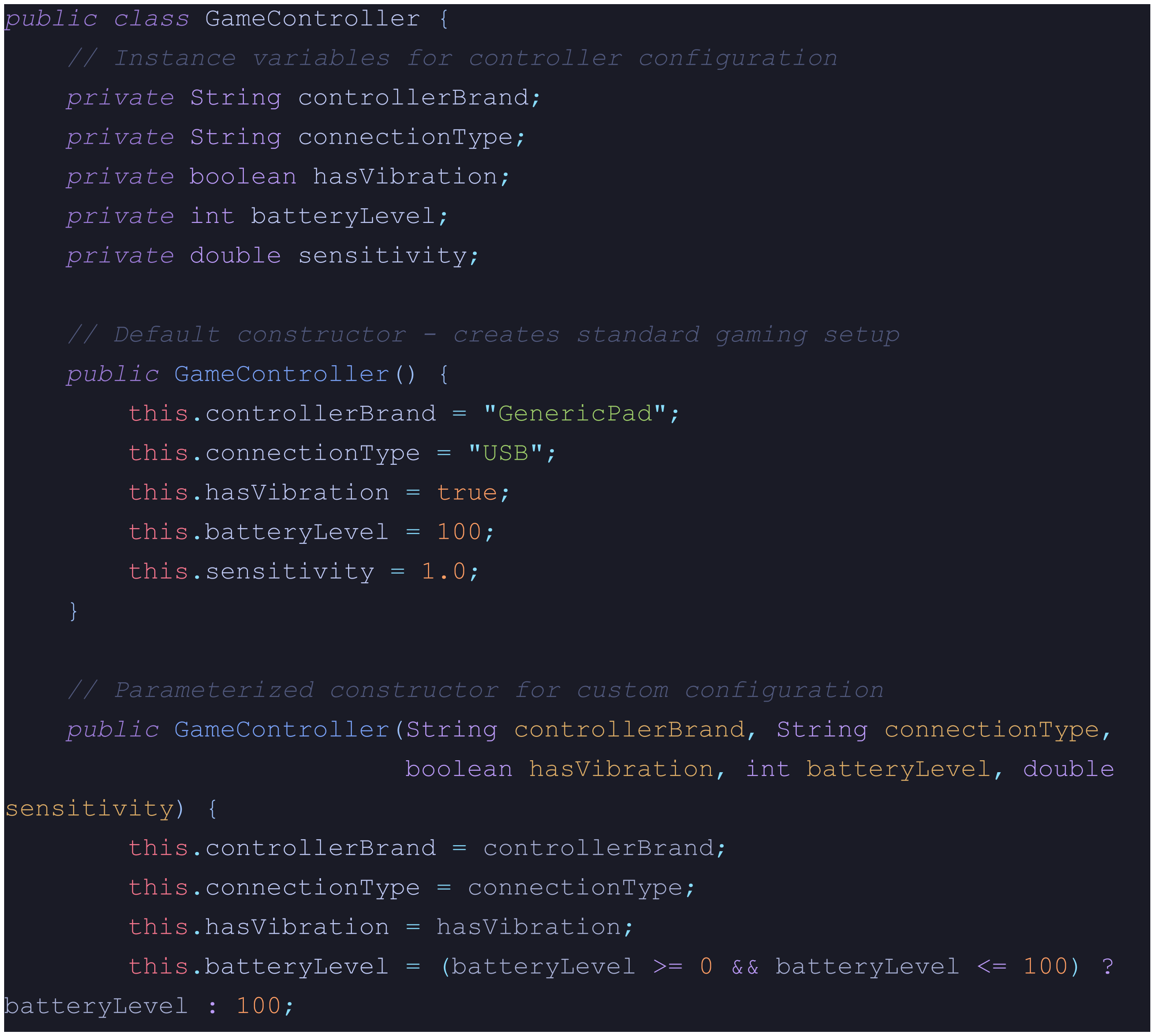
```java
public class Book {
    String title;
    String author;
    double price;

    // Default constructor
    public Book() {
        this.title = "Unknown Title";
        this.author = "Unknown Author";
        this.price = 0.0;
    }

    // Parameterized constructor
    public Book(String title, String author, double price) {
        this.title = title;
        this.author = author;
        this.price = price;
    }

    // Display method
    public void display() {
        System.out.println("Title: " + title);
