Practical 1

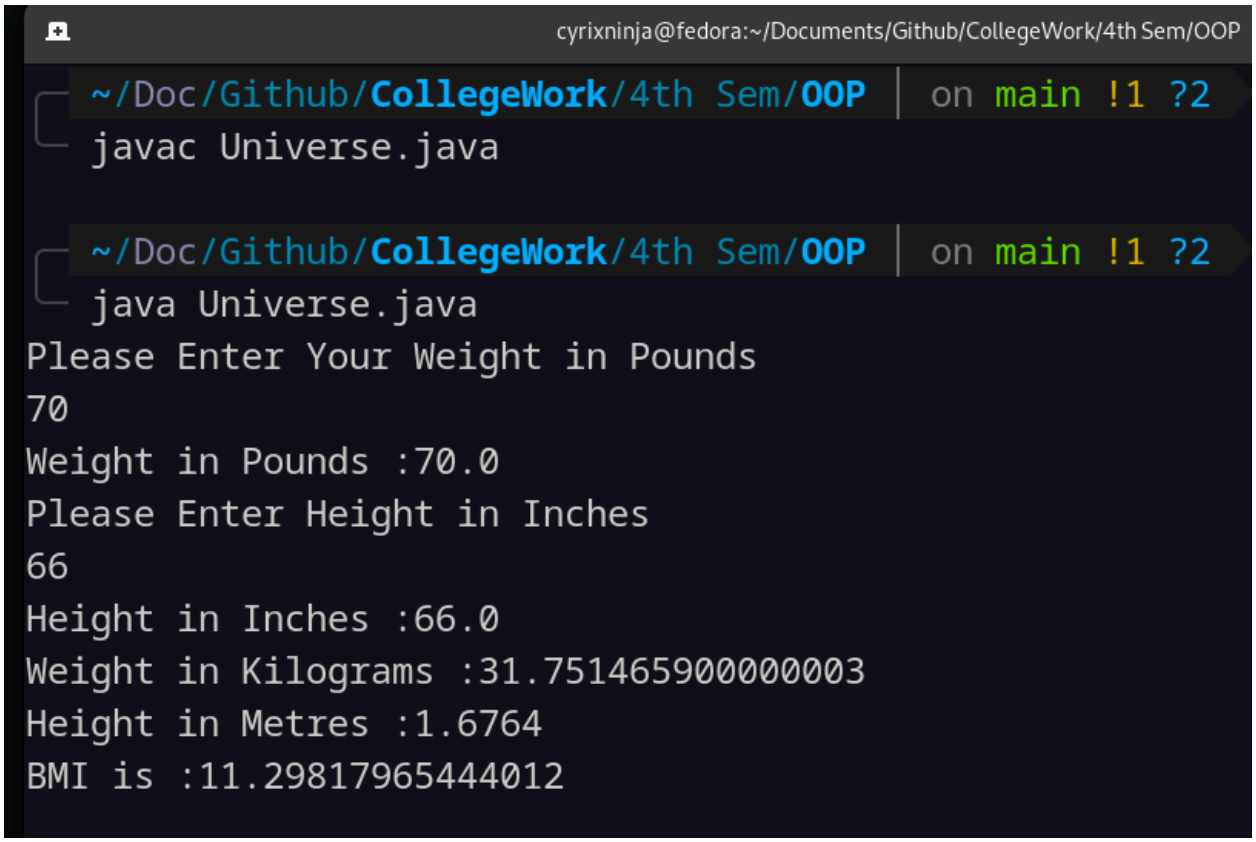
AIM: Body Mass Index (BMI) is a measure of health on weight. It can be calculated by taking your weight in kilograms and dividing by the square of your height in meters.
Write a program that prompts the user to enter a weight in pounds and height in inches and displays the BMI.
Note: - 1 pound=.45359237 Kg and 1 inch=.0254 meters.

