// 13

Q.1 Write a Java Program to implement an Adapter design pattern in mobile charger. Define two classes – Volt (to measure volts) and Socket (producing constant volts of 120V). Build an adapter that can produce 3 volts, 12 volts and default 120 volts.

Implements Adapter pattern using Class Adapter

```
class Volt {
  private int volts;

  public Volt(int volts) {
    this.volts = volts;
  }

  public int getVolts() {
    return volts;
  }
}

class Socket {
  public Volt getVolts() {
    return new Volt(120);
  }
}

interface MobileChargerAdapter {
  Volt get3Volts();
```

```
  Volt get12Volts();
  Volt getDefaultVolts();
}

class ClassMobileChargerAdapter extends Socket implements MobileChargerAdapter {
  private Volt convertVolts(Volt volts, int divisor) {
    return new Volt(volts.getVolts() / divisor);
  }

  @Override
  public Volt get3Volts() {
    return convertVolts(getVolts(), 40);
  }

  @Override
  public Volt get12Volts() {
    return convertVolts(getVolts(), 10);
  }

  @Override
  public Volt getDefaultVolts() {
    return getVolts();
  }
}

public class AdapterPatternExample {
  public static void main(String[] args) {
    MobileChargerAdapter chargerAdapter = new ClassMobileChargerAdapter();

    Volt volts3 = chargerAdapter.get3Volts();
    Volt volts12 = chargerAdapter.get12Volts();
    Volt volts120 = chargerAdapter.getDefaultVolts(); // Use the new method

    System.out.println("3 Volts: " + volts3.getVolts() + "V");
    System.out.println("12 Volts: " + volts12.getVolts() + "V");
    System.out.println("Default 120 Volts: " + volts120.getVolts() + "V");
  }
}
```

Q.2. Write a Python program to prepare Scatter Plot for Iris Dataset

```
import matplotlib.pyplot as plt
import pandas as pd

# Load Iris dataset from sklearn
from sklearn.datasets import load_iris
iris_data = load_iris()
iris_df = pd.DataFrame(data=iris_data.data, columns=iris_data.feature_names)
iris_df['species'] = iris_data.target_names[iris_data.target]

# Create a scatter plot
plt.figure(figsize=(8, 6))

colors = {'setosa': 'red', 'versicolor': 'green', 'virginica': 'blue'}
for species, color in colors.items():
  species_data = iris_df[iris_df['species'] == species]
  plt.scatter(species_data['sepal length (cm)'], species_data['sepal width (cm)'],
          label=species, color=color, edgecolors='black', s=50)

# Set plot labels and title
plt.title("Scatter Plot for Iris Dataset")
plt.xlabel("Sepal Length (cm)")
plt.ylabel("Sepal Width (cm)")

# Show legend
plt.legend()

# Show the plot
plt.show()
```

```
# import numpy as np
# import pandas as pd
# import matplotlib.pyplot as plt
# iris = pd.read_csv("Iris.csv") # Reading the dataset “Iris.csv”.
# print (iris.head(10)) # head() will display the top rows of the dataset, the default value of this function is 5,
# #that is it will show top 5 rows when no argument is given to it.
# plt.plot(iris.sepal_length, iris["sepal_length"],"r--")
# plt.show
# plt.show()
# # will display the current figure that you are working on
# iris.plot(kind ="scatter", x ='sepal_length', y ='petal_length')
# plt.grid() # grid () function to add grid lines to the plot
```

// Iris.csv extra

```
sepal_length,sepal_width,petal_length,petal_width,species
5.1,3.5,1.4,0.2,setosa
4.9,3.0,1.4,0.2,setosa
4.7,3.2,1.3,0.2,setosa
4.6,3.1,1.5,0.2,setosa
5.0,3.6,1.4,0.2,setosa
5.4,3.9,1.7,0.4,setosa
```

