Report: Implementation of Visitor Pattern for Musical Instruments

1. Introduction

The goal of this project was to implement the **Visitor design pattern** in Java, applying **Clean Code principles**, and demonstrate its use with musical instruments.
The project includes two instruments: **Dombyra** and **Guitar**, and three types of visitors:

- **SoundVisitor** – produces the sound of the instrument.
- **ActionVisitor** – shows the action or way to play the instrument.
- **MusicVisitor** – simulates playing music.

The Visitor pattern allows adding new operations to existing object structures without modifying them, which is ideal for adding new actions or behaviors to instruments.

---

2. Design

Classes and Interfaces

1. **Instruments (Interface)**
    - Method: accept(Visitor visitor)
    - Implemented by Dombyra and Guitar.
2. **Visitor (Interface)**
    - Methods: visitDombyra(Dombyra dombyra) and visitGuitar(Guitar guitar)
    - Implemented by SoundVisitor, ActionVisitor, MusicVisitor.
3. **SoundVisitor**
    - Prints the sound produced by the instrument.
    - Plays .wav files using SoundPlayer class.
4. **ActionVisitor**
    - Prints the action for playing each instrument.
5. **MusicVisitor**
    - Prints that the instrument is playing music.
6. **SoundPlayer**
    - A helper class for playing .wav files using Java's javax.sound.sampled API.
    - Ensures sounds play correctly for both instruments.

Clean Code Principles Applied

- **Single Responsibility:** Each class has one responsibility (instrument, visitor, or sound player).
- **Readable Names:** Clear class and method names (SoundVisitor, accept, visitDombyra).
- **Separation of Concerns:** Playing sound is handled in SoundPlayer, not inside Visitor.
- **Extensibility:** Adding new instruments or visitors is easy without modifying existing code.

---

3. Implementation Example

```
Dombyra dombyra = new Dombyra();

Guitar guitar = new Guitar();


Visitor soundVisitor = new SoundVisitor();

Visitor actionVisitor = new ActionVisitor();

Visitor musicVisitor = new MusicVisitor();


dombyra.accept(soundVisitor);

guitar.accept(soundVisitor);

dombyra.accept(actionVisitor);

guitar.accept(actionVisitor);

dombyra.accept(musicVisitor);

guitar.accept(musicVisitor);
```

Output example:

```
SoundVisitor produces the Dombyra sound

SoundVisitor produces the Guitar sound

ActionVisitor shows how to play the Dombyra

ActionVisitor shows how to play the Guitar

MusicVisitor is playing the Dombyra

MusicVisitor is playing the Guitar
```

- If sound files are present, SoundVisitor plays actual instrument sounds.

---

4. Challenges and Solutions

- **Problem:** Java's standard Clip class cannot play .opus files.
- **Solution:** Converted .opus to .wav files and placed them in resources/sounds.
- **Problem:** File paths were not found during runtime.
- **Solution:** Used ClassLoader to load resources reliably from src/main/resources.

---

5. Conclusion

- The project demonstrates the **Visitor design pattern** with musical instruments.
- New instruments or visitors can be added without modifying existing code.
- Clean Code principles were applied to make the project readable, maintainable, and extendable.
- Real sound playback was implemented, enhancing the demonstration.