COMPLETE CODE

```
// Import Libraries
import java.util.Scanner;

// Main Class
public class Universe {
	public static void main(String[] args) {
	double weight;
	System.out.println("Please Enter Your Weight in Pounds");
	Scanner weight_in_pounds = new Scanner(System.in);
	weight = weight_in_pounds.nextDouble();
	System.out.println("Weight in Pounds :" + weight);


	double height;
	System.out.println("Please Enter Height in Inches");
	Scanner height_in_inches = new Scanner(System.in);
	height = height_in_inches.nextDouble();
	System.out.println("Height in Inches :" + height);


	Double height_in_metres= height*0.0254;
	Double weight_in_kilogram = weight*0.45359237;
	System.out.println("Weight in Kilograms :" + weight_in_kilogram);
	System.out.println("Height in Metres :" + height_in_metres);
```

```
        Double bmi = (weight_in_kilogram)/(height_in_metres*height_in_metres);
        System.out.println("BMI is :"+bmi);

        // Closing Scanners
        weight_in_pounds.close();
        height_in_inches.close();
        }
}
```

OUTPUT:



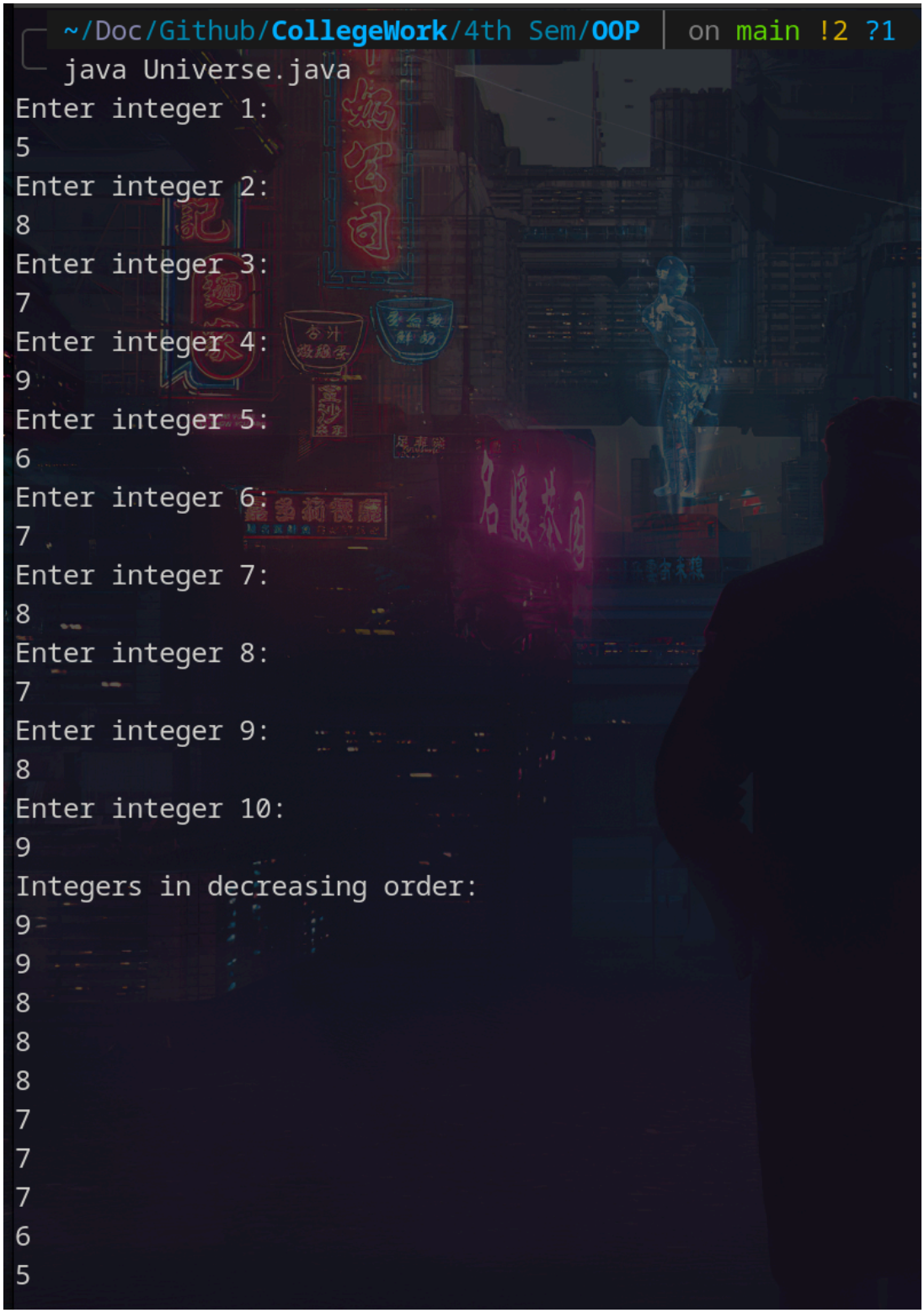
Practical 2

AIM: Write a program that prompts the user to enter 10 integers and display the integers in decreasing order using loop. Use the 'break'.

COMPLETE CODE

```
// Import the required classes
import java.util.Scanner;
import java.util.Arrays;
import java.util.Collections;

public class Universe {
	public static void main(String[] args) {
	// Create an array to store the integers
	Integer[] numbers = new Integer[10];
	// Create a scanner object to read input