        System.out.println("Author: " + author);
        System.out.println("Price: ₹" + price);
        System.out.println("------------------------");
    }

    public static void main(String[] args) {
        // Create book1 using default constructor
        Book book1 = new Book();

        // Create book2 using parameterized constructor
        Book book2 = new Book("Clean Code", "Robert C. Martin", 599.99);

        // Display both books
        book1.display();
        book2.display();
    }
}
```

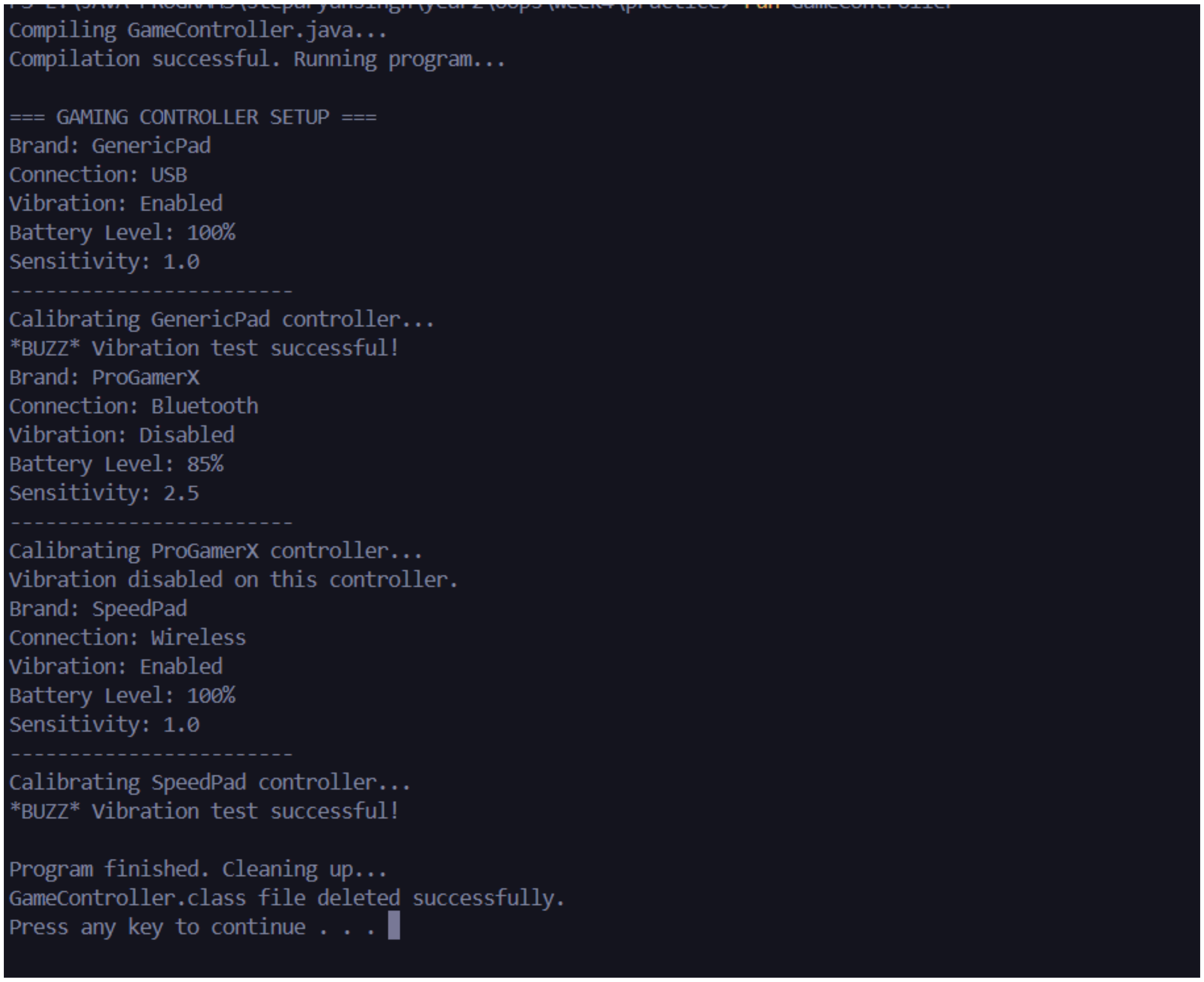


```
Compiling Book.java...
