The front controller design pattern is used to provide a centralized request handling mechanism so that all requests will be handled by a single handler. This handler can do the authentication/authorization/logging or tracking of request and then pass the requests to corresponding handlers. Following are the entities of this type of design pattern.

- **Front Controller** - Single handler for all kinds of requests coming to the application either web-based/desktop-based.

- **Dispatcher** - Front Controller may use a dispatcher object which can dispatch the request to corresponding specific handler.

- **View** - Views are the object for which the requests are made.

**Implementation**

We are going to create a *FrontController* and *Dispatcher* to act as Front Controller and Dispatcher correspondingly. *HomeView* and *StudentView* represent various views for which requests can come to front controller.

*FrontControllerPatternDemo*, our demo class, will use *FrontController* to demonstrate Front Controller Design Pattern.

```java
public class FrontControllerPatternDemo {
    public static void main(String[] args) {
        HomeView homeView = new HomeView();
        StudentView studentView = new StudentView();
        DispatchRequest(request, homeView);
        DispatchRequest(request, studentView);
    }

    private static void DispatchRequest(String request, Object view) {
        // Implement request dispatching logic
    }
}
```

**Step 1**

Create Views.

*HomeView.java*

```java
public class HomeView {
    public void show()
    { 
        System.out.println("Displaying Home Page");
    }
}
```

*StudentView.java*

```java
public class StudentView {
    public void show()
    { 
        System.out.println("Displaying Student Page");
    }
}
```
Step 2
Create Dispatcher.

Dispatcher.java

```java
public class Dispatcher {
    private StudentView studentView;
    private HomeView homeView;

    public Dispatcher(){
        studentView = new StudentView();
        homeView = new HomeView();
    }

    public void dispatch(String request){
        if(request.equalsIgnoreCase("STUDENT")){
            studentView.show();
        } else{
            homeView.show();
        }
    }
}
```

Step 3
Create FrontController

FrontController.java

```java
public class FrontController {
    private Dispatcher dispatcher;

    public FrontController(){
        dispatcher = new Dispatcher();
    }

    private boolean isAuthenticUser(){
        System.out.println("User is authenticated successfully.");
        return true;
    }

    private void trackRequest(String request){
        System.out.println("Page requested: "+ request);
    }

    public void dispatchRequest(String request){
        //log each request
        trackRequest(request);
        //authenticate the user
        if(isAuthenticUser()){
            dispatcher.dispatch(request);
        }
    }
}
```

Step 4
Use the FrontController to demonstrate Front Controller Design Pattern.

FrontControllerPatternDemo.java

```java
public class FrontControllerPatternDemo {
    public static void main(String[] args) {
```
Step 5

Verify the output.

Page requested: HOME
User is authenticated successfully.
Displaying Home Page
Page requested: STUDENT
User is authenticated successfully.
Displaying Student Page

Loading [MathJax]/jax/output/HTML-CSS/fonts/TeX/fontdata.js