	Scanner scanner = new Scanner(System.in);

	for (int i = 0; i < 10; i++) {
	System.out.println("Enter integer " + (i + 1) + ":");
	numbers[i] = scanner.nextInt();

	// Break the loop if 10 integers have been entered
	if (i == 9) {
	break;
	} }

	// Sort the array in decreasing order
	Arrays.sort(numbers, Collections.reverseOrder());
	System.out.println("Integers in decreasing order:");
	for (int number : numbers) {
	System.out.println(number); }
	scanner.close();
	} }
```

OUTPUT:


~/Doc/Github/CollegeWork/4th Sem/OOP | on main !2 ?1
java Universe.java
Enter integer 1:
5
Enter integer 2:
8
Enter integer 3:
7
Enter integer 4:
9
Enter integer 5:
6
Enter integer 6:
7
Enter integer 7:
8
Enter integer 8:
7
Enter integer 9:
8
Enter integer 10:
9
Integers in decreasing order:
9
9
8
8
8
7
7
7
6
5

Practical 3

AIM:

Write a program to demonstrate the following for String:

a. Input a string from standard input
b. Input a string from command line argument
c. Find the length of it.
d. Reverse a string
e. Copy a string to another string
f. Concatenate two strings
g. Extract some bytes from string
h. Get Substring
i. Check string starts and ends with particular string
j. Convert any data type object/variable to string
k. Split a string using regular expressions
l. Replace string with other
m. Find the indexes of a string in another string
n. Convert string to other types (byte and character array)
o. Convert into uppercase and lowercase
p. Check the equality of two strings (with and without consideration of case)
q. Print the hashcode of string

