*** SLIP 1 ***

** Q1) Write a Java Program to implement I/O Decorator for converting uppercase letters to lower case letters.
==> import java.io.*;

```
class LowerCaseInputStream extends FilterInputStream {
   protected LowerCaseInputStream(InputStream in) {
      super(in);
   }

   public int read() throws IOException {
      int c = super.read();
      return (c == -1 ? c : Character.toLowerCase((char)c));
   }

   public int read(byte[] b, int off, int len) throws IOException {
      int result = super.read(b, off, len);
      for (int i = off; i < off + result; i++) {
         b[i] = (byte)Character.toLowerCase((char)b[i]);
      }
      return result;
   }
}

public class DecoratorDemo {
   public static void main(String[] args) throws Exception {
      InputStream in = new LowerCaseInputStream(
                        new FileInputStream("input.txt"));

      int c;
      while ((c = in.read()) != -1) {
         System.out.print((char)c);
      }
      in.close();
   }
}
```

☑ How to Perform (Steps)
Create a text file named input.txt with uppercase text.
Create file DecoratorDemo.java → paste the code.
Compile:
javac DecoratorDemo.java
Run:
java DecoratorDemo

** Q2) Write a program to sense the available networks using Arduino
==> #include <ESP8266WiFi.h>

```
void setup() {
  Serial.begin(115200);
```

```
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(1000);

  Serial.println("Scanning for WiFi Networks...");
}

void loop() {
  int n = WiFi.scanNetworks();
  Serial.println("Scan Complete");

  if (n == 0) {
    Serial.println("No networks found");
  } else {
    for (int i = 0; i < n; i++) {
      Serial.print(i + 1);
      Serial.print(": ");
      Serial.print(WiFi.SSID(i));
      Serial.print(" (RSSI: ");
      Serial.print(WiFi.RSSI(i));
      Serial.println(" dBm)");
      delay(10);
    }
  }

  Serial.println();
  delay(5000);  // Scan every 5 seconds
}
```

☑ How to Perform (Steps)
Open Arduino IDE
Select board:
Tools → Board → NodeMCU 1.0 (ESP-12E Module
Paste the code.
Connect ESP8266 using USB cable.
Select port:
Tools → Port → COMx
Click Upload.
Open Serial Monitor (Ctrl + Shift + M).
Set 115200 baud rate.

*** SLIP 2 ***

** Q1) Write a Java Program to implement Singleton pattern for multithreading

```
==> class Singleton {
    private static volatile Singleton instance;

    private Singleton() { }

    public static Singleton getInstance() {
        if (instance == null) {
            synchronized (Singleton.class) {
                if (instance == null) {
```

```
                instance = new Singleton();
            }
        }
    }
    return instance;
  }

  public void show() {
    System.out.println("Singleton instance: " + this);
  }
}

public class SingletonThreadDemo {
  public static void main(String[] args) {
    Runnable task = () -> {
      Singleton s = Singleton.getInstance();
      s.show();
    };

    Thread t1 = new Thread(task);
    Thread t2 = new Thread(task);
    Thread t3 = new Thread(task);

    t1.start();
    t2.start();
    t3.start();
  }
}
```

☑ How to Perform (Steps)
Create file:
SingletonThreadDemo.java
Paste the program.
Compile:
javac SingletonThreadDemo.java
Run:
java SingletonThreadDemo

** Q2) Write a program to measure the distance using ultrasonic sensor and make LED blink using Arduino.
==>

```
#define TRIG 9
#define ECHO 8
#define LED 7

void setup() {
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(TRIG, LOW);
```

```
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);

  long duration = pulseIn(ECHO, HIGH);
  long distance = duration * 0.034 / 2;

  Serial.print("Distance: ");
  Serial.println(distance);

  if (distance < 20) {
    digitalWrite(LED, HIGH);
    delay(200);
    digitalWrite(LED, LOW);
    delay(200);
  } else {
    digitalWrite(LED, LOW);
  }

  delay(300);
}
```

☑ How to Perform (Steps)
Connect:
Trig → D9
Echo → D8
LED → D7 (with resistor)
Vcc → 5V, GND → GND
Open Arduino IDE.
Paste the program.
Select board:
Tools → Board → Arduino Uno
Upload.
Open Serial Monitor → 9600 baud.
Move hand/object →
Distance < 20 cm → LED blinks
Else LED OFF

***** SLIP 3 *****

** Q1) Write a JAVA Program to implement built-in support (java.util.Observable) Weather station with members temperature, humidity, pressure and methods mesurmentsChanged(), setMesurment(), getTemperature(), getHumidity(), getPressure()
==> import java.util.Observable;
import java.util.Observer;

```
class WeatherData extends Observable {
    private float temperature;
    private float humidity;
    private float pressure;

    public float getTemperature() { return temperature; }
```

```
    public float getHumidity() { return humidity; }
    public float getPressure() { return pressure; }

    public void measurementsChanged() {
        setChanged();
        notifyObservers();
    }

    public void setMeasurements(float t, float h, float p) {
        this.temperature = t;
        this.humidity = h;
        this.pressure = p;
        measurementsChanged();
    }
}

class CurrentDisplay implements Observer {
    public void update(Observable o, Object arg) {
        WeatherData wd = (WeatherData)o;
        System.out.println("Temp: " + wd.getTemperature() +
                           " Humidity: " + wd.getHumidity() +
                           " Pressure: " + wd.getPressure());
    }
}

public class WeatherStationDemo {
    public static void main(String[] args) {
        WeatherData wd = new WeatherData();
        CurrentDisplay cd = new CurrentDisplay();

        wd.addObserver(cd);

        wd.setMeasurements(30, 70, 1010);
        wd.setMeasurements(32, 60, 1008);
    }
}
```

☑ How to Perform (Steps)
Create file: WeatherStationDemo.java
Paste the code.
Compile:
javac WeatherStationDemo.java
Run:
java WeatherStationDemo

** Q2) Write a program to detects the vibration of an object with sensor using Arduino.
==>
```
int vibrationPin = 8;   // Vibration sensor output pin
int ledPin = 7;         // LED indicator

void setup() {
  pinMode(vibrationPin, INPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
```

```
}

void loop() {
  int value = digitalRead(vibrationPin);

  if (value == HIGH) {
    Serial.println("Vibration Detected!");
    digitalWrite(ledPin, HIGH);
    delay(200);
    digitalWrite(ledPin, LOW);
  } else {
    digitalWrite(ledPin, LOW);
  }

  delay(100);
}
```

☑ How to Perform (Steps)
Connect vibration sensor:
Signal → D8
VCC → 5V
GND → GND
Connect LED to D7 with resistor.
Open Arduino IDE → paste the code.
Select board → Arduino Uno.
Upload the program.
Open Serial Monitor → 9600 baud.
Tap/shake the sensor → LED blinks and message appears