Using node js create a User Login System.

```
const express = require('express');
const bodyParser = require('body-parser');
const bcrypt = require('bcrypt');
const session = require('express-session');

const app = express();
const PORT = process.env.PORT || 3000;

// Middleware
app.use(bodyParser.urlencoded({ extended: true }));
app.use(session({ secret: 'your-secret-key', resave: true, saveUninitialized: true }));

// Dummy user data (replace this with a database in a real application)
const users = [
 { id: 1, username: 'user', password: 'password' } // Password: password
   // { id: 1, username: 'user1', password:
'$2b$10$L3C0xyOTFEMTFLn1xszou.PNeZKZLJ6iMi4F7vn9mW4Mz93f0M.zi' } //
Password: secret1
];

// Middleware to check if the user is logged in
const checkAuth = (req, res, next) => {
 if (req.session.userId) {
   next();
 } else {
   res.redirect('/login');
 }
};

// Serve HTML file for the login page
app.get('/login', (req, res) => {
 res.sendFile(__dirname + '/login.html');
```

```
});

// Login route
app.post('/login', (req, res) => {
 const { username, password } = req.body;
 const user = users.find((user) => user.username === username);

 // if (user && bcrypt.compareSync(password, user.password)) {
 if (user && password === user.password) {
   req.session.userId = user.id;
   res.redirect('/dashboard');
 } else {
   res.send('Invalid username or password');
 }
});

// Dashboard route (requires authentication)
app.get('/dashboard', checkAuth, (req, res) => {
 res.send(`Welcome, user${req.session.userId}!`);
});

// Logout route
app.get('/logout', (req, res) => {
 req.session.destroy(() => {
   res.redirect('/login');
 });
});

// Start the server
app.listen(PORT, () => {
 console.log(`Server is running on http://localhost:${PORT}`);
});




// const express = require('express');
// const bodyParser = require('body-parser');
// const bcrypt = require('bcrypt');
// const session = require('express-session');
```

```
// const app = express();
// const PORT = process.env.PORT || 3000;

// // Middleware
// app.use(bodyParser.urlencoded({ extended: true }));
// app.use(session({ secret: 'your-secret-key', resave: true, saveUninitialized: true }));

// // Dummy user data (replace this with a database in a real application)
// const users = [
//   { id: 1, username: 'user1', password:
'$2b$10$L3C0xyOTFEMTFLn1xszou.PNeZKZLJ6iMi4F7vn9mW4Mz93f0M.zi' } //
Password: secret1
// ];

// // Middleware to check if the user is logged in
// const checkAuth = (req, res, next) => {
//   if (req.session.userId) {
//     next();
//   } else {
//     res.redirect('/login');
//   }
// };

// // Routes
// app.get('/', (req, res) => {
//   res.send('Home Page');
// });

// app.get('/login', (req, res) => {
//   res.send('Login Page');
// });

// app.post('/login', (req, res) => {
//   const { username, password } = req.body;
//   console.log('\n\n\n\n\nUser hitting route post: login ', req.body)
//   const user = users.find((user) => user.username === username);

//   if (user && bcrypt.compareSync(password, user.password)) {
//     // req.session.userId = user.id;
```

```
//     // res.redirect('/');
//     res.send('Welcome user')
//   } else {
//     res.send('Invalid username or password');
//   }
// });

// app.get('/dashboard', checkAuth, (req, res) => {
//   res.send(`Welcome, user${req.session.userId}!`);
// });

// app.get('/logout', (req, res) => {
//   req.session.destroy(() => {
//     res.redirect('/login');
//   });
// });

// app.listen(PORT, () => {
//   console.log(`Server is running on http://localhost:${PORT}`);
// });
```

```
// index.html
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>User Login</title>
</head>
<body>
  <h1>User Login</h1>
  <form action="/login" method="post">
      <label for="username">Username:</label>
      <input type="text" name="username" required><br>

      <label for="password">Password:</label>
      <input type="password" name="password" required><br>
```

```
      <button type="submit">Login</button>
    </form>
</body>
</html>
```

