Command Pattern: Assignment-2

Course ID: SWE-4502

Course Name: Design Pattern Lab

Name: Farzana Tabassum

ID: 180042119

Department: CSE (SWE)

Semester: Winter (5th semester)

Date: 13/08/2021

GitLink: https://github.com/farzana-snigdha/dpatterna/tree/command-pattern

Code implementation

command-lightChanges.ts

Page.svelte | TS command-provider.ts | TS command-lightChange.ts 1 | TS command-lightChange.test.ts

dpatterna > src > patterns > command > TS command-lightChange.ts > ...

```ts
// Light class act as a request
export class Light {
  public on(): string {
    return "on";
  }
  public off(): string {
    return "off";
  }

  increase: number = 0;
  redOn(): string {
    this.increase = 0;
    return "red0";
  }

  //increase red light intensity on each click, after reaching maximum intesity,
  //it does not change even if the intensity is increased
  increaseRedLight(): string {
    if (this.increase < 3) {
      this.increase++;
      return `red${this.increase}`;
    } else {
      return `red3`;
    }
  }

  //decrease red light intensity on each click, after reaching minimum intesity,
  //it turned of the light
  decreaseRedLight(): string {
    this.increase--;
    if (this.increase < 0) {
      return "off";
    } else {
      return `red${this.increase}`;
    }
  }
}
```

```
    }
  }
}

//Command interface encapsulates a request by binding together a set of actions
//on a specific receiver. It does so by exposing just one method setCommand()
// that causes some actions to be invoked on the receiver.
export interface Command {
  setCommand(): string;
}

//Light classe's corresponding command classes
export class TurnOnLightCommand implements Command {
  light: Light;

  // The constructor is passed the light it is going to control.
  constructor(light: Light) {
    this.light = light;
  }
  setCommand(): string {
    return this.light.on();
  }
}

export class TurnOffLightCommand implements Command {
  light: Light;

  constructor(light: Light) {
    this.light = light;
  }
  setCommand(): string {
    return this.light.off();
  }
}

```

```
68  }
69
70  //RedLight classe's corresponding command classes
71  export class TurnOnRedLightCommand implements Command {
72    light: Light;
73    constructor(light: Light) {
74      this.light = light;
75    }
76    setCommand(): string {
77      return this.light.redOn();
78    }
79  }
80
81  export class IncreaseRedLightCommand implements Command {
82    light: Light;
83    constructor(light: Light) {
84      this.light = light;
85    }
86    setCommand(): string {
87      return this.light.increaseRedLight();
88    }
89  }
90
91  export class DecreaseRedLightCommand implements Command {
92    light: Light;
93    constructor(light: Light) {
94      this.light = light;
95    }
96    setCommand(): string {
97      return this.light.decreaseRedLight();
98    }
99  }
100
101 //invoker class
102 export class RemoteControl {
```

```
100
101   //invoker class
102   export class RemoteControl {
103     command: Command;
104
105     // set the command the remote will execute
106     execute(command: Command) {
107       this.command = command;
108     }
109
110     public executeCommand() {
111       return this.command.setCommand();
112     }
113   }
114
```

command-provider.ts

```ts
//this class uses RemoteControl class to demonstrate command pattern
import {
  Command,
  DecreaseRedLightCommand,
  IncreaseRedLightCommand,
  Light,

  RemoteControl,
  TurnOffLightCommand,
  TurnOnLightCommand,
  TurnOnRedLightCommand,
} from "patterns/command/command-lightChange";

let light;
let redLight;
let turnOnRedLight: Command;
let turnOnLight: Command;
let turnOffLight: Command;
let remoteControl: RemoteControl;
let increaseRedLight: Command;
let decreaseRedLight: Command;

light = new Light();

turnOnRedLight = new TurnOnRedLightCommand(light);
turnOnLight = new TurnOnLightCommand(light);
turnOffLight = new TurnOffLightCommand(light);
increaseRedLight = new IncreaseRedLightCommand(light);
decreaseRedLight = new DecreaseRedLightCommand(light);

var isRedLightOn = false;

//Use the RemoteControl class to take and execute commands.
remoteControl = new RemoteControl();

//setLightCommands() is called in Page.svelte and there it takes 2 values
//to determine action type. here we can change command dynamically. it calls
```

```
35
36  //setLightCommands() is called in Page.svelte and there it takes 2 values
37  //to determine action type. here we can change command dynamically. it calls
38  //execute() from RemoteControl class and executes user commands and
39  // returns light value using executeCommand()
40  export function setLightCommands(command: string, count: number) {
41    console.log(" count>0  ", count);
42    if (command == "on") {
43      isRedLightOn = false;
44      remoteControl.execute(turnOnLight);
45    } else if (command == "off") {
46      isRedLightOn = false;
47      remoteControl.execute(turnOffLight);
48    } else if (command == "redLight") {
49      isRedLightOn = true;
50      remoteControl.execute(turnOnRedLight);
51      console.log("provider ", command);
52    } else if (command == "increase" && count > 0) {
53      remoteControl.execute(increaseRedLight);
54    } else if (command == "decrease" && count > 0) {
55      remoteControl.execute(decreaseRedLight);
56    }
57
58    return remoteControl.executeCommand();
59  }
60
```

