Servant-Pattern

A Mini Project for Design-Patterns(14CS413) course by:

1PI14CS081 - Ramadas S Mahale,

1PI14CS081 - Sunil Pai G,

7th semester,CSE.

GitHub: Sunil-Pai-G/Servant-Pattern

OVERVIEW

Servant pattern is a design pattern which is used to provide common functionality to a set of classes. This pattern is used when it does not make sense to provide this functionality in a common parent class and when disparate classes need to show the same behavior.Hence, this pattern prevents duplication of code across multiple classes.The class in which the common behavior is implemented is called the servant class. The classes needing the service from the service class must implement an interface. This interface specifies the capabilities needed by classes in order to be serviced by the servant.

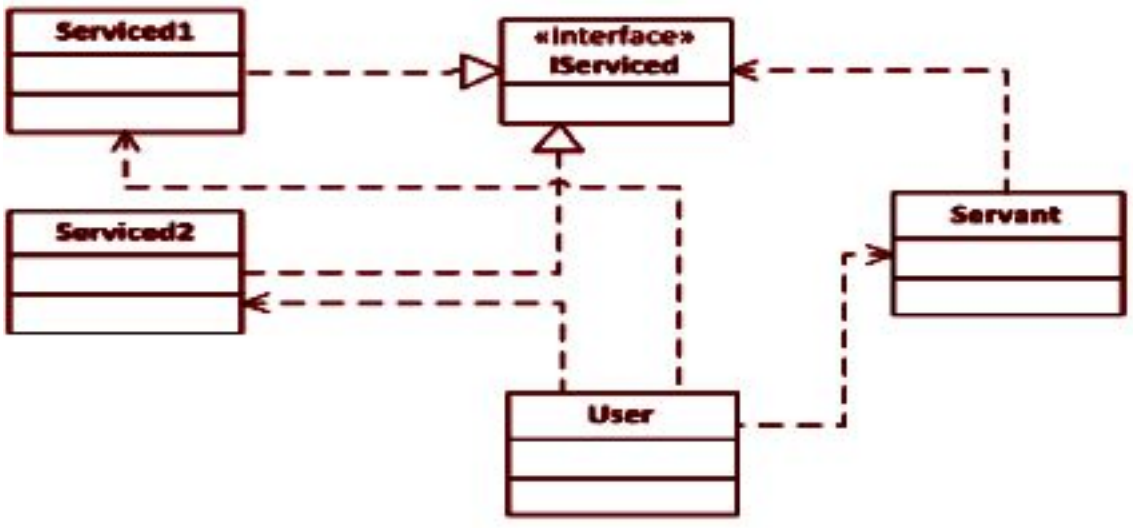
IMPLEMENTATION DETAILS

(Note: The implementations are also available in C++.Please check the CPP folder on the github link.)

There are two ways to implement the pattern:-

1. The client knows about the servant and passes the instances of the class to be serviced to the servant. The servant then calls appropriate methods on the instance passed.

UML diagram:



Code:

```
public class MoveServant {

    public void moveTo(Movable serviced, Position where) {
        serviced.setPosition(where);
    }

    public void moveTo(Movable serviced, int dx, int dy)  {
        serviced.setPosition(new Position(dx,dy));
    }

    public void moveBy(Movable serviced, int dx, int dy) {
        dx += serviced.getPosition().xPosition;
        dy += serviced.getPosition().yPosition;
        serviced.setPosition(new Position(dx, dy));
    }

    public void moveBy(Movable serviced, Position where) {
        where.xPosition += serviced.getPosition().xPosition;
        where.yPosition += serviced.getPosition().yPosition;
        serviced.setPosition(new Position(where.xPosition, where.yPosition));
    }
}
```