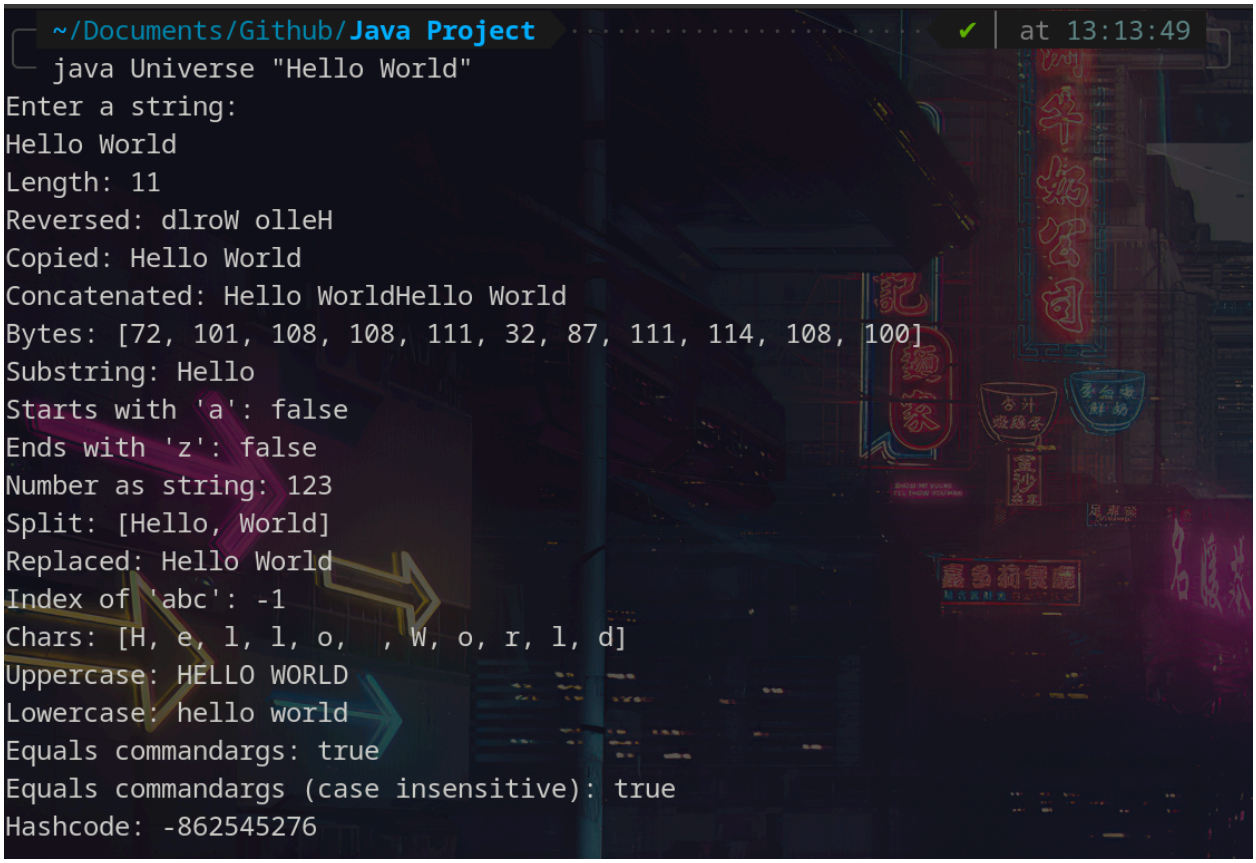
COMPLETE CODE

```
import java.util.Arrays;
import java.util.Scanner;

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


		// a. Input a string from standard input
		Scanner scanner = new Scanner(System.in);
		System.out.println("Enter a string:");
		String input = scanner.nextLine();
```

```
// b. Input a string from command line argument
String commandargs = args.length > 0 ? args[0] : "";

// c. Find the length of it.
System.out.println("Length: " + input.length());

// d. Reverse a string
String reversed = new StringBuilder(input).reverse().toString();
System.out.println("Reversed: " + reversed);

// e. Copy a string to another string
String copied = new String(input);
System.out.println("Copied: " + copied);

// f. Concatenate two strings
String concatenated = input + commandargs;
System.out.println("Concatenated: " + concatenated);

// g. Extract some bytes from string
byte[] bytes = input.getBytes();
System.out.println("Bytes: " + Arrays.toString(bytes));

// h. Get Substring
String substring = input.substring(0, input.length() / 2);
System.out.println("Substring: " + substring);

// i. Check string starts and ends with particular string
System.out.println("Starts with 'a': " + input.startsWith("a"));
System.out.println("Ends with 'z': " + input.endsWith("z"));

// j. Convert any data type object/variable to string
int number = 123;
String numberStr = Integer.toString(number);
System.out.println("Number as string: " + numberStr);

// k. Split a string using regular expressions
```

```
        String[] split = input.split("\\s+");
        System.out.println("Split: " + Arrays.toString(split));

        // l. Replace string with other
        String replaced = input.replace('a', 'b');
        System.out.println("Replaced: " + replaced);

        // m. Find the indexes of a string in another string
        int index = input.indexOf("abc");
        System.out.println("Index of 'abc': " + index);

        // n. Convert string to other types (byte and character array)
        char[] chars = input.toCharArray();
        System.out.println("Chars: " + Arrays.toString(chars));

        // o. Convert into uppercase and lowercase
        System.out.println("Uppercase: " + input.toUpperCase());
        System.out.println("Lowercase: " + input.toLowerCase());

        // p. Check the equality of two strings (with and without consideration of case)
        System.out.println("Equals commandargs: " + input.equals(commandargs));
        System.out.println("Equals commandargs (case insensitive): " +
input.equalsIgnoreCase(commandargs));

        // q. Print the hashcode of string
        System.out.println("Hashcode: " + input.hashCode());
        }
}
```