*** SLIP 4 ***

** Q1) Write a Java Program to implement Factory method for Pizza Store with createPizza(), orederPizza(), prepare(), Bake(), cut(), box(). Use this to create variety of pizza's like NyStyleCheesePizza, ChicagoStyleCheesePizza etc.
==>
```
// ----- Product -----
abstract class Pizza {
    String name;

    void prepare() { System.out.println("Preparing " + name); }
    void bake()    { System.out.println("Baking " + name); }
    void cut()     { System.out.println("Cutting " + name); }
    void box()     { System.out.println("Boxing " + name); }

    public String getName() { return name; }
}

// ----- Concrete Products -----
class NYStyleCheesePizza extends Pizza {
    NYStyleCheesePizza() {
        name = "NY Style Cheese Pizza";
    }
}
```

```
class ChicagoStyleCheesePizza extends Pizza {
    ChicagoStyleCheesePizza() {
        name = "Chicago Style Cheese Pizza";
    }

    @Override
    void cut() {
        System.out.println("Cutting Chicago Deep-Dish Style");
    }
}

// ----- Creator -----
abstract class PizzaStore {
    public Pizza orderPizza(String type) {
        Pizza pizza = createPizza(type);

        pizza.prepare();
        pizza.bake();
        pizza.cut();
        pizza.box();

        return pizza;
    }

    abstract Pizza createPizza(String type);
}

// ----- Concrete Creators -----
class NYPizzaStore extends PizzaStore {
    Pizza createPizza(String type) {
        if (type.equals("cheese")) return new NYStyleCheesePizza();
        return null;
    }
}

class ChicagoPizzaStore extends PizzaStore {
    Pizza createPizza(String type) {
        if (type.equals("cheese")) return new ChicagoStyleCheesePizza();
        return null;
    }
}

// ----- Test -----
public class FactoryPizzaDemo {
    public static void main(String[] args) {
        PizzaStore nyStore = new NYPizzaStore();
        PizzaStore chicagoStore = new ChicagoPizzaStore();

        nyStore.orderPizza("cheese");
        chicagoStore.orderPizza("cheese");
    }
}
```

☑ How to Perform (Steps)

Create file:
FactoryPizzaDemo.java
Paste the code.
Compile:
javac FactoryPizzaDemo.java
Run:
java FactoryPizzaDemo

** Q2) Write a program to sense a finger when it is placed on the board Arduino.
==> int touchPin = 8;   // Touch sensor signal pin
int ledPin = 7;   // LED indicator

```
void setup() {
  pinMode(touchPin, INPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int value = digitalRead(touchPin);

  if (value == HIGH) {          // Finger touched
    Serial.println("Finger Detected!");
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }

  delay(100);
}
```

☑ How to Perform (Steps)
Connect Touch Sensor (TTP223):
OUT → D8
VCC → 5V
GND → GND
Connect LED to D7 with resistor.
Open Arduino IDE → paste the program.
Select board → Arduino Uno.
Upload.
Open Serial Monitor → 9600 baud.
Place finger on the sensor → you will see:
Finger Detected!
And LED turns ON.
Remove finger → LED OFF.

*** SLIP 5 ***

** Q1) Write a Java Program to implement Adapter pattern for Enumeration iterator
==> import java.util.*;

```
class EnumIteratorAdapter implements Iterator<Object> {
```

```
    private Enumeration<?> enumeration;

    EnumIteratorAdapter(Enumeration<?> e) {
        this.enumeration = e;
    }

    public boolean hasNext() {
        return enumeration.hasMoreElements();
    }

    public Object next() {
        return enumeration.nextElement();
    }
}

public class AdapterDemo {
    public static void main(String[] args) {
        Vector<String> v = new Vector<>();
        v.add("A");
        v.add("B");
        v.add("C");

        Enumeration<String> e = v.elements();
        Iterator<Object> it = new EnumIteratorAdapter(e);

        while (it.hasNext()) {
            System.out.println(it.next());
        }
    }
}
```

☑ How to Perform (Steps)
Create file: AdapterDemo.java
Paste the above program.
Compile:
javac AdapterDemo.java
Run:
java AdapterDemo

** Q2) Write a program to connect with the available Wi-Fi using Arduino.
==> #include <ESP8266WiFi.h>

```
const char* ssid = "YourWiFiName";
const char* password = "YourWiFiPassword";

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);

  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);
```

```
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi Connected!");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
}
```

☑ How to Perform (Steps)

Replace:
YourWiFiName
YourWiFiPassword
with your actual Wi-Fi name and password.
Open Arduino IDE.
Select board:
Tools → Board → NodeMCU 1.0 (ESP8266)
Paste the code.
Connect ESP8266 with USB.
Select COM port:
Tools → Port → COMx
Click Upload.
Open Serial Monitor → set 115200 baud.

*** SLIP 6 ***

** Q1) Write a Java Program to implement command pattern to test Remote Control

```
==> // Command Interface
interface Command {
    void execute();
}

// Receiver
class Light {
    public void on() {
        System.out.println("Light is ON");
    }
    public void off() {
        System.out.println("Light is OFF");
    }
}

// Concrete Commands
class LightOnCommand implements Command {
    Light light;
    LightOnCommand(Light l) { light = l; }
    public void execute() { light.on(); }
```

```
}

class LightOffCommand implements Command {
    Light light;
    LightOffCommand(Light l) { light = l; }
    public void execute() { light.off(); }
}

// Invoker
class RemoteControl {
    Command slot;
    public void setCommand(Command c) { slot = c; }
    public void buttonPressed() { slot.execute(); }
}

// Test Program
public class RemoteControlTest {
    public static void main(String[] args) {
        Light light = new Light();

        Command onCommand = new LightOnCommand(light);
        Command offCommand = new LightOffCommand(light);

        RemoteControl remote = new RemoteControl();

        remote.setCommand(onCommand);
        remote.buttonPressed();

        remote.setCommand(offCommand);
        remote.buttonPressed();
    }
}
```

☑ How to Perform (Steps)
Create file:
RemoteControlTest.java
Paste the program.
Compile:
javac RemoteControlTest.java
Run:
java RemoteControlTest

** Q2) Write a program to get temperature notification using Arduino.
==>

```
int sensorPin = A0;   // LM35 output pin
int ledPin = 7;       // LED for alert
float temperature = 0;

void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
}

void loop() {
```

```
  int reading = analogRead(sensorPin);

  // Convert LM35 reading to °C
  temperature = (reading * 5.0 * 100.0) / 1024;

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" C");

  // Notification when temperature is high
  if (temperature > 30) {
    Serial.println("TEMP ALERT!");
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }

  delay(1000);
}
```

☑ How to Perform (Steps)
Connect LM35 sensor:
VCC → 5V
GND → GND
OUT → A0
Connect LED to pin 7 with resistor.
Open Arduino IDE → paste the code.
Select board → Arduino Uno.
Upload the code.
Open Serial Monitor → set 9600 baud.
Heat the LM35 slightly:
Temperature > 30°C → LED ON + “TEMP ALERT!”
Temperature ≤ 30°C → LED OFF

*** SLIP 7 ***

** Q1) Write a Java Program to implement undo command to test Ceiling fan.
==>

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

// Receiver
class CeilingFan {
    public static final int HIGH = 3;
    public static final int MEDIUM = 2;
    public static final int LOW = 1;
    public static final int OFF = 0;

    private int speed;

    public CeilingFan() {
```

```
        speed = OFF;
    }

    public void high() {
        speed = HIGH;
        System.out.println("Ceiling Fan set to HIGH");
    }

    public void medium() {
        speed = MEDIUM;
        System.out.println("Ceiling Fan set to MEDIUM");
    }

    public void low() {
        speed = LOW;
        System.out.println("Ceiling Fan set to LOW");
    }

    public void off() {
        speed = OFF;
        System.out.println("Ceiling Fan turned OFF");
    }

    public int getSpeed() {
        return speed;
    }
}

// Concrete Command
class FanHighCommand implements Command {
    CeilingFan fan;
    int prevSpeed;