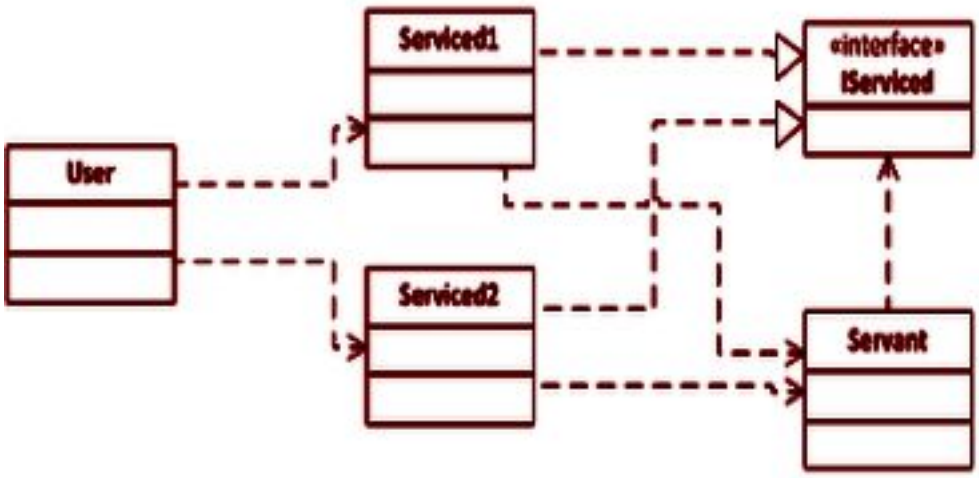
```
public class Rectangle implements Movable {
    private Position p;

    public void setPosition(Position p) {
        this.p = p;
    }

    public Position getPosition() {
        return this.p;
    }
}
```

```
public class ServantClient
{
    public static void main(String args[])
    {

        Triangle triangle = new Triangle();
        Ellipse ellipse = new Ellipse();
        MoveServant moveservant = new MoveServant();

        moveservant.moveTo(triangle,new Position(10,10));

        System.out.println("New Position : "+triangle.getPosition());

    }
}
```

Fig:Servant,serviced class and client

2. The client knows nothing about the servant.When the client needs a certain service he calls a function in the serviced class which in turn calls the appropriate function in the servant.

   UML diagram:



   Code:

```
public class ServantClient
{
    public static void main(String args[])
    {
        Triangle triangle = new Triangle();
        Ellipse ellipse = new Ellipse();

        triangle.moveTo(new Position(10,10));

        System.out.println("New Position : "+triangle.getPosition());

    }
}
```

```
public class Rectangle implements Movable {
    // Position of the geometric object on some canvas
    private Position p;
    private MoveServant m;
    public Rectangle()
    {
        m = new MoveServant();
    }

    // Method, which sets position of geometric object
    public void setPosition(Position p) {
        this.p = p;
    }


    // Method, which returns position of geometric object
    public Position getPosition() {
        return this.p;
    }

    public void moveTo(int dx, int dy)
    {
        m.moveTo(this,dx,dy);
    }
    public void moveTo(Position p)
    {
        m.moveTo(this,p);
    }
    public void moveBy(Position where)
    {
        m.moveBy(this,where);
    }
    public void moveBy(int dx, int dy)
    {
        m.moveBy(this,dx,dy);
    }
}
```

Fig:The client and serviced classes

THREAD SAFE SERVANT PATTERN

The above implementations of the command pattern are not thread safe. Consider what happens when two threads are executing the moveBy function simultaneously on the same object as shown below.Thread 1 wants to move the position of the triangle by (10,10) and thread 2 by (20,20).In effect, the triangle should have an eventual displacement of (30,30). However, if one of the threads reads the position of the triangle before it is updated by the other thread, then there is a race condition. We try to simulate this by making the thread sleep before it updates the position of the triangle.(Note that this doesn't ensure that the race condition will occur every time the code executes).

```
public void moveBy(Movable serviced, Position where) {
    // this is the place to offer the functionality
    where.xPosition += serviced.getPosition().xPosition;
    where.yPosition += serviced.getPosition().yPosition;

    try{
        Thread.sleep(2000);
    }
    catch(InterruptedException e){
        System.out.println(e);
    }

    serviced.setPosition(new Position(where.xPosition, where.yPosition));
}
```