OUTPUT:


~/Documents/Github/Java Project
at 13:13:49
java Universe "Hello World"
Enter a string:
Hello World
Length: 11
Reversed: dlroW olleH
Copied: Hello World
Concatenated: Hello WorldHello World
Bytes: [72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100]
Substring: Hello
Starts with 'a': false
Ends with 'z': false
Number as string: 123
Split: [Hello, World]
Replaced: Hello World
Index of 'abc': -1
Chars: [H, e, l, l, o, , W, o, r, l, d]
Uppercase: HELLO WORLD
Lowercase: hello world
Equals commandargs: true
Equals commandargs (case insensitive): true
Hashcode: -862545276

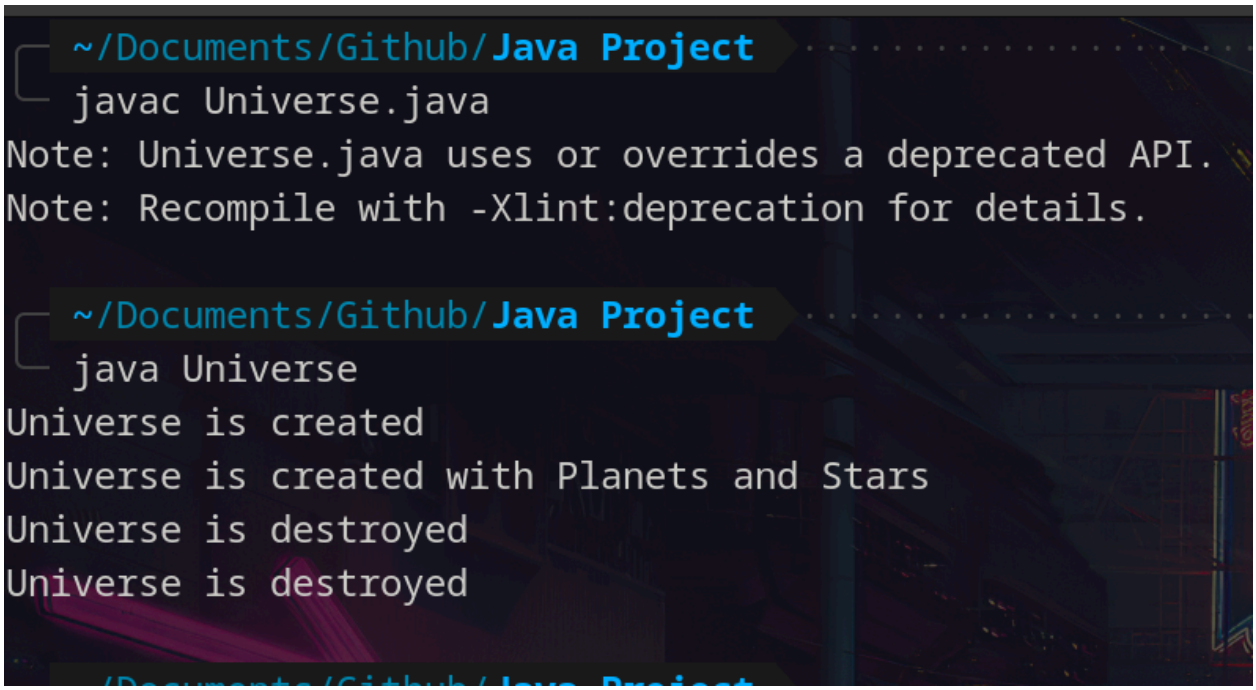
Practical 4

AIM: Write a program to demonstrate the use of constructor (with and without parameter) and destructor.