    FanHighCommand(CeilingFan f) {
        fan = f;
    }

    public void execute() {
        prevSpeed = fan.getSpeed();
        fan.high();
    }

    public void undo() {
        if (prevSpeed == CeilingFan.HIGH) fan.high();
        else if (prevSpeed == CeilingFan.MEDIUM) fan.medium();
        else if (prevSpeed == CeilingFan.LOW) fan.low();
        else fan.off();
    }
}

// Invoker
class RemoteControl {
    Command slot;
```

```
    public void setCommand(Command c) {
        slot = c;
    }

    public void buttonPressed() {
        slot.execute();
    }

    public void undoPressed() {
        slot.undo();
    }
}

// Test Program
public class CeilingFanUndoTest {
    public static void main(String[] args) {
        CeilingFan fan = new CeilingFan();

        Command fanHigh = new FanHighCommand(fan);
        RemoteControl remote = new RemoteControl();

        remote.setCommand(fanHigh);

        remote.buttonPressed();   // Set HIGH
        remote.undoPressed();     // Undo to previous speed
    }
}
```

☑ How to Perform (Steps)
Create file:
CeilingFanUndoTest.java
Paste the program.
Compile:
javac CeilingFanUndoTest.java
Run:
java CeilingFanUndoTest

** Q2) Write a program for LDR to vary the light intensity of LED using Arduino.

```
==> int ldrPin = A0;     // LDR output
int ledPin = 9;      // PWM pin for LED
int ldrValue = 0;
int ledValue = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  ldrValue = analogRead(ldrPin);          // Read LDR value (0–1023)
  ledValue = map(ldrValue, 0, 1023, 255, 0); // Convert to LED brightness (reverse)

  analogWrite(ledPin, ledValue);          // Set LED brightness
```

```
  Serial.print("LDR: ");
  Serial.print(ldrValue);
  Serial.print("  LED: ");
  Serial.println(ledValue);

  delay(100);
}
```

☑ How to Perform (Steps)

1. LDR Wiring

LDR → A0

One side → 5V

Other side → A0 + 10kΩ resistor → GND

2. LED Wiring

LED +ve → Pin 9

LED -ve → GND (through resistor)

3. Upload Program

Open Arduino IDE

Paste code

Select Arduino Uno

Upload

4. Test

Cover the LDR → LED becomes brighter

Shine light → LED becomes dimmer

*** SLIP 8 ***

** Q1) Write a Java Program to implement State Pattern for Gumball Machine. Create instance variable that holds current state from there, we just need to handle all actions, behaviors and state transition that can happen

==>

```
// ----- State Interface -----
interface State {
    void insertCoin();
    void ejectCoin();
    void turnCrank();
    void dispense();
}

// ----- Concrete States -----
class NoCoinState implements State {
    GumballMachine machine;

    NoCoinState(GumballMachine m) { machine = m; }

    public void insertCoin() {
        System.out.println("Coin inserted");
        machine.setState(machine.hasCoinState);
    }

    public void ejectCoin() {
        System.out.println("No coin to eject");
    }
```

```
    public void turnCrank() {
        System.out.println("Insert coin first");
    }

    public void dispense() {
        System.out.println("No coin inserted");
    }
}

class HasCoinState implements State {
    GumballMachine machine;

    HasCoinState(GumballMachine m) { machine = m; }

    public void insertCoin() {
        System.out.println("Already inserted a coin");
    }

    public void ejectCoin() {
        System.out.println("Coin returned");
        machine.setState(machine.noCoinState);
    }

    public void turnCrank() {
        System.out.println("Crank turned...");
        machine.setState(machine.soldState);
    }

    public void dispense() {
        System.out.println("Turn crank first");
    }
}

class SoldState implements State {
    GumballMachine machine;

    SoldState(GumballMachine m) { machine = m; }

    public void insertCoin() {
        System.out.println("Please wait... dispensing");
    }

    public void ejectCoin() {
        System.out.println("Already turned crank");
    }

    public void turnCrank() {
        System.out.println("Turning again won't help");
    }

    public void dispense() {
        machine.releaseBall();
        if (machine.count > 0)
```

```
            machine.setState(machine.noCoinState);
        else {
            System.out.println("Machine is empty");
            machine.setState(machine.soldOutState);
        }
    }
}

class SoldOutState implements State {
    GumballMachine machine;

    SoldOutState(GumballMachine m) { machine = m; }

    public void insertCoin() {
        System.out.println("Machine out of gumballs");
    }

    public void ejectCoin() {
        System.out.println("No coin");
    }

    public void turnCrank() {
        System.out.println("No gumballs left");
    }

    public void dispense() {
        System.out.println("Nothing to dispense");
    }
}

// ----- Context (Gumball Machine) -----
class GumballMachine {
    State noCoinState;
    State hasCoinState;
    State soldState;
    State soldOutState;

    State currentState;
    int count = 0;

    GumballMachine(int numberGumballs) {
        noCoinState = new NoCoinState(this);
        hasCoinState = new HasCoinState(this);
        soldState = new SoldState(this);
        soldOutState = new SoldOutState(this);

        count = numberGumballs;

        currentState = (count > 0) ? noCoinState : soldOutState;
    }

    public void insertCoin() { currentState.insertCoin(); }
    public void ejectCoin()  { currentState.ejectCoin(); }
    public void turnCrank()  { currentState.turnCrank(); currentState.dispense(); }
```

```
    void setState(State s) { currentState = s; }

    void releaseBall() {
        System.out.println("A gumball comes rolling out...");
        if (count > 0) count--;
    }
}

// ----- Test Program -----
public class GumballStateTest {
    public static void main(String[] args) {
        GumballMachine gm = new GumballMachine(2);

        gm.insertCoin();
        gm.turnCrank();

        gm.insertCoin();
        gm.turnCrank();

        gm.insertCoin();
        gm.turnCrank();  // Machine empty
    }
}
```

☑ How to Perform (Steps)
Create file: GumballStateTest.java
Paste the entire program.
Compile:
javac GumballStateTest.java
Run:
java GumballStateTest

** Q2) Start Raspberry Pi and execute various Linux commands in command terminal window: ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less, ps, sudo, cron, chown, chgrp, pingetc.
==> ☑ How to Start Raspberry Pi
Power on Raspberry Pi.
Login (if asked):
pi
raspberry
Open Terminal from the taskbar.

☑ Linux Commands to Execute (with sample usage)
You ONLY need to type these in the terminal.

1. ls – list files
ls
ls -l

2. cd – change directory
cd /home/pi
cd Desktop

3. touch – create empty file
touch file1.txt

4. mv – rename or move file
mv file1.txt newname.txt

5. rm – delete file
rm newname.txt

6. man – show help manual
man ls

7. mkdir – create directory
mkdir myfolder

8. rmdir – remove empty directory
rmdir myfolder

9. tar – archive files

Create tar:
tar -cvf data.tar myfolder

Extract:
tar -xvf data.tar

10. gzip – compress file
gzip file.txt
gunzip file.txt.gz

11. cat – show file content
cat file.txt

12. more – paged view
more file.txt

13. less – better paged view
less file.txt

14. ps – process list
ps

15. sudo – run command as root
sudo apt update

16. cron – schedule tasks (open crontab)
crontab -e

17. chown – change owner
sudo chown pi:pi file.txt

18. chgrp – change group
sudo chgrp pi file.txt

19. ping – test network
ping google.com

***** SLIP 9 *****

**** Q1) Design simple HR Application using Spring Framework**
==> src/
└── com/hr/
├── Employee.java
├── EmployeeService.java
├── HRApp.java
applicationContext.xml

1. Employee.java