14

Q.1 Write a Java Program to implement Command Design Pattern for Command Interface with execute() . Use this to create variety of commands for LightOnCommand, LightOffCommand, GarageDoorUpCommand, StereoOnWithCDComman.

```
// Command interface
interface Command {
  void execute();
}

// Receiver classes
class Light {
  void turnOn() { System.out.println("Light is ON"); }
  void turnOff() { System.out.println("Light is OFF"); }
}

class GarageDoor {
  void up() { System.out.println("Garage Door is UP"); }
}

class Stereo {
  void onWithCD() { System.out.println("Stereo is ON with CD"); }
}

// Concrete Command classes
class LightOnCommand implements Command {
  private Light light;

  public LightOnCommand(Light light) {
```

```
    this.light = light;
  }

  public void execute() {
    light.turnOn();
  }
}

class LightOffCommand implements Command {
  private Light light;

  public LightOffCommand(Light light) {
    this.light = light;
  }

  public void execute() {
    light.turnOff();
  }
}

class GarageDoorUpCommand implements Command {
  private GarageDoor door;

  public GarageDoorUpCommand(GarageDoor door) {
    this.door = door;
  }

  public void execute() {
    door.up();
  }
}

class StereoOnWithCDCommand implements Command {
  private Stereo stereo;

  public StereoOnWithCDCommand(Stereo stereo) {
    this.stereo = stereo;
  }

  public void execute() {
```

```
        stereo.onWithCD();
    }
}

// Invoker class
class RemoteControl {
    private Command command;

    void setCommand(Command command) {
        this.command = command;
    }

    void pressButton() {
        command.execute();
    }
}

// Client code
public class CommandPatternExample {
    public static void main(String[] args) {
        Light light = new Light();
        GarageDoor door = new GarageDoor();
        Stereo stereo = new Stereo();

        Command lightOn = new LightOnCommand(light);
        Command lightOff = new LightOffCommand(light);
        Command garageDoorUp = new GarageDoorUpCommand(door);
        Command stereoOnWithCD = new StereoOnWithCDCommand(stereo);

        RemoteControl remoteControl = new RemoteControl();

        remoteControl.setCommand(lightOn);
        remoteControl.pressButton();

        remoteControl.setCommand(lightOff);
        remoteControl.pressButton();

        remoteControl.setCommand(garageDoorUp);
        remoteControl.pressButton();
```

```
    remoteControl.setCommand(stereoOnWithCD);
    remoteControl.pressButton();
  }
}
```

Q.2. Write a python program to find all null values in a given dataset and remove them.

```
import pandas as pd

# Assuming you have a DataFrame named 'df' with your dataset
# Replace 'your_dataset.csv' with the actual file name or provide your dataset in another way
df = pd.read_csv('dataset.csv')

# # Example DataFrame
# data = {'Column1': [1, 2, None, 4, 5],
#         'Column2': ['A', 'B', 'C', None, 'E'],
#         'Column3': [10.1, 20.2, 30.3, 40.4, None]}
# df = pd.DataFrame(data)

# Display the original DataFrame
print("Original DataFrame:")
print(df)

# Find and display null values
```

```
null_values = df.isnull().sum()
print("\nNull Values:")
print(null_values)

# Remove rows with null values
df_cleaned = df.dropna()

# Display the DataFrame after removing null values
print("\nDataFrame after removing null values:")
print(df_cleaned)
```

Write node js script to interact with the filesystem, and serve a web page from a file .

```
const express = require('express');
const fs = require('fs');
const path = require('path');

const app = express();
const PORT = process.env.PORT || 3000;

// Set the path to your HTML file
const filePath = path.join(__dirname, 'index.html');

app.get('/', (req, res) => {
 // Read the HTML file
 fs.readFile(filePath, 'utf8', (err, data) => {
   if (err) {
     res.status(500).send('Internal Server Error');
     return;
   }

   // Send the HTML content as the response
   res.status(200).send(data);
 });
});

app.listen(PORT, () => {
 console.log(`Server is running at http://localhost:${PORT}`);
});
```

```
// const express = require('express');
// const path = require('path');

// const app = express();
// const PORT = process.env.PORT || 4000;

// // Set the static directory to serve HTML, CSS, and JS files
// app.use(express.static(path.join(__dirname, 'public')));

// // Define a route to serve the HTML file
// app.get('/', (req, res) => {
//     res.sendFile(path.join(__dirname, 'public', 'index.html'));
// });

// app.listen(PORT, () => {
//     console.log(`Server is running on http://localhost:${PORT}`);
// });
```

// index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Simple Web Page</title>
</head>
<body>
  <h1>Hello, World!</h1>
  <p>This is a simple web page served by Express.</p>
</body>
</html>
```