Compilation successful. Running program...

Title: Unknown Title
Author: Unknown Author
Price: ?0.0
------------------------
Title: Clean Code
Author: Robert C. Martin
Price: ?599.99
------------------------
```

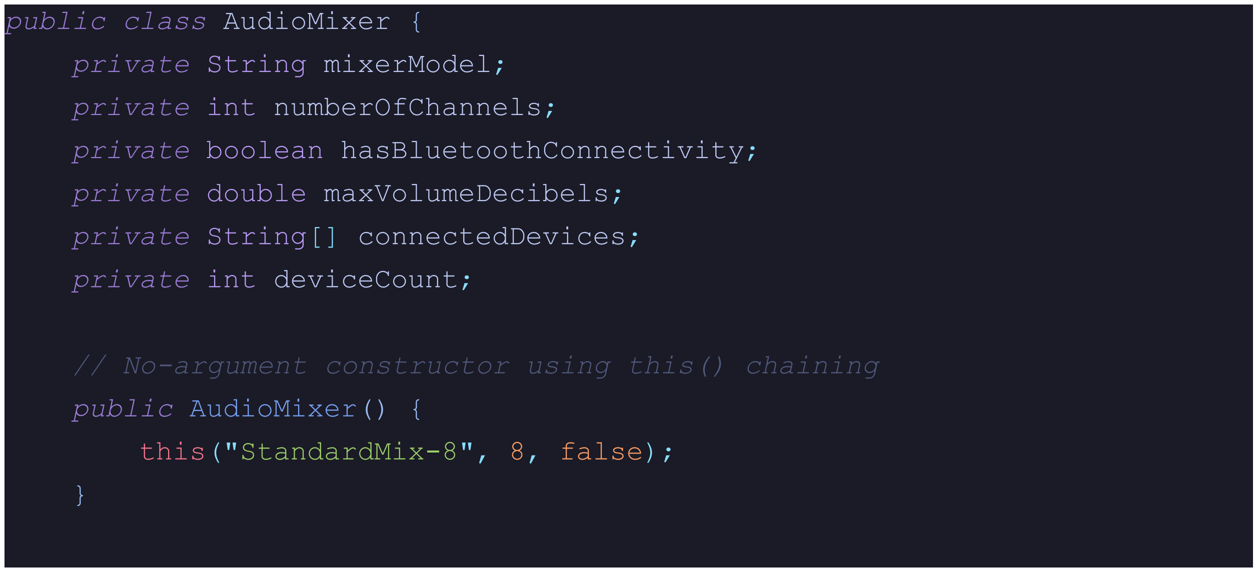


```java
public class GameController {
    // Instance variables for controller configuration
    private String controllerBrand;
    private String connectionType;
    private boolean hasVibration;
    private int batteryLevel;
    private double sensitivity;

    // Default constructor - creates standard gaming setup
    public GameController() {
        this.controllerBrand = "GenericPad";
        this.connectionType = "USB";
        this.hasVibration = true;
        this.batteryLevel = 100;
        this.sensitivity = 1.0;
    }

    // Parameterized constructor for custom configuration
    public GameController(String controllerBrand, String connectionType,
                          boolean hasVibration, int batteryLevel, double
sensitivity) {
        this.controllerBrand = controllerBrand;
        this.connectionType = connectionType;
        this.hasVibration = hasVibration;
        this.batteryLevel = (batteryLevel >= 0 && batteryLevel <= 100) ?