```
package com.hr;

public class Employee {
    private int id;
    private String name;
    private String department;

    public Employee() {}

    public Employee(int id, String name, String dept) {
        this.id = id;
        this.name = name;
        this.department = dept;
    }

    public void show() {
        System.out.println("ID: " + id);
        System.out.println("Name: " + name);
        System.out.println("Department: " + department);
    }
}
```

2. EmployeeService.java

```
package com.hr;

public class EmployeeService {
    private Employee emp;

    public void setEmployee(Employee e) {
        this.emp = e;
    }

    public void displayEmployee() {
        emp.show();
    }
}
```

3. HRApp.java

```
package com.hr;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class HRApp {
   public static void main(String[] args) {
      ApplicationContext ctx =
            new ClassPathXmlApplicationContext("applicationContext.xml");

      EmployeeService service = (EmployeeService) ctx.getBean("empService");
      service.displayEmployee();
   }
}
```

4. applicationContext.xml
Place this file in src/ or resources/ folder.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="
      http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd">

   <bean id="employee" class="com.hr.Employee">
      <constructor-arg value="101" />
      <constructor-arg value="John Doe" />
      <constructor-arg value="HR" />
   </bean>

   <bean id="empService" class="com.hr.EmployeeService">
      <property name="employee" ref="employee" />
   </bean>

</beans>
```

☑ How to Perform (Steps)
1. Add Spring JARs
You MUST include the following jars in your lib folder:
spring-core
spring-context
spring-beans
spring-expression
(or use Maven if allowed)
2. Compile

```
javac -cp "lib/*" com/hr/*.java
```

3. Run

```
java -cp "lib/*:." com.hr.HRApp
```

**** Q2) Write python programs on Pi : a) Read your name and print Hello message with name b) Read two numbers and print their sum, difference, product and division. c) Word and character count of a given string. d) Area of a given shape (rectangle, triangle and circle)**

reading shape and appropriate values from standard input.
==> Perfect to run on Raspberry Pi Terminal using:
python3 filename.py

☑ a) Read your name and print Hello message

```
name = input("Enter your name: ")
print("Hello", name)
```

☑ b) Read two numbers and print sum, difference, product, division

```
a = float(input("Enter first number: "))
b = float(input("Enter second number: "))

print("Sum =", a + b)
print("Difference =", a - b)
print("Product =", a * b)

if b != 0:
    print("Division =", a / b)
else:
    print("Division not possible")
```

☑ c) Word and character count of a given string

```
s = input("Enter a string: ")

words = len(s.split())
chars = len(s)

print("Word Count =", words)
print("Character Count =", chars)
```

☑ d) Area of a given shape (rectangle, triangle, circle)

```
shape = input("Enter shape (rectangle/triangle/circle): ")

if shape == "rectangle":
    l = float(input("Enter length: "))
    w = float(input("Enter width: "))
    print("Area =", l * w)

elif shape == "triangle":
    b = float(input("Enter base: "))
    h = float(input("Enter height: "))
    print("Area =", 0.5 * b * h)

elif shape == "circle":
    r = float(input("Enter radius: "))
    print("Area =", 3.14 * r * r)

else:
    print("Invalid shape")
```

*** SLIP 10 ***

** Q1) Write a Java Program to implement Strategy Pattern for Duck Behavior. Create

instance variable that holds current state of Duck from there, we just need to handle all Flying Behaviors and Quack Behavior

```
==> // ----- Strategy Interfaces -----
interface FlyBehavior {
    void fly();
}

interface QuackBehavior {
    void quack();
}

// ----- Fly Behaviors -----
class FlyWithWings implements FlyBehavior {
    public void fly() {
        System.out.println("Flying with wings");
    }
}

class FlyNoWay implements FlyBehavior {
    public void fly() {
        System.out.println("Cannot fly");
    }
}

// ----- Quack Behaviors -----
class Quack implements QuackBehavior {
    public void quack() {
        System.out.println("Quack!");
    }
}

class MuteQuack implements QuackBehavior {
    public void quack() {
        System.out.println("<< Silent >>");
    }
}

// ----- Duck Base Class -----
abstract class Duck {
    FlyBehavior flyBehavior;
    QuackBehavior quackBehavior;

    public void performFly() {
        flyBehavior.fly();
    }

    public void performQuack() {
        quackBehavior.quack();
    }

    public void setFlyBehavior(FlyBehavior fb) {
        flyBehavior = fb;
    }
```

```
    public void setQuackBehavior(QuackBehavior qb) {
        quackBehavior = qb;
    }

    abstract void display();
}

// ----- Concrete Duck -----
class MallardDuck extends Duck {
    MallardDuck() {
        flyBehavior = new FlyWithWings();
        quackBehavior = new Quack();
    }

    void display() {
        System.out.println("I am a Mallard Duck");
    }
}

// Test Program
public class DuckStrategyTest {
    public static void main(String[] args) {
        Duck d = new MallardDuck();

        d.display();
        d.performFly();
        d.performQuack();

        // Change behavior at runtime
        d.setFlyBehavior(new FlyNoWay());
        d.performFly();
    }
}
```

☑ How to Perform (Steps)
Create file: DuckStrategyTest.java
Paste the entire program.
Compile:
javac DuckStrategyTest.java
Run:
java DuckStrategyTest

** Q2) Write python programs on Pi like: a) Print a name 'n' times, where name and n are read from standard input, using for and while loops. b) Handle Divided by Zero Exception. c) Print current time for 10 times with an interval of10seconds. d) Read a fileline byline and print the word count of each line
==> ☑ a) Print a name ‘n’ times (using for loop & while loop)
1. For Loop

```
name = input("Enter name: ")
n = int(input("Enter number: "))

for i in range(n):
    print(name)
```

2. While Loop

```
name = input("Enter name: ")
n = int(input("Enter number: "))

i = 0
while i < n:
    print(name)
    i += 1
```

☑ b) Handle Divide-by-Zero Exception

```
try:
    a = int(input("Enter numerator: "))
    b = int(input("Enter denominator: "))
    print("Result =", a / b)

except ZeroDivisionError:
    print("Error: Division by zero not allowed")
```

☑ c) Print current time 10 times with 10-second interval

```
import time
from datetime import datetime

for i in range(10):
    print(datetime.now())
    time.sleep(10)   # wait 10 seconds
```

☑ d) Read a file line by line & print word count of each line

```
filename = input("Enter file name: ")

with open(filename, "r") as f:
    for line in f:
        words = len(line.split())
        print("Line:", line.strip())
        print("Word Count:", words)
```

***** SLIP 11 *****

**** Q1) Write a java program to implement Adapter pattern to design Heart Model to Beat Model**

==>

```
// Target Interface (what we want)
interface BeatModel {
    int getBeat();
}

// Adaptee (existing class)
class HeartModel {
    private int heartRate = 72;

    public int getHeartRate() {
        return heartRate;
    }
}
```

```
// Adapter (converts HeartModel to BeatModel)
class HeartAdapter implements BeatModel {
    private HeartModel heart;

    public HeartAdapter(HeartModel h) {
        this.heart = h;
    }

    public int getBeat() {
        return heart.getHeartRate();   // adaptation happens here
    }
}

// Test Class
public class HeartBeatAdapterTest {
    public static void main(String[] args) {

        HeartModel heart = new HeartModel();       // Adaptee
        BeatModel beat = new HeartAdapter(heart);  // Adapter

        System.out.println("Heart Rate: " + heart.getHeartRate());
        System.out.println("Beat from Adapter: " + beat.getBeat());
    }
}
```