COMPLETE CODE

```
class Universe {
        Universe() {
        System.out.println("Universe is created");
        }

        Universe(String s) {
        System.out.println("Universe is created with " + s);
        }

        public void finalize() {
        System.out.println("Universe is destroyed");
        }

        public static void main(String[] args) {
        Universe u1 = new Universe();
        Universe u2 = new Universe("Planets and Stars");
        u1 = null;
        u2 = null;
        System.gc();
        }
}
```

OUTPUT:


~/Documents/Github/Java Project
javac Universe.java
Note: Universe.java uses or overrides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
~/Documents/Github/Java Project
java Universe
Universe is created
Universe is created with Planets and Stars
Universe is destroyed
Universe is destroyed

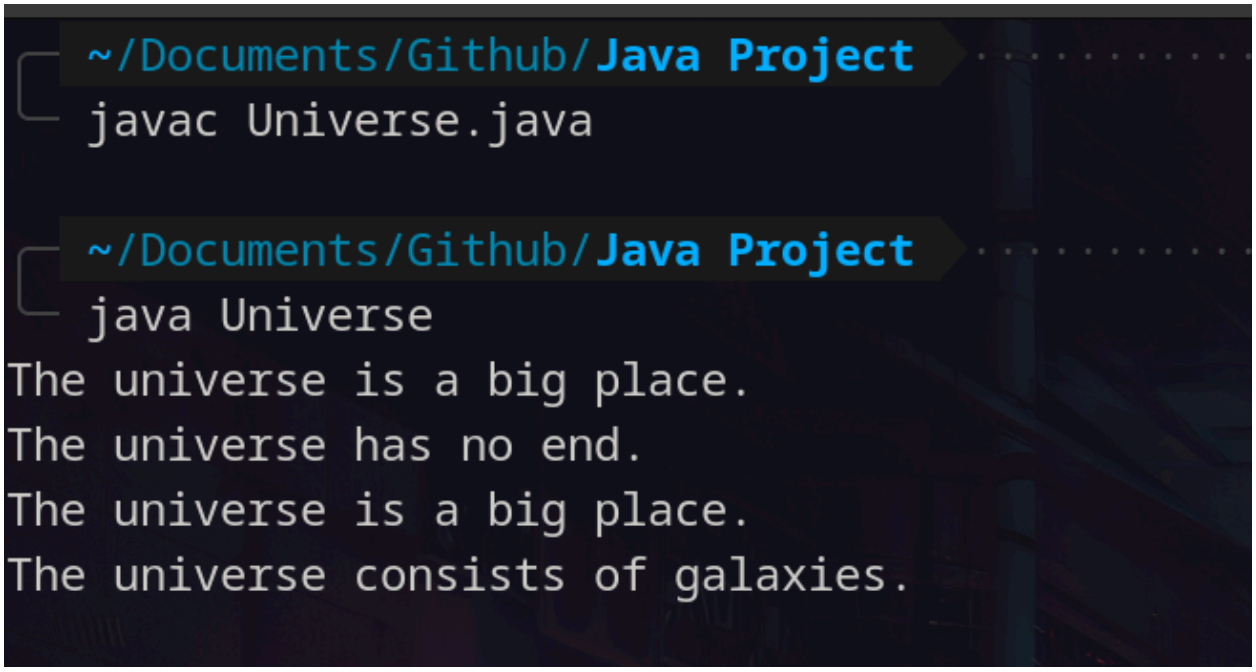
Practical 5

AIM: Write a program to demonstrate method overloading and constructor overloading.