batteryLevel : 100;
```

```java
        this.sensitivity = (sensitivity >= 0.1 && sensitivity <= 3.0) ? sensitivity : 1.0;
    }

    // Two-parameter convenience constructor
    public GameController(String brand, String connectionType) {
        this.controllerBrand = brand;
        this.connectionType = connectionType;
        this.hasVibration = true;
        this.batteryLevel = 100;
        this.sensitivity = 1.0;
    }

    // Method to calibrate controller
    public void calibrateController() {
        System.out.println("Calibrating " + controllerBrand + " controller...");
    }

    // Method to display configuration
    public void displayConfiguration() {
        System.out.println("Brand: " + controllerBrand);
        System.out.println("Connection: " + connectionType);
        System.out.println("Vibration: " + (hasVibration ? "Enabled" : "Disabled"));
        System.out.println("Battery Level: " + batteryLevel + "%");
        System.out.println("Sensitivity: " + sensitivity);
        System.out.println("------------------------");
    }

    // Method to test vibration
    public void testVibration() {
        if (hasVibration) {
            System.out.println("*BUZZ* Vibration test successful!");
        } else {
            System.out.println("Vibration disabled on this controller.");
        }
    }

    // Main method to test all configurations
    public static void main(String[] args) {
        System.out.println("=== GAMING CONTROLLER SETUP ===");

        // Controller with default constructor
        GameController defaultController = new GameController();

        // Controller with full parameterized constructor
        GameController customController = new GameController("ProGamerX",
"Bluetooth", false, 85, 2.5);

        // Controller with convenience constructor
        GameController quickSetupController = new