☑ How to Perform (VERY SIMPLE)
Save as:
HeartBeatAdapterTest.java
Compile:
javac HeartBeatAdapterTest.java
Run:
java HeartBeatAdapterTest

** Q2) Run some python programs on Pi like a) Light an LED through Python program b) Get input from two switches and switch on corresponding LEDs c) Flash an LED at a given on time and off time cycle, where the two times are taken from a file
==> ⚠ Before running any Python GPIO program, always include this at the top:

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
```

And at the end:

```
GPIO.cleanup()
```

☑ a) Light an LED through Python Program
Circuit
LED → GPIO 18
Resistor → GND

```
import RPi.GPIO as GPIO
import time
```

```
GPIO.setmode(GPIO.BCM)
led = 18

GPIO.setup(led, GPIO.OUT)

GPIO.output(led, True)
print("LED ON")
time.sleep(3)

GPIO.output(led, False)
print("LED OFF")

GPIO.cleanup()
```

☑ b) Get input from two switches and turn ON corresponding LEDs
Circuit
Switch1 → GPIO 17
Switch2 → GPIO 27
LED1 → GPIO 22
LED2 → GPIO 23

```
import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)

sw1 = 17
sw2 = 27
led1 = 22
led2 = 23

GPIO.setup(sw1, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(sw2, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(led1, GPIO.OUT)
GPIO.setup(led2, GPIO.OUT)

try:
    while True:
        if GPIO.input(sw1) == 0:
            GPIO.output(led1, True)
        else:
            GPIO.output(led1, False)

        if GPIO.input(sw2) == 0:
            GPIO.output(led2, True)
        else:
            GPIO.output(led2, False)

        time.sleep(0.1)

except KeyboardInterrupt:
    GPIO.cleanup()
```

☑ c) Flash an LED with ON/OFF time taken from a file

```
File: timing.txt
on=1
off=2

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BCM)
led = 18
GPIO.setup(led, GPIO.OUT)

# Read from file
on_time = 1
off_time = 1

with open("timing.txt", "r") as f:
    for line in f:
        if "on" in line:
            on_time = float(line.split("=")[1])
        if "off" in line:
            off_time = float(line.split("=")[1])

try:
    while True:
        GPIO.output(led, True)
        time.sleep(on_time)

        GPIO.output(led, False)
        time.sleep(off_time)

except KeyboardInterrupt:
    GPIO.cleanup()
```

***** SLIP 12 *****

** Q1) Write a Java Program to implement Decorator Pattern for interface Car to define the assemble() method and then decorate it to Sports car and Luxury Car

==>

```
// ----- Component Interface -----
interface Car {
    void assemble();
}

// ----- Concrete Component -----
class BasicCar implements Car {
    public void assemble() {
        System.out.println("Basic Car");
    }
}

// ----- Decorator Base Class -----
class CarDecorator implements Car {
    protected Car car;
```

```
    public CarDecorator(Car c) {
        this.car = c;
    }

    public void assemble() {
        car.assemble();
    }
}

// ----- Concrete Decorators -----
class SportsCar extends CarDecorator {
    public SportsCar(Car c) {
        super(c);
    }

    public void assemble() {
        super.assemble();
        System.out.println("Adding features of Sports Car");
    }
}

class LuxuryCar extends CarDecorator {
    public LuxuryCar(Car c) {
        super(c);
    }

    public void assemble() {
        super.assemble();
        System.out.println("Adding features of Luxury Car");
    }
}

// ----- Test Class -----
public class CarDecoratorTest {
    public static void main(String[] args) {

        System.out.println("Sports Car:");
        Car sports = new SportsCar(new BasicCar());
        sports.assemble();

        System.out.println("\nLuxury Car:");
        Car luxury = new LuxuryCar(new BasicCar());
        luxury.assemble();

        System.out.println("\nSports + Luxury Car:");
        Car both = new SportsCar(new LuxuryCar(new BasicCar()));
        both.assemble();
    }
}
```

☑ How to Perform (Steps)
Save the program as:
CarDecoratorTest.java
Compile:

javac CarDecoratorTest.java
Run:
java CarDecoratorTest

** Q2) Write a program to sense the available networks using Arduino
==> #include <ESP8266WiFi.h>

```
void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);   // Station mode
  WiFi.disconnect();
  delay(1000);

  Serial.println("Scanning for available WiFi networks...");
}

void loop() {
  int n = WiFi.scanNetworks();
  Serial.println("Scan complete");

  if (n == 0) {
    Serial.println("No networks found");
  } else {
    for (int i = 0; i < n; i++) {
      Serial.print(i + 1);
      Serial.print(": ");
      Serial.print(WiFi.SSID(i));     // Network name
      Serial.print(" (Signal: ");
      Serial.print(WiFi.RSSI(i));     // Signal strength
      Serial.println(" dBm)");
      delay(10);
    }
  }

  Serial.println();
  delay(5000); // Scan every 5 seconds
}
```

☑ How to Perform (Steps)
Open Arduino IDE.
Select board:
Tools → Board → NodeMCU 1.0 (ESP-12E Module)
Copy–paste the program.
Connect ESP8266 → USB cable.
Select port:
Tools → Port → COMx
Click Upload.
Open Serial Monitor → set 115200 baud.

*** SLIP 13 ***

** Q1) 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

```
==> // ----- Volt Class -----
class Volt {
    private int volts;

    public Volt(int v) { volts = v; }
    public int getVolts() { return volts; }
    public void setVolts(int v) { volts = v; }
}

// ----- Socket Class (Adaptee) -----
class Socket {
    public Volt getVolt() {
        return new Volt(120);   // default voltage
    }
}

// ----- Adapter Interface -----
interface MobileCharger {
    Volt get3Volt();
    Volt get12Volt();
    Volt get120Volt();
}

// ----- Class Adapter (uses inheritance) -----
class SocketAdapter extends Socket implements MobileCharger {

    public Volt get3Volt() {
        return convertVolt(getVolt(), 40);   // 120 / 40 = 3
    }

    public Volt get12Volt() {
        return convertVolt(getVolt(), 10);   // 120 / 10 = 12
    }

    public Volt get120Volt() {
        return getVolt();   // original
    }

    private Volt convertVolt(Volt v, int divisor) {
        return new Volt(v.getVolts() / divisor);
    }
}

// ----- Test Program -----
public class MobileChargerTest {
    public static void main(String[] args) {

        MobileCharger charger = new SocketAdapter();

        System.out.println("3 Volt Output: " + charger.get3Volt().getVolts());
        System.out.println("12 Volt Output: " + charger.get12Volt().getVolts());
```

```
        System.out.println("120 Volt Output: " + charger.get120Volt().getVolts());
    }
}
```

☑ How to Perform (Steps)
Save file as:
MobileChargerTest.java
Compile:
javac MobileChargerTest.java
Run:
java MobileChargerTest

** Q2) Write a program to measure the distance using ultrasonic sensor and make LED blink using Arduino.
==>

```
#define TRIG 9
#define ECHO 8
#define LED 7

void setup() {
  pinMode(TRIG, OUTPUT);
  pinMode(ECHO, INPUT);
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(TRIG, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);

  long duration = pulseIn(ECHO, HIGH);
  long distance = duration * 0.034 / 2;