COMPLETE CODE

```
class Universe {
	// Constructor overloading

	Universe() {
	System.out.println("The universe is a big place.");
	}

	Universe(String s) {
	System.out.println(s);
	}

	// Method overloading

	void display() {
	System.out.println("The universe is a big place.");
	}

	void display(String s) {
	System.out.println(s);
	}

	public static void main(String[] args) {
	Universe u = new Universe();
	Universe u1 = new Universe("The universe has no end.");
	u.display();
	u.display("The universe consists of galaxies.");
	}
}
```

OUTPUT:


~/Documents/Github/Java Project
javac Universe.java
~/Documents/Github/Java Project
java Universe
The universe is a big place.
The universe has no end.
The universe is a big place.
The universe consists of galaxies.

Practical 6

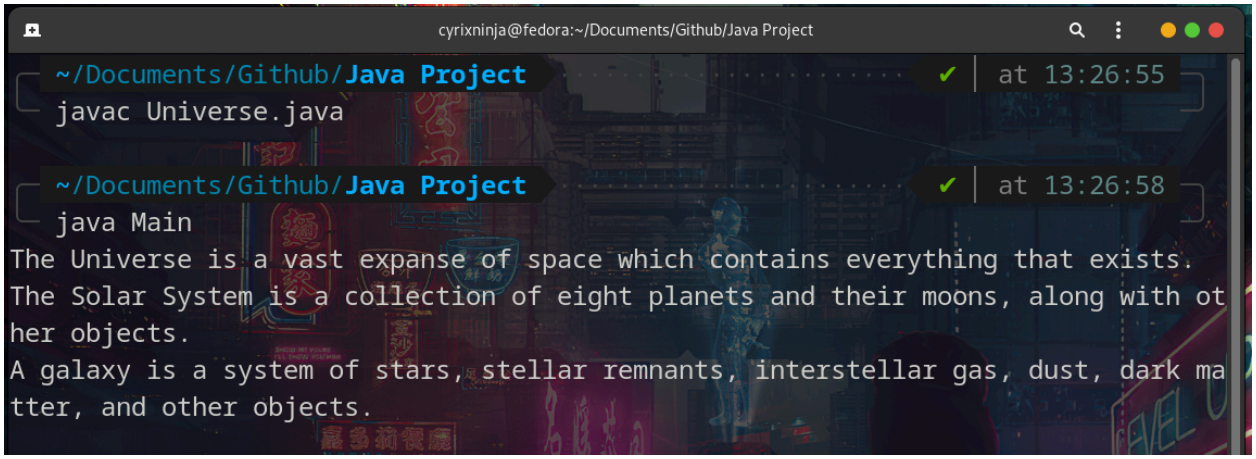
AIM: Write a program to demonstrate method overriding.

COMPLETE CODE

```
class Universe {
	void display() {
	System.out.println("The Universe is a vast expanse of space which contains everything that exists.");
	}
}

// Inheritance
class SolarSystem extends Universe {
	void display() {
	System.out.println("The Solar System is a collection of eight planets and their moons, along with other objects.");
	}
}

// Inheritance
class Galaxy extends Universe {
	void display() {
	System.out.println("A galaxy is a system of stars, stellar remnants, interstellar gas, dust, dark matter, and other objects.");
	}
}
```

```
class Main {
        public static void main(String[] args) {
        Universe universe = new Universe();
        universe.display();

        SolarSystem solarSystem = new SolarSystem();
        solarSystem.display();

        Galaxy galaxy = new Galaxy();
        galaxy.display();
        }
}
```

OUTPUT:



Practical 7

AIM: Write a program to demonstrate the use of following types of inheritance:

1. Single Inheritance
2. Multilevel Inheritance
3. Multiple Inheritance
4. Hierarchical Inheritance
5. Hybrid Inheritance