GameController("SpeedPad", "Wireless");

        // Test all methods
        defaultController.displayConfiguration();
        defaultController.calibrateController();
        defaultController.testVibration();

        customController.displayConfiguration();
        customController.calibrateController();
        customController.testVibration();

        quickSetupController.displayConfiguration();
        quickSetupController.calibrateController();
        quickSetupController.testVibration();
    }
}
```



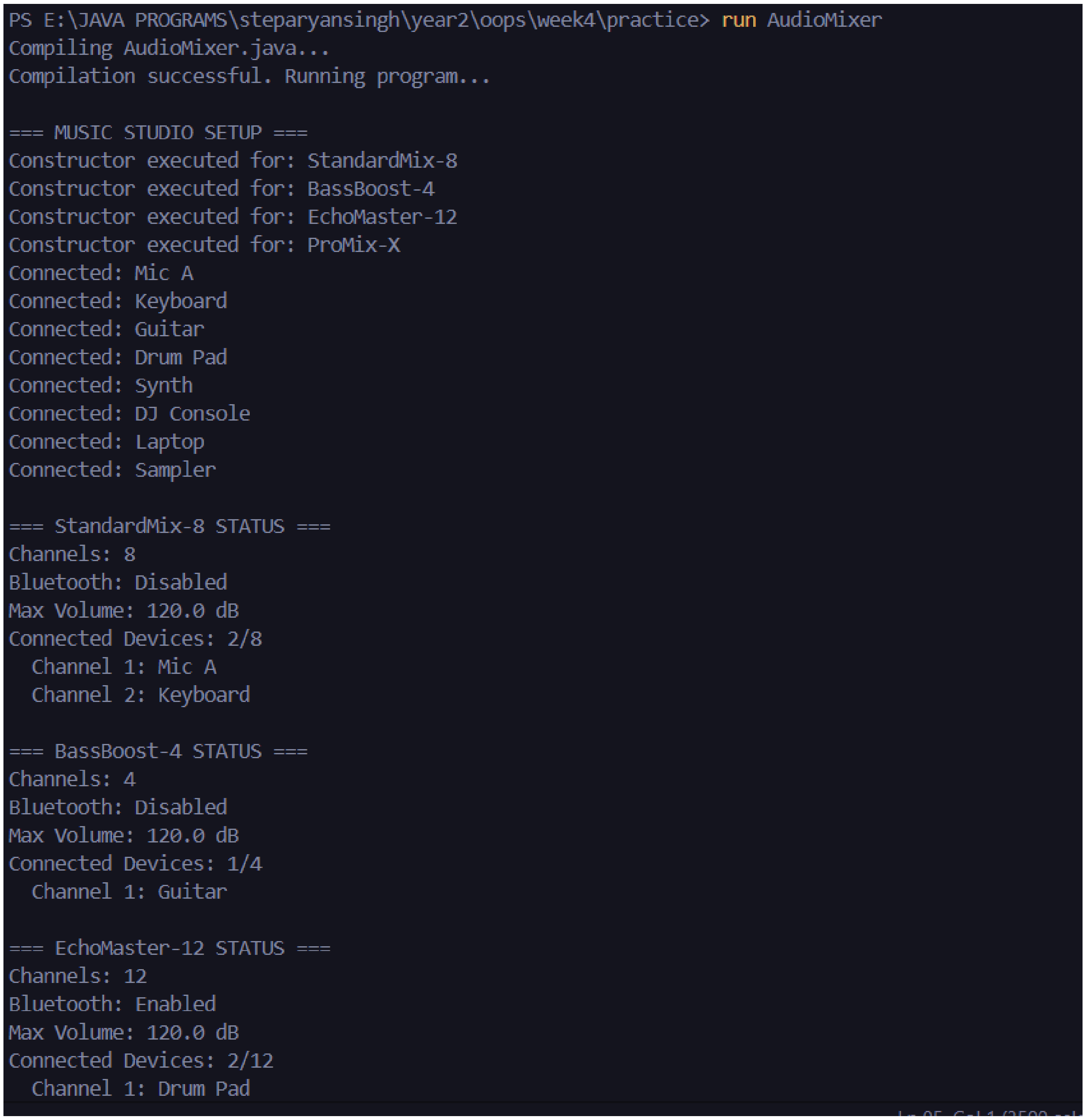
```
Compiling GameController.java...
Compilation successful. Running program...

=== GAMING CONTROLLER SETUP ===
Brand: GenericPad
Connection: USB
Vibration: Enabled
Battery Level: 100%
Sensitivity: 1.0
------------------------
Calibrating GenericPad controller...
*BUZZ* Vibration test successful!
Brand: ProGamerX
Connection: Bluetooth
Vibration: Disabled
Battery Level: 85%
Sensitivity: 2.5
------------------------
Calibrating ProGamerX controller...
Vibration disabled on this controller.
Brand: SpeedPad
Connection: Wireless
Vibration: Enabled
Battery Level: 100%
Sensitivity: 1.0
------------------------
Calibrating SpeedPad controller...
*BUZZ* Vibration test successful!

Program finished. Cleaning up...
GameController.class file deleted successfully.
Press any key to continue . . .
```



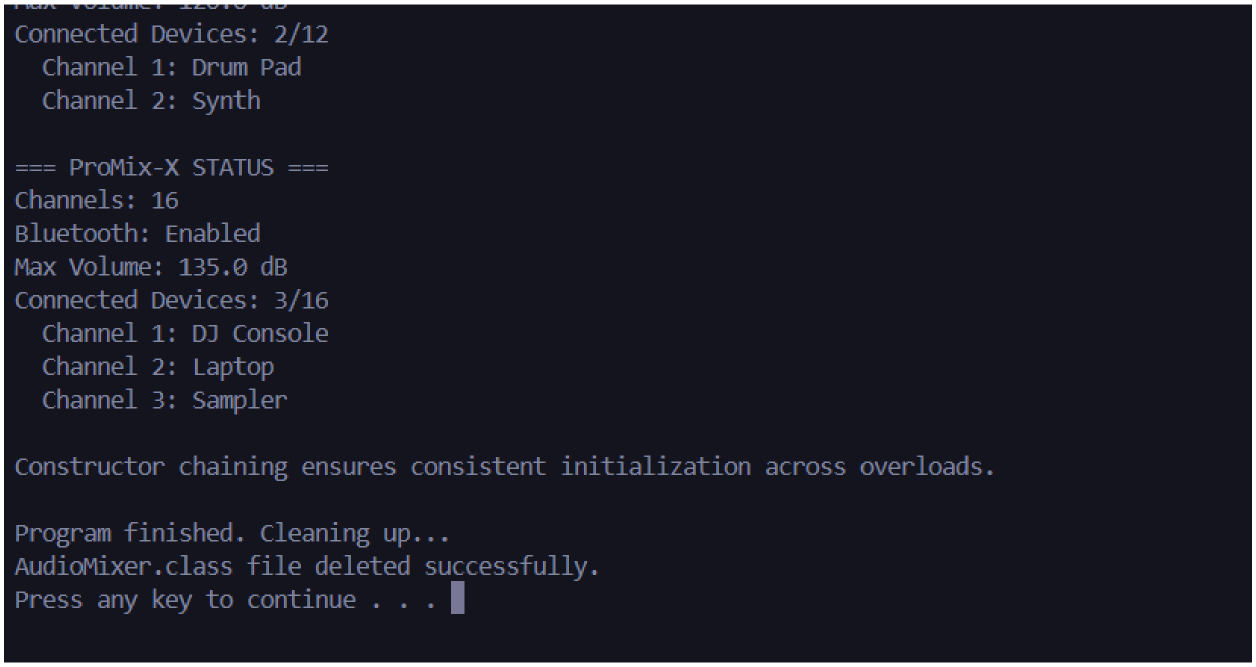
```java
public class AudioMixer {
    private String mixerModel;
    private int numberOfChannels;
    private boolean hasBluetoothConnectivity;
    private double maxVolumeDecibels;
    private String[] connectedDevices;
    private int deviceCount;

    // No-argument constructor using this() chaining
    public AudioMixer() {
        this("StandardMix-8", 8, false);
    }
    // Two-parameter constructor using this() chaining