  Serial.print("Distance: ");
  Serial.println(distance);

  if (distance < 20) {       // threshold 20 cm
    digitalWrite(LED, HIGH);
    delay(200);
    digitalWrite(LED, LOW);
    delay(200);
  } else {
    digitalWrite(LED, LOW);
  }

  delay(300);
}
```

☑ How to Perform (Steps)
1. Connections
Ultrasonic (HC-SR04):

VCC → 5V
GND → GND
Trig → D9
Echo → D8
LED:
Positive → D7
Negative → GND (via resistor)

2. Upload
Open Arduino IDE
Paste the program
Select board: Arduino UNO
Upload code

3. Test
Open Serial Monitor → 9600 baud
Move your hand near the sensor
If distance < 20 cm, LED blinks
Else LED remains OFF

***** SLIP 14 *****

**** Q1) 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();
}

// ----- Receivers -----
class Light {
    public void on()  { System.out.println("Light is ON"); }
    public void off() { System.out.println("Light is OFF"); }
}

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

class Stereo {
    public void on() { System.out.println("Stereo is ON"); }
    public void setCD() { System.out.println("Stereo set to CD mode"); }
    public void setVolume(int v) { System.out.println("Stereo volume set to " + v); }
}

// ----- Concrete Commands -----
class LightOnCommand implements Command {
    Light light;
    LightOnCommand(Light l) { light = l; }
    public void execute() { light.on(); }
}
```

```
class LightOffCommand implements Command {
    Light light;
    LightOffCommand(Light l) { light = l; }
    public void execute() { light.off(); }
}

class GarageDoorUpCommand implements Command {
    GarageDoor door;
    GarageDoorUpCommand(GarageDoor d) { door = d; }
    public void execute() { door.up(); }
}

class StereoOnWithCDCommand implements Command {
    Stereo stereo;
    StereoOnWithCDCommand(Stereo s) { stereo = s; }
    public void execute() {
        stereo.on();
        stereo.setCD();
        stereo.setVolume(10);
    }
}

// ----- Invoker (Remote Control) -----
class RemoteControl {
    private Command slot;
    public void setCommand(Command c) { slot = c; }
    public void buttonPressed() { slot.execute(); }
}

// ----- Test Program -----
public class CommandPatternTest {
    public static void main(String[] args) {

        RemoteControl remote = new RemoteControl();

        Light light = new Light();
        GarageDoor door = new GarageDoor();
        Stereo stereo = new Stereo();

        remote.setCommand(new LightOnCommand(light));
        remote.buttonPressed();

        remote.setCommand(new LightOffCommand(light));
        remote.buttonPressed();

        remote.setCommand(new GarageDoorUpCommand(door));
        remote.buttonPressed();

        remote.setCommand(new StereoOnWithCDCommand(stereo));
        remote.buttonPressed();
    }
}
```

☑ How to Perform (Steps)

Save file as:
CommandPatternTest.java
Compile:
javac CommandPatternTest.java
Run:
java CommandPatternTest

**** Q2) Write a program to detects the vibration of an object with sensor using Arduino.**
==> int sensorPin = 8; // vibration sensor output
int ledPin = 7; // LED for alert

```
void setup() {
  pinMode(sensorPin, INPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int val = digitalRead(sensorPin);

  if (val == HIGH) {
    Serial.println("VIBRATION DETECTED!");
    digitalWrite(ledPin, HIGH);
    delay(200);
    digitalWrite(ledPin, LOW);
  } else {
    digitalWrite(ledPin, LOW);
  }

  delay(100);
}
```

☑ How to Perform (Steps)
1. Connections
Vibration Sensor (SW-420):
VCC → 5V
GND → GND
OUT → D8
LED:
Positive → D7
Negative → GND (through resistor)

2. Upload Program
Open Arduino IDE
Paste code
Select board: Arduino UNO
Upload

3. Test
Open Serial Monitor → 9600 baud
Shake/tap the sensor

** Q1) Write a Java Program to implement Facade Design Pattern for Home Theater
==>

```
// ----- Subsystems -----
class DVDPlayer {
    public void on() { System.out.println("DVD Player ON"); }
    public void off() { System.out.println("DVD Player OFF"); }
    public void play(String movie) { System.out.println("Playing: " + movie); }
}

class Projector {
    public void on() { System.out.println("Projector ON"); }
    public void off() { System.out.println("Projector OFF"); }
}

class SoundSystem {
    public void on() { System.out.println("Sound System ON"); }
    public void off() { System.out.println("Sound System OFF"); }
    public void setVolume(int v) { System.out.println("Volume set to: " + v); }
}

// ----- Facade Class -----
class HomeTheaterFacade {
    private DVDPlayer dvd;
    private Projector projector;
    private SoundSystem sound;

    public HomeTheaterFacade(DVDPlayer d, Projector p, SoundSystem s) {
        dvd = d;
        projector = p;
        sound = s;
    }

    public void watchMovie(String movie) {
        System.out.println("\n--- Starting Movie ---");
        dvd.on();
        projector.on();
        sound.on();
        sound.setVolume(8);
        dvd.play(movie);
    }

    public void endMovie() {
        System.out.println("\n--- Shutting Down Movie ---");
        dvd.off();
        projector.off();
        sound.off();
    }
}

// ----- Test Class -----
public class HomeTheaterTest {
    public static void main(String[] args) {
```

```
        DVDPlayer dvd = new DVDPlayer();
        Projector projector = new Projector();
        SoundSystem sound = new SoundSystem();

        HomeTheaterFacade home = new HomeTheaterFacade(dvd, projector, sound);

        home.watchMovie("Avengers Endgame");
        home.endMovie();
    }
}
```

☑ How to Perform (Steps)
Save as:
HomeTheaterTest.java
Compile:
javac HomeTheaterTest.java
Run:
java HomeTheaterTest

** Q2) Write a program to sense a finger when it is placed on the board Arduino.
==>
```
int touchPin = 8;   // Touch sensor output pin
int ledPin = 7;     // LED indicator

void setup() {
  pinMode(touchPin, INPUT);
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  int value = digitalRead(touchPin);

  if (value == HIGH) {
    Serial.println("FINGER DETECTED!");
    digitalWrite(ledPin, HIGH);
  } else {
    digitalWrite(ledPin, LOW);
  }

  delay(100);
}
```

☑ How to Perform (Steps)
Connections
Touch Sensor (TTP223):
VCC → 5V
GND → GND
OUT → D8
LED:
+ → D7
− → GND

Upload

Open Arduino IDE
Paste code
Select Arduino UNO
Upload

Test
Touch the sensor's metal pad

*** SLIP 16 ***

** Q1) Write a Java Program to implement Observer Design Pattern for number conversion. Accept a number in Decimal form and represent it in Hexadecimal, Octal and Binary. Change the Number and it reflects in other forms also
==> import java.util.*;

```
// ----- Subject -----
class NumberSubject {
    private int number;
    private List<Observer> observers = new ArrayList<>();

    public void attach(Observer o) {
        observers.add(o);
    }

    public void setNumber(int n) {
        number = n;
        notifyObservers();
    }

    public int getNumber() {
        return number;
    }

    private void notifyObservers() {
        for (Observer o : observers) {
            o.update();
        }
    }
}

// ----- Observer Interface -----
interface Observer {
    void update();
}

// ----- Concrete Observers -----
class HexObserver implements Observer {
    private NumberSubject subject;

    HexObserver(NumberSubject s) { subject = s; }

    public void update() {
        System.out.println("Hexadecimal: " + Integer.toHexString(subject.getNumber()));
```

```
    }
}

class OctalObserver implements Observer {
    private NumberSubject subject;

    OctalObserver(NumberSubject s) { subject = s; }

    public void update() {
        System.out.println("Octal: " + Integer.toOctalString(subject.getNumber()));
    }
}

class BinaryObserver implements Observer {
    private NumberSubject subject;

    BinaryObserver(NumberSubject s) { subject = s; }

    public void update() {
        System.out.println("Binary: " + Integer.toBinaryString(subject.getNumber()));
    }
}

// ----- Test Class -----
public class NumberObserverTest {
    public static void main(String[] args) {

        NumberSubject subject = new NumberSubject();

        new HexObserver(subject);
        new OctalObserver(subject);
        new BinaryObserver(subject);

        Scanner sc = new Scanner(System.in);

        System.out.print("Enter a decimal number: ");
        int n = sc.nextInt();

        subject.setNumber(n);

        System.out.print("\nEnter new number: ");
        n = sc.nextInt();

        subject.setNumber(n);
    }
}
```

