**CST8288 Design Patterns Worksheet: Exploring Common Design Patterns in Java**

**Name: Bo Qu**   
**Date: Sept. 18, 2024**

**Objective:**

Understand and apply common design patterns used in software development using Java. This worksheet will focus on patterns such as Singleton, Factory, Observer, and Command, along with introducing voxel concepts for game development.

**Part 1: Design Pattern Identification**

**Instructions:**  
For each description below, identify the correct design pattern and provide a Java-based solution.

**1. Singleton Pattern**

**Description:**  
You need to ensure that only one instance of a class is created and provide a global point of access to that instance.

* **Question:**  
  Explain how to implement the Singleton pattern in Java. Write code to demonstrate it with a DatabaseConnection class.

public class DatabaseConnection {

// Your solution here

}

**2. Factory Pattern**

**Description:**  
You want to create objects without specifying the exact class of the object that will be created. Think of a scenario where you need to generate different types of shapes (Circle, Square, Rectangle) based on user input.

* **Question:**  
  Write a Java Factory Pattern example that creates instances of different shapes based on a string parameter.

interface Shape {

void draw();

}

class Circle implements Shape {

public void draw() {

System.out.println("Drawing a Circle");

}

}

class Square implements Shape {

public void draw() {

System.out.println("Drawing a Square");

}

}

class ShapeFactory {

// Your solution here

}

**Part 2: Applying Design Patterns in a Game Development Context (Voxels)**

**Voxel Concept Introduction:**  
A **voxel** is a volumetric pixel, used in 3D game development to represent objects in a grid-based format. Imagine building a Minecraft-like game where objects are made of cubes (voxels).

**3. Command Pattern (Voxels Game Example)**

**Description:**  
In your voxel-based game, you want to allow players to issue commands such as "place voxel," "remove voxel," and "move voxel" to change the game world. Use the **Command Pattern** to achieve this.

* **Question:**  
  Design a system using the Command Pattern for controlling voxel operations in a 3D world. Write a Command interface and concrete commands to place and remove voxels.

interface Command {

void execute();

}

class Voxel {

private String type;

public Voxel(String type) { this.type = type; }

public void place() { System.out.println("Placing voxel: " + type); }

public void remove() { System.out.println("Removing voxel: " + type); }

}

class PlaceVoxelCommand implements Command {

private Voxel voxel;

public PlaceVoxelCommand(Voxel voxel) { this.voxel = voxel; }

public void execute() { voxel.place(); }

}

class RemoveVoxelCommand implements Command {

private Voxel voxel;

public RemoveVoxelCommand(Voxel voxel) { this.voxel = voxel; }

public void execute() { voxel.remove(); }

}

// Your solution here

**4. Observer Pattern (Game Event System)**

**Description:**  
In the same voxel game, you want to notify various systems (e.g., HUD, audio manager) whenever a voxel is placed or removed. Implement an event system using the **Observer Pattern**.

* **Question:**  
  Create an Observer Pattern where multiple systems listen to voxel changes and respond accordingly.

interface Observer {

void update(String eventType, Voxel voxel);

}

class AudioManager implements Observer {

public void update(String eventType, Voxel voxel) {

System.out.println("Audio Manager: Playing sound for " + eventType + " of voxel: " + voxel);

}

}

class HUD implements Observer {

public void update(String eventType, Voxel voxel) {

System.out.println("HUD: Updating display for " + eventType + " of voxel: " + voxel);

}

}

class VoxelSubject {

// Your solution here

}

**Part 3: Short Answer Questions**

1. **Decorator Pattern:**  
   Describe how you would use the **Decorator Pattern** to add different behaviors to a Player object in a game, such as giving the player the ability to fly or swim.
2. **Strategy Pattern:**  
   Imagine you are developing an AI system for different types of enemies in your voxel game. Explain how the **Strategy Pattern** can help you design AI behaviors like "aggressive," "defensive," and "passive."

**Part 4: Additional Challenges**

1. **Extend the Command Pattern**:  
   In the voxel game, create additional commands for rotating and scaling voxels. Implement and describe the steps.
2. **Prototype Pattern (Bonus)**:  
   Explain how the **Prototype Pattern** would be useful when creating multiple instances of complex voxel structures (e.g., buildings). Provide an example using Java.