    public AudioMixer(String mixerModel, int numberOfChannels) {
        this(mixerModel, numberOfChannels, false);
    }

    // Three-parameter constructor using this() chaining
    public AudioMixer(String mixerModel, int numberOfChannels, boolean
hasBluetoothConnectivity) {
        this(mixerModel, numberOfChannels, hasBluetoothConnectivity,
120.0);
    }

    // Main constructor - all parameters
    public AudioMixer(String mixerModel, int numberOfChannels,
                      boolean hasBluetoothConnectivity, double
maxVolumeDecibels) {
        this.mixerModel = mixerModel;
        this.numberOfChannels = numberOfChannels;
        this.hasBluetoothConnectivity = hasBluetoothConnectivity;
        this.maxVolumeDecibels = maxVolumeDecibels;
        this.connectedDevices = new String[numberOfChannels];
        this.deviceCount = 0;
        System.out.println("Constructor executed for: " + mixerModel);
    }

    public void connectDevice(String deviceName) {
        if (deviceCount < connectedDevices.length) {
            connectedDevices[deviceCount] = deviceName;
            deviceCount++;
            System.out.println("Connected: " + deviceName);
        } else {
            System.out.println("All channels occupied!");
        }
    }

    public void displayMixerStatus() {
        System.out.println("\n=== " + mixerModel + " STATUS ===");
        System.out.println("Channels: " + numberOfChannels);
        System.out.println("Bluetooth: " + (hasBluetoothConnectivity ?
"Enabled" : "Disabled"));
```

```
        System.out.println("Max Volume: " + maxVolumeDecibels + " dB");
        System.out.println("Connected Devices: " + deviceCount + "/" +
numberOfChannels);
        for (int i = 0; i < deviceCount; i++) {
            System.out.println("  Channel " + (i + 1) + ": " +
connectedDevices[i]);
        }
    }