☑ How to Perform (Steps)

Save as:

NumberObserverTest.java

Compile:

```
javac NumberObserverTest.java
```

Run:

```
java NumberObserverTest
```

** Q2) Write a program to connect with the available Wi-Fi using Arduino.
==> #include <ESP8266WiFi.h>

```
const char* ssid = "YourWiFiName";
const char* password = "YourWiFiPassword";

void setup() {
  Serial.begin(115200);
  WiFi.mode(WIFI_STA);

  Serial.print("Connecting to ");
  Serial.println(ssid);

  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.println("");
  Serial.println("WiFi Connected!");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}

void loop() {
}
```

☑ How to Perform (Steps)
1. Replace your Wi-Fi credentials
Change:
YourWiFiName
YourWiFiPassword
to your actual Wi-Fi name and password.
2. Open Arduino IDE
3. Select ESP8266 board
Tools → Board → NodeMCU 1.0 (ESP-12E Module)
4. Paste the program
5. Select COM port
Tools → Port → COMx
6. Upload
7. Open Serial Monitor
Set 115200 baud

*** SLIP 17 ***

** Q1) Write a Java Program to implement Abstract Factory Pattern for Shape interface.
==> // ----- Shape Interface -----

```
interface Shape {
    void draw();
```

```
}

// ----- Concrete Shapes -----
class Circle implements Shape {
    public void draw() {
        System.out.println("Drawing Circle");
    }
}

class Square implements Shape {
    public void draw() {
        System.out.println("Drawing Square");
    }
}

class Rectangle implements Shape {
    public void draw() {
        System.out.println("Drawing Rectangle");
    }
}

// ----- Abstract Factory -----
abstract class AbstractFactory {
    abstract Shape getShape(String shapeType);
}

// ----- Concrete Factory 1 -----
class RoundedShapeFactory extends AbstractFactory {
    Shape getShape(String shapeType) {
        if (shapeType.equalsIgnoreCase("circle")) return new Circle();
        if (shapeType.equalsIgnoreCase("square")) return new Square();
        return null;
    }
}

// ----- Concrete Factory 2 -----
class NormalShapeFactory extends AbstractFactory {
    Shape getShape(String shapeType) {
        if (shapeType.equalsIgnoreCase("rectangle")) return new Rectangle();
        return null;
    }
}

// ----- Factory Producer -----
class FactoryProducer {
    public static AbstractFactory getFactory(String choice) {
        if (choice.equalsIgnoreCase("rounded"))
            return new RoundedShapeFactory();
        else
            return new NormalShapeFactory();
    }
}

// ----- Test Class -----
```

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

      AbstractFactory roundedFactory = FactoryProducer.getFactory("rounded");
      Shape c = roundedFactory.getShape("circle");
      c.draw();

      Shape s = roundedFactory.getShape("square");
      s.draw();

      AbstractFactory normalFactory = FactoryProducer.getFactory("normal");
      Shape r = normalFactory.getShape("rectangle");
      r.draw();
   }
}
```

☑ How to Perform (Steps)
Save file as:
AbstractFactoryTest.java
Compile:
javac AbstractFactoryTest.java
Run:
java AbstractFactoryTest

** Q2) Write a program to get temperature notification using Arduino.
==>
```
int sensorPin = A0;   // LM35 output pin
int ledPin = 7;       // LED for alert
float temperature = 0;

void setup() {
  Serial.begin(9600);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  int reading = analogRead(sensorPin);

  // Convert LM35 reading to °C
  temperature = (reading * 5.0 * 100.0) / 1024;

  Serial.print("Temperature: ");
  Serial.print(temperature);
  Serial.println(" °C");

  // Notification when temperature is high
  if (temperature > 30) {
    Serial.println("TEMPERATURE ALERT!");
    digitalWrite(ledPin, HIGH);
    delay(300);
    digitalWrite(ledPin, LOW);
    delay(300);
  } else {
    digitalWrite(ledPin, LOW);
```

```
  }

  delay(1000);
}
```

☑ How to Perform (Steps)

1. Wiring LM35 Sensor

LM35 Pin1 → 5V
LM35 Pin2 → A0
LM35 Pin3 → GND

2. Wiring LED

LED + → Pin 7
LED – → GND (with resistor)

3. Upload

Open Arduino IDE
Paste code
Select Arduino UNO
Upload program

4. Test

Open Serial Monitor → 9600 baud
Heat LM35 slightly (finger or lighter far away)
When temperature > 30°C, LED blinks and you see:

*** SLIP 18***

** Q1) Write a JAVA Program to implement built-in support (java.util.Observable) Weather station with members temperature, humidity, pressure and methods mesurmentsChanged(), setMesurment(), getTemperature(), getHumidity(), getPressure()

==>

```
import java.util.Observable;
import java.util.Observer;

// ----- Subject -----
class WeatherData extends Observable {
    private float temperature;
    private float humidity;
    private float pressure;

    public float getTemperature() { return temperature; }
    public float getHumidity() { return humidity; }
    public float getPressure() { return pressure; }

    public void measurementsChanged() {
        setChanged();
        notifyObservers();
    }

    public void setMeasurement(float t, float h, float p) {
        this.temperature = t;
        this.humidity = h;
        this.pressure = p;
```

```
        measurementsChanged();
    }
}

// ----- Observer -----
class CurrentDisplay implements Observer {
    public void update(Observable o, Object arg) {
        WeatherData wd = (WeatherData) o;

        System.out.println("Temperature: " + wd.getTemperature());
        System.out.println("Humidity: " + wd.getHumidity());
        System.out.println("Pressure: " + wd.getPressure());
    }
}

// ----- Test Program -----
public class WeatherStationTest {
    public static void main(String[] args) {

        WeatherData wd = new WeatherData();
        CurrentDisplay display = new CurrentDisplay();

        wd.addObserver(display);

        wd.setMeasurement(30, 65, 1010);
        wd.setMeasurement(32, 70, 1008);
    }
}
```

☑ How to Perform (Steps)
Save file:
WeatherStationTest.java
Compile:
javac WeatherStationTest.java
Run:
java WeatherStationTest

** Q2) Write a program for LDR to vary the light intensity of LED using Arduino.
==>
```
int ldrPin = A0;     // LDR analog input
int ledPin = 9;      // PWM pin for LED
int ldrValue = 0;
int ledValue = 0;

void setup() {
  pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
}

void loop() {
  ldrValue = analogRead(ldrPin);            // Read LDR (0–1023)
  ledValue = map(ldrValue, 0, 1023, 255, 0);  // Convert to LED brightness

  analogWrite(ledPin, ledValue);            // Set LED brightness
```

```
  Serial.print("LDR Value: ");
  Serial.print(ldrValue);
  Serial.print("   LED Brightness: ");
  Serial.println(ledValue);

  delay(100);
}
```

☑ How to Perform (Steps)

1. LDR Circuit

One leg → 5V

Other leg → A0 and 10kΩ resistor → GND

2. LED Circuit

LED + → Pin 9 (PWM)

LED – → GND (through resistor)

3. Upload

Open Arduino IDE

Paste code

Select Arduino UNO

Upload

4. Test

Cover LDR → LED brightens

Shine light → LED dims

*** SLIP 19***

** Q1) Write a Java Program to implement Factory method for Pizza Store with createPizza(), orederPizza(), prepare(), Bake(), cut(), box(). Use this to create variety of pizza's like NyStyleCheesePizza, ChicagoStyleCheesePizza etc.

==>

