**CODE**

// Interface for location allocation

interface LocationAllocator {

void allocateLocation(User user);

}

// Class representing a User

class User {

private String name;

private String preferredLocation;

public User(String name, String preferredLocation) {

this.name = name;

this.preferredLocation = preferredLocation;

}

public String getName() {

return name;

}

public String getPreferredLocation() {

return preferredLocation;

}

}

// Class for allocating locations based on user preferences

class PreferredLocationAllocator implements LocationAllocator {

@Override

public void allocateLocation(User user) {

System.out.println("Allocating preferred location for " + user.getName() + ": " + user.getPreferredLocation());

}

}

// Class for allocating a default location if no preference is set

class DefaultLocationAllocator implements LocationAllocator {

@Override

public void allocateLocation(User user) {

System.out.println("Allocating default location for " + user.getName() + ": Default Location");

}

}

// Class to manage the allocation process

class LocationManager {

private LocationAllocator allocator;

public LocationManager(LocationAllocator allocator) {

this.allocator = allocator;

}

public void setAllocator(LocationAllocator allocator) {

this.allocator = allocator;

}

public void allocate(User user) {

allocator.allocateLocation(user);

}

}

// Main class to demonstrate the allocation

public class Main {

public static void main(String[] args) {

User user1 = new User("Alice", "Downtown");

User user2 = new User("Bob", null);

LocationManager manager = new LocationManager(new PreferredLocationAllocator());

manager.allocate(user1);

// Switch to default allocator for user2

manager.setAllocator(new DefaultLocationAllocator());

manager.allocate(user2);

}

}

**Explanation of the Code**

1. **Single Responsibility Principle (SRP)**: Each class has a specific responsibility. User holds user information, PreferredLocationAllocator allocates based on preferences, and DefaultLocationAllocator allocates a default location.
2. **Open/Closed Principle (OCP)**: The LocationAllocator interface allows for new allocation strategies to be added without modifying existing code. You can create new classes that implement LocationAllocator without changing the LocationManager.
3. **Liskov Substitution Principle (LSP)**: Any class that implements LocationAllocator can be used interchangeably in LocationManager.
4. **Interface Segregation Principle (ISP)**: The LocationAllocator interface is focused on a single responsibility (location allocation), ensuring that clients only implement methods they need.
5. **Dependency Inversion Principle (DIP)**: LocationManager depends on the abstraction LocationAllocator, not on concrete implementations. This allows for easy swapping of allocation strategies.

This structure provides a flexible and maintainable design for a location allocation system, following SOLID principles.