    public static void main(String[] args) {
        System.out.println("=== MUSIC STUDIO SETUP ===");

        // Mixer using no-argument constructor
        AudioMixer mixer1 = new AudioMixer();

        // Mixer using two-parameter constructor
        AudioMixer mixer2 = new AudioMixer("BassBoost-4", 4);

        // Mixer using three-parameter constructor
        AudioMixer mixer3 = new AudioMixer("EchoMaster-12", 12, true);

        // Mixer using full constructor
        AudioMixer mixer4 = new AudioMixer("ProMix-X", 16, true, 135.0);

        // Connect devices
        mixer1.connectDevice("Mic A");
        mixer1.connectDevice("Keyboard");

        mixer2.connectDevice("Guitar");

        mixer3.connectDevice("Drum Pad");
        mixer3.connectDevice("Synth");

        mixer4.connectDevice("DJ Console");
        mixer4.connectDevice("Laptop");
        mixer4.connectDevice("Sampler");

        // Display status
        mixer1.displayMixerStatus();
        mixer2.displayMixerStatus();
```

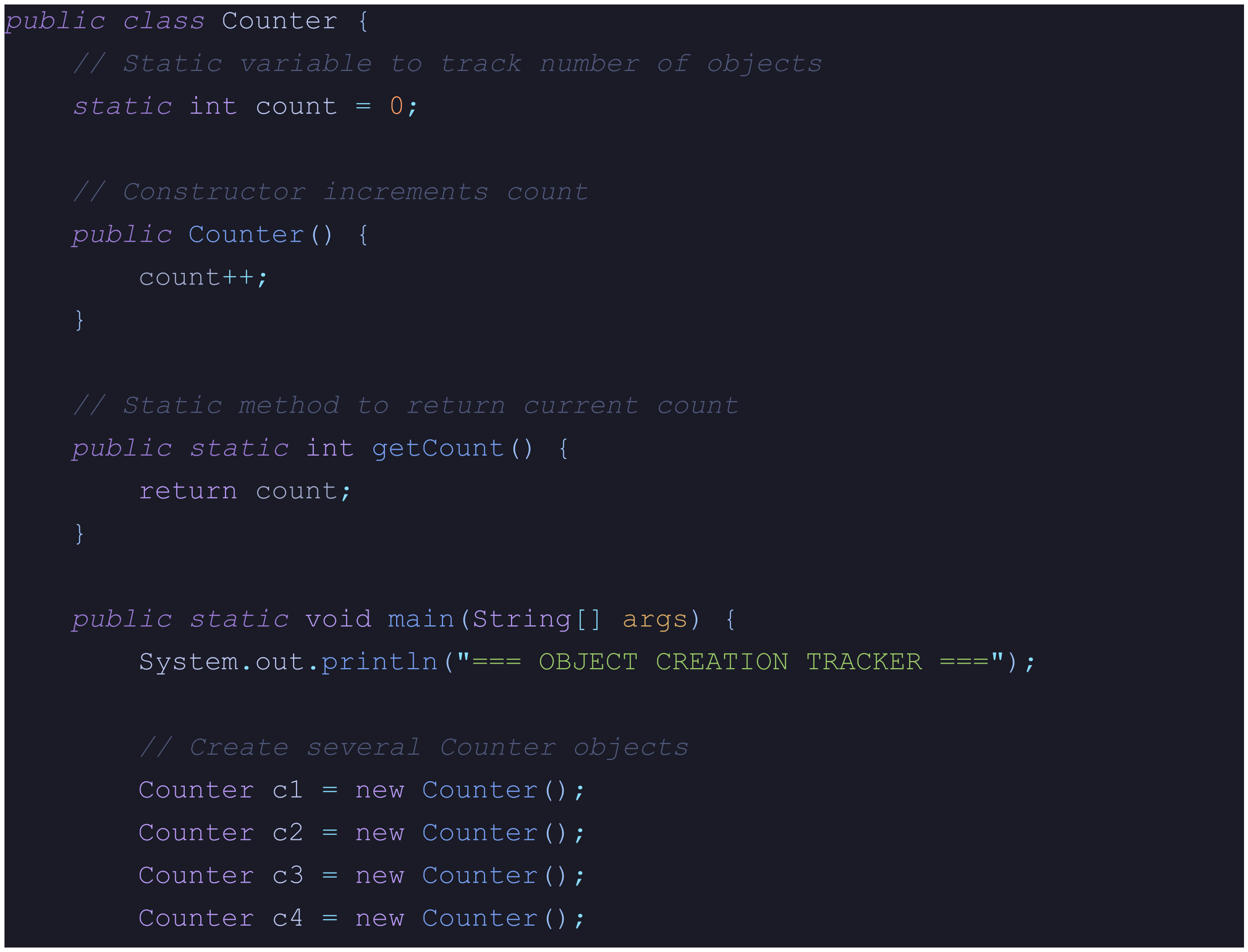
```java
        mixer3.displayMixerStatus();
        mixer4.displayMixerStatus();

        // Comment on constructor chaining
        System.out.println("\nConstructor chaining ensures consistent
initialization across overloads.");
    }
}
```



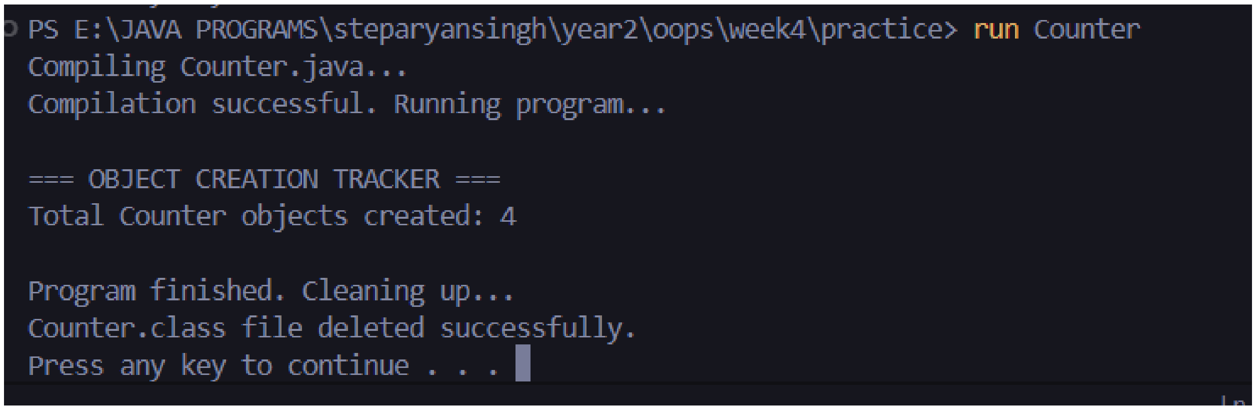
```
PS E:\JAVA PROGRAMS\steparyansingh\year2\oops\week4\practice> run AudioMixer
Compiling AudioMixer.java...
Compilation successful. Running program...

=== MUSIC STUDIO SETUP ===
Constructor executed for: StandardMix-8
Constructor executed for: BassBoost-4
Constructor executed for: EchoMaster-12
Constructor executed for: ProMix-X
Connected: Mic A
Connected: Keyboard
Connected: Guitar
Connected: Drum Pad
Connected: Synth
Connected: DJ Console
Connected: Laptop
Connected: Sampler

=== StandardMix-8 STATUS ===
Channels: 8