page.svelte

dpatterna > src > pages > hello-command > Page.svelte > script

```svelte
<script>
  import { setLightCommands } from "./command-provider";

  let commands = {
    on: "on",
    off: "off",
    red0: "red/0",
    red1: "red/1",
    red2: "red/2",
    red3: "red/3",
  };
  let commandType = "off";
  let src = `./images/light-receiver/${commands[commandType]}.png`;

  //set intensity for +/- buttons. initially 0; means red light is turned off. after
  // clicking red button i.e. turn on red light, count value will
  // increase and then +/- will increase/decrease red light intensity accordingly
  let count = 0;

  //after clicking on a particular button, this function will execute.
  //it takes 2 params: 1st on for the type of action user wants to see
  //and the 2nd one is count value to check whether red light is turned on or not
  function executeCommand(command, counts) {
    console.log("src ", command);

    //commandType determines which type of light should be visible to the user
    commandType = setLightCommands(command, counts);
    src = `./images/light-receiver/${commands[commandType]}.png`;
    console.log(commandType);
  }

</script>

<h1>Command buttons</h1>

```

```
<h1>Command buttons</h1>

<div class="btn-group">
  <button class="on" on:click={() => executeCommand('on', count)}>On</button>
  <button class="off" on:click={() => executeCommand('off', count)}>Off</button>

  <button
    class="increase-lum"
    on:click={() => executeCommand('increase', count)}
  >+</button>
  <button
    class="decrease-lum"
    on:click={() => executeCommand('decrease', count)}
  >-</button>

  <button
    class="red-light"
    on:click={() => executeCommand('redLight', count++)}
  >Red</button>
</div>

<div class="portrait"><img {src} alt={src} /></div>

<style>
  .btn-group button {
    padding: 10px 24px;
    cursor: pointer;
    width: 20%;
    display: block;
  }

  .btn-group button:not(:last-child) {
    border-bottom: none; /* Prevent double borders */
  }
```

```
66
67    /* Add a background color on hover */
68    .btn-group button:hover {
69      filter: brightness(85%);
70    }
71
72    .decrease-lum {
73      background-color: #a9c1c9;
74    }
75    .red-light {
76      background-color: #f44336;
77    }
78    .increase-lum {
79      background-color: #8bb19b;
80    }
81    .on {
82      background-color: #e7e7e7;
83      color: black;
84    }
85    .off {
86      background-color: #555555;
87      color: white;
88    }
89    img {
90      max-width: 100%;
91      max-height: 100%;
92    }
93    .portrait {
94      height: 300px;
95      width: 500px;
96    }
97
98  </style>
99
```