```
// ----- Product -----
abstract class Pizza {
    String name;

    void prepare() {
        System.out.println("Preparing " + name);
    }

    void bake() {
        System.out.println("Baking " + name);
    }

    void cut() {
        System.out.println("Cutting " + name);
    }

    void box() {
        System.out.println("Boxing " + name);
    }
```

```
    public String getName() { return name; }
}

// ----- Concrete Products -----
class NYStyleCheesePizza extends Pizza {
    NYStyleCheesePizza() {
        name = "NY Style Cheese Pizza";
    }
}

class ChicagoStyleCheesePizza extends Pizza {
    ChicagoStyleCheesePizza() {
        name = "Chicago Style Cheese Pizza";
    }

    @Override
    void cut() {
        System.out.println("Cutting in square slices (Chicago Style)");
    }
}

// ----- Creator -----
abstract class PizzaStore {

    public Pizza orderPizza(String type) {
        Pizza pizza = createPizza(type);

        pizza.prepare();
        pizza.bake();
        pizza.cut();
        pizza.box();

        return pizza;
    }

    abstract Pizza createPizza(String type);
}

// ----- Concrete Creators -----
class NYPizzaStore extends PizzaStore {
    Pizza createPizza(String type) {
        if (type.equals("cheese"))
            return new NYStyleCheesePizza();
        return null;
    }
}

class ChicagoPizzaStore extends PizzaStore {
    Pizza createPizza(String type) {
        if (type.equals("cheese"))
            return new ChicagoStyleCheesePizza();
        return null;
    }
}
```

```
// ----- Test Class -----
public class FactoryPizzaTest {
    public static void main(String[] args) {

        PizzaStore nyStore = new NYPizzaStore();
        PizzaStore chicagoStore = new ChicagoPizzaStore();

        System.out.println("NY Order:");
        nyStore.orderPizza("cheese");

        System.out.println("\nChicago Order:");
        chicagoStore.orderPizza("cheese");
    }
}
```

☑ How to Perform (Steps)
Save file as:
FactoryPizzaTest.java
Compile:
javac FactoryPizzaTest.java
Run:
java FactoryPizzaTest

** Q2) Start Raspberry Pi and Execute various Linux commands in command terminal window: ls, cd, touch, mv, rm, man, mkdir, rmdir, tar, gzip, cat, more, less, ps, sudo, cron, chown, chgrp, pingetc.
==> ☑ Start Raspberry Pi
Power ON Raspberry Pi.
Login (if asked):
Username: pi
Password: raspberry

Click the Terminal icon.

☑ Run Linux Commands (Type these in Terminal)
1. ls – list files
ls
ls -l

2. cd – change directory
cd Desktop
cd /home/pi

3. touch – create a file
touch file1.txt

4. mv – rename/move file
mv file1.txt newfile.txt

5. rm – delete file
rm newfile.txt

6. man – help manual
man ls

7. mkdir – create folder
mkdir mydir

8. rmdir – remove folder
rmdir mydir

9. tar – archive files

Create tar:
tar -cvf data.tar mydir

Extract tar:
tar -xvf data.tar

10. gzip – compress file
gzip sample.txt
gunzip sample.txt.gz

11. cat – view file contents
cat sample.txt

12. more – page viewer
more sample.txt

13. less – better page viewer
less sample.txt

14. ps – list running processes
ps

15. sudo – run as root
sudo apt update

16. cron – schedule tasks
(Edit crontab)
crontab -e

17. chown – change file owner
sudo chown pi:pi sample.txt

18. chgrp – change file group
sudo chgrp pi sample.txt

19. ping – check network
ping google.com

***** SLIP 20 *****

**** Q1) Write a Java Program to implement I/O Decorator for converting uppercase letters to lower case letters.**

```
==> import java.io.*;

// Custom Decorator InputStream
class LowerCaseInputStream extends FilterInputStream {

    protected LowerCaseInputStream(InputStream in) {
        super(in);
    }

    @Override
    public int read() throws IOException {
        int c = super.read();
        return (c == -1 ? c : Character.toLowerCase((char)c));
    }

    @Override
    public int read(byte[] b, int off, int len) throws IOException {
        int result = super.read(b, off, len);
        for (int i = off; i < off + result; i++) {
            b[i] = (byte)Character.toLowerCase((char)b[i]);
        }
        return result;
    }
}

public class DecoratorTest {
    public static void main(String[] args) throws Exception {

        InputStream in = new LowerCaseInputStream(
                        new FileInputStream("input.txt"));

        int c;
        while ((c = in.read()) != -1) {
            System.out.print((char)c);
        }

        in.close();
    }
}
```

☑ How to Perform (Steps)
Create a text file named input.txt with uppercase text:
HELLO WORLD
THIS IS JAVA
Save program as:
DecoratorTest.java
Compile:
javac DecoratorTest.java
Run:
java DecoratorTest

** Q2) Write python programs on Pi like: a) Read your name and print Hello message with name b) Read two numbers and print their sum, difference, product and division. c) Word

and character count of a given string. d) Area of a given shape (rectangle, triangle and circle) reading shape and appropriate values from standard input.

==> ☑ a) Read your name and print Hello message

```
name = input("Enter your name: ")
print("Hello", name)
```

☑ b) Read two numbers and print sum, difference, product, division

```
a = float(input("Enter first number: "))
b = float(input("Enter second number: "))

print("Sum =", a + b)
print("Difference =", a - b)
print("Product =", a * b)

if b != 0:
    print("Division =", a / b)
else:
    print("Division not possible")
```

☑ c) Word and character count of a given string

```
s = input("Enter a string: ")

words = len(s.split())
chars = len(s)

print("Word Count =", words)
print("Character Count =", chars)
```

☑ d) Area of a shape (rectangle, triangle, circle)

```
shape = input("Enter shape (rectangle/triangle/circle): ")

if shape == "rectangle":
    l = float(input("Enter length: "))
    w = float(input("Enter width: "))
    print("Area =", l * w)

elif shape == "triangle":
    b = float(input("Enter base: "))
    h = float(input("Enter height: "))
    print("Area =", 0.5 * b * h)

elif shape == "circle":
    r = float(input("Enter radius: "))
    print("Area =", 3.14 * r * r)

else:
    print("Invalid shape")
```

☑ HOW TO RUN PYTHON PROGRAMS ON RASPBERRY PI

Step 1: Open Terminal

Click the Terminal icon on the top bar.

☑ Step 2: Create a Python file

Example:

For part (a), create a file named a.py:
In Terminal type:
nano a.py
This opens the editor.

☑ Step 3: Write your Python code
Example:
name = input("Enter your name: ")
print("Hello", name)

☑ Step 4: Save the file
Press:
CTRL + O
ENTER
CTRL + X
This saves and closes nano.

☑ Step 5: Run the Python program
Type:
python3 a.py