Bluetooth: Disabled
Max Volume: 120.0 dB
Connected Devices: 2/8
  Channel 1: Mic A
  Channel 2: Keyboard

=== BassBoost-4 STATUS ===
Channels: 4
Bluetooth: Disabled
Max Volume: 120.0 dB
Connected Devices: 1/4
  Channel 1: Guitar

=== EchoMaster-12 STATUS ===
Channels: 12
Bluetooth: Enabled
Max Volume: 120.0 dB
Connected Devices: 2/12
  Channel 1: Drum Pad
```



```
Connected Devices: 2/12
  Channel 1: Drum Pad
  Channel 2: Synth

=== ProMix-X STATUS ===
Channels: 16
Bluetooth: Enabled
Max Volume: 135.0 dB
Connected Devices: 3/16
  Channel 1: DJ Console
  Channel 2: Laptop
  Channel 3: Sampler

Constructor chaining ensures consistent initialization across overloads.

Program finished. Cleaning up...
AudioMixer.class file deleted successfully.
Press any key to continue . . .
```



```java
public class Counter {
    // Static variable to track number of objects
    static int count = 0;

    // Constructor increments count
    public Counter() {
        count++;
    }

    // Static method to return current count
    public static int getCount() {
        return count;
    }

    public static void main(String[] args) {
        System.out.println("=== OBJECT CREATION TRACKER ===");

        // Create several Counter objects
        Counter c1 = new Counter();
        Counter c2 = new Counter();
        Counter c3 = new Counter();
        Counter c4 = new Counter();
        // Display number of objects created
        System.out.println("Total Counter objects created: " +
Counter.getCount());
    }
}
```



```
PS E:\JAVA PROGRAMS\stepary ansingh\year2\oops\week4\practice> run Counter
Compiling Counter.java...
Compilation successful. Running program...

=== OBJECT CREATION TRACKER ===
Total Counter objects created: 4

Program finished. Cleaning up...
Counter.class file deleted successfully.
Press any key to continue . . .
```