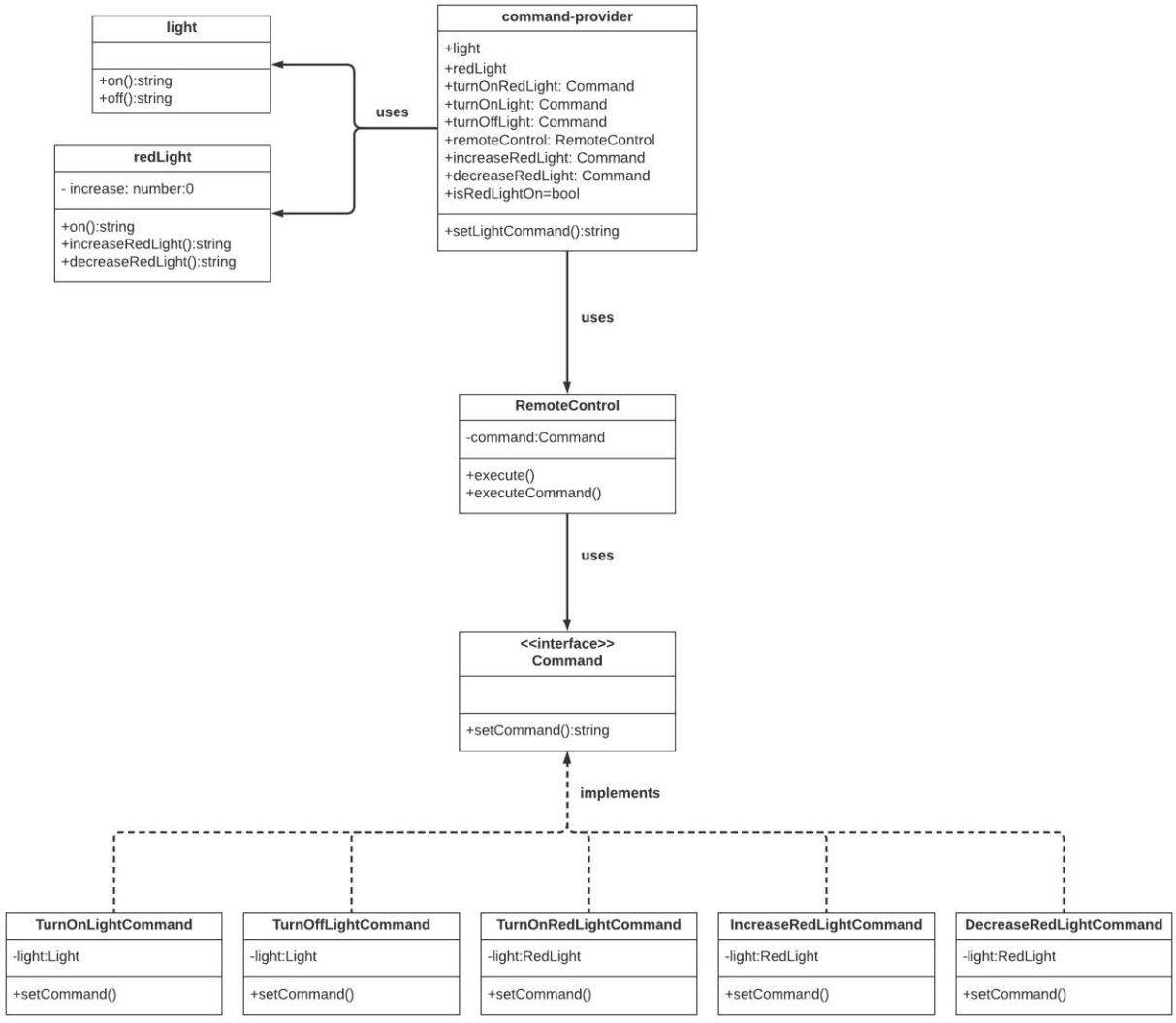
Unit tests

dpatterna > src > __tests__ > TS command-lightChange.test.ts > describe("light command pattern") callback >

```ts
import { setLightCommands } from "pages/hello-command/command-provider";
import {
  Light,

  TurnOnLightCommand,
  TurnOffLightCommand,
  TurnOnRedLightCommand,
  IncreaseRedLightCommand,
  DecreaseRedLightCommand,
} from "patterns/command/command-lightChange";

 | | Run | Debug
describe("light command pattern", () => {
  let light = new Light();

  let count = 0;

  | | Run | Debug
  test("light on", () => {
    let expectation = setLightCommands("on", count);
    let reality = new TurnOnLightCommand(light);
    expect(expectation).toEqual(reality.setCommand());
  });

  | | Run | Debug
  test("light off", () => {
    let expectation = setLightCommands("off", count);
    let reality = new TurnOffLightCommand(light);
    expect(expectation).toEqual(reality.setCommand());
  });

  | | Run | Debug
  test("red light on", () => {
    let expectation = setLightCommands("redLight", count);
    let reality = new TurnOnRedLightCommand(light);
    expect(expectation).toEqual(reality.setCommand());
  });
```

```
    | | Run | Debug
29  test("red light on", () => {
30    let expectation = setLightCommands("redLight", count);
31    let reality = new TurnOnRedLightCommand(light);
32    expect(expectation).toEqual(reality.setCommand());
33  });
34
    | | Run | Debug
35  test("increase red light", () => {
36    count++;
37    let expectation = setLightCommands("increase", count);
38    let reality = new IncreaseRedLightCommand(light);
39    expect(expectation).toEqual(reality.setCommand());
40  });
41
    | | Run | Debug
42  test("decrease red light", () => {
43    count++;
44    let expectation = setLightCommands("decrease", count);
45    let reality = new DecreaseRedLightCommand(light);
46    expect(expectation).toEqual(reality.setCommand());
47  });
48 });
49
```

UML diagram


light
+on():string
+off():string
redLight
- increase: number:0
+on():string
+increaseRedLight():string
+decreaseRedLight():string
uses
command-provider
+light
+redLight
+turnOnRedLight: Command
+turnOnLight: Command
+turnOffLight: Command
+remoteControl: RemoteControl
+increaseRedLight: Command
+decreaseRedLight: Command
+isRedLightOn=bool
+setLightCommand():string
uses
RemoteControl
-command:Command
+execute()
+executeCommand()
uses
<<interface>>
Command
+setCommand():string
implements
TurnOnLightCommand
-light:Light
+setCommand()
TurnOffLightCommand
-light:Light
+setCommand()
TurnOnRedLightCommand
-light:RedLight
+setCommand()
IncreaseRedLightCommand
-light:RedLight
+setCommand()
DecreaseRedLightCommand
-light:RedLight
+setCommand()