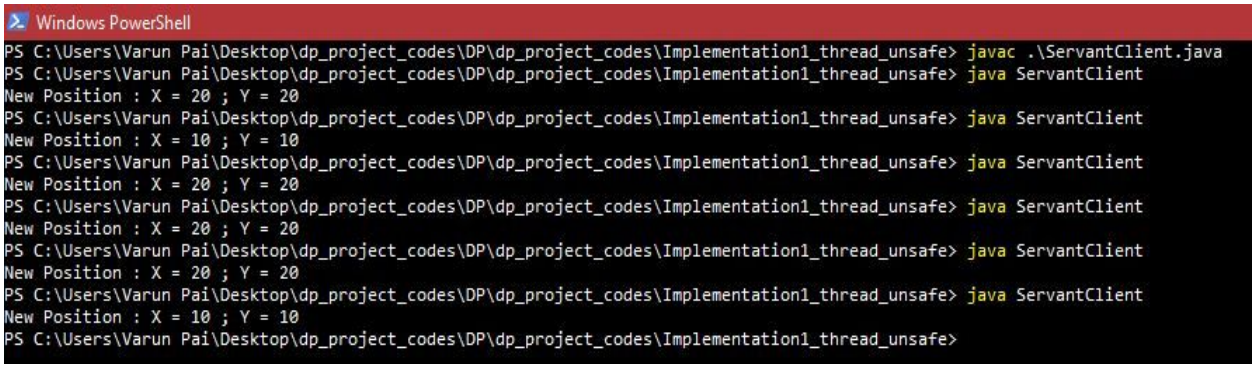


Fig:Thread unsafe servant and its output

The above problem can be resolved by allowing only one thread to access the moveBy function at any instance of time. This can be achieved in JAVA using the synchronized keyword in the method signature and in C++ using a std::mutex and a std::scoped_lock. The implementation of this is as follows.

```java
public void moveBy(Movable serviced, Position where) {

    synchronized(serviced){
        // this is the place to offer the functionality
        where.xPosition += serviced.getPosition().xPosition;
        where.yPosition += serviced.getPosition().yPosition;

        try{
            Thread.sleep(2000);
        }
        catch(InterruptedException e){
            System.out.println(e);
        }

        serviced.setPosition(new Position(where.xPosition, where.yPosition));
    }
}
```

```
Windows PowerShell
PS C:\Users\Varun Pai\Desktop\dp_project_codes\DP\dp_project_codes\Implementation1_thread_safe> javac .\ServantClient.java
PS C:\Users\Varun Pai\Desktop\dp_project_codes\DP\dp_project_codes\Implementation1_thread_safe> java ServantClient
New Position : X = 30 ; Y = 30
PS C:\Users\Varun Pai\Desktop\dp_project_codes\DP\dp_project_codes\Implementation1_thread_safe> java ServantClient
New Position : X = 30 ; Y = 30
PS C:\Users\Varun Pai\Desktop\dp_project_codes\DP\dp_project_codes\Implementation1_thread_safe> java ServantClient
New Position : X = 30 ; Y = 30
PS C:\Users\Varun Pai\Desktop\dp_project_codes\DP\dp_project_codes\Implementation1_thread_safe> java ServantClient
New Position : X = 30 ; Y = 30
PS C:\Users\Varun Pai\Desktop\dp_project_codes\DP\dp_project_codes\Implementation1_thread_safe> java ServantClient
New Position : X = 30 ; Y = 30
PS C:\Users\Varun Pai\Desktop\dp_project_codes\DP\dp_project_codes\Implementation1_thread_safe>
```

Fig:Thread safe implementation of servant pattern and its output

SERVANT PATTERN A SINGLETON?

As the servant class has no attributes of its own and only performs operations on serviced classes, the servant pattern can be made a singleton class. This approach is effective in saving memory. Making the servant a singleton doesn’t affect concurrency in any way as the singleton servant can still be operated by multiple threads.

COMMAND vs SERVANT PATTERN

Though the command and the servant patterns seem similar, there are subtle differences between them.In servant pattern, there is only one class that implements the required functionality(the servant class).In the command pattern however, it is necessary for the command class to know the receiver of the action. Hence, for every receiver (the ellipse,the triangle and rectangle classes) there is a need for a separate command class. In command class there is control on when to execute the commands.In servant there is no control on when commands are executed.

REFERENCES:

- https://en.wikipedia.org/wiki/Servant_(design_pattern)
- http://en.cppreference.com
- https://theboostcpplibraries.com/
- https://softwareengineering.stackexchange.com/questions/299520/do-we-need-servant-pattern-what-about-implementing-in-parent-class
- https://stackoverflow.com/questions/31986332/visitor-vs-servant-vs-command-patterns
- https://stackoverflow.com/questions/27939046/design-patterns-for-php-visitor-pattern-vs-servant-pattern
- https://www.codeproject.com/Articles/598695/Cplusplus-threads-locks-and-condition-variables