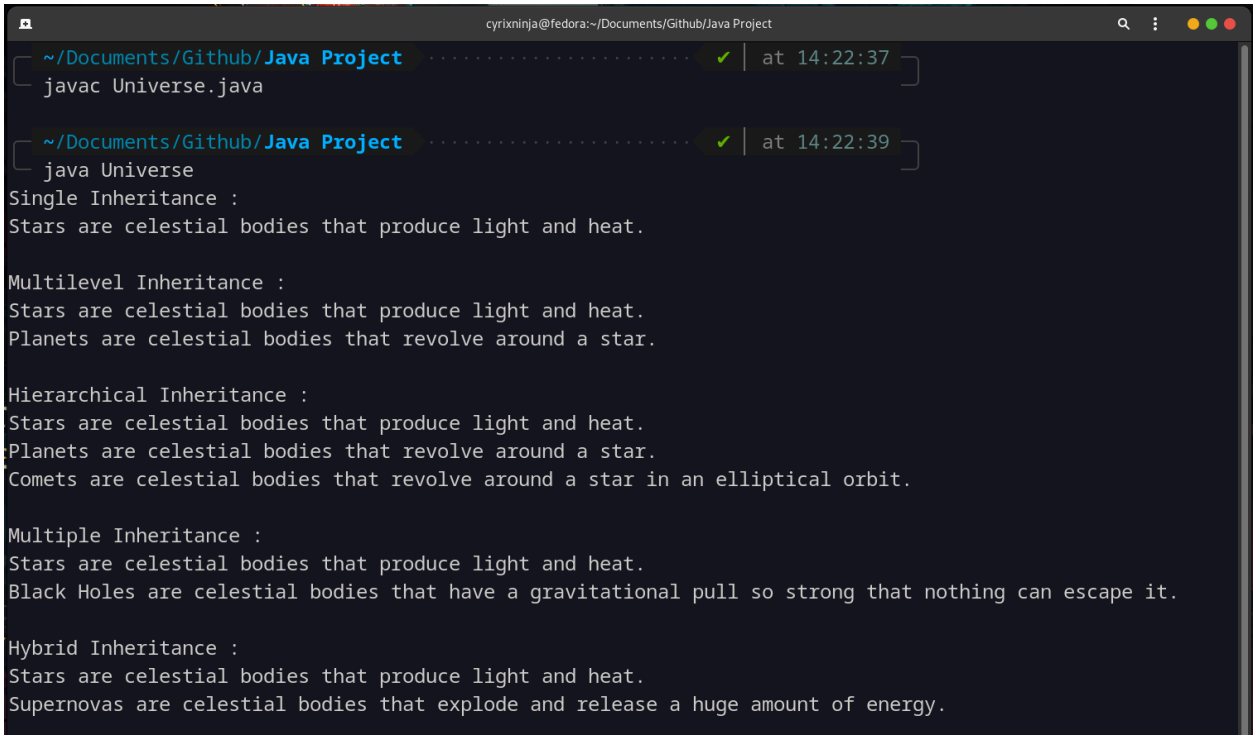
COMPLETE CODE

```
interface CelestialBody {
        void display();
}

class Stars implements CelestialBody {
        public void display() {
        System.out.println("Stars are celestial bodies that produce light and heat.");
        }
}

class Planets extends Stars {
        public void display() {
        super.display();
        System.out.println("Planets are celestial bodies that revolve around a star.");
        }
}

class Comets extends Planets {
        public void display() {
        super.display();
        System.out.println("Comets are celestial bodies that revolve around a star in an elliptical orbit.");
        }
}
```

```
interface BlackHoleProperties {
        default void displayBlackHoleProperties() {
        System.out.println("Black Holes are celestial bodies that have a gravitational pull so strong that nothing can escape it.");
        }
}

class BlackHoles extends Stars implements BlackHoleProperties {
        public void display() {
        super.display();
        displayBlackHoleProperties();
        }
}

class Supernovas extends Stars {
        public void display() {
        super.display();
        System.out.println("Supernovas are celestial bodies that explode and release a huge amount of energy.");
        }
}

public class Universe {
        public static void main(String[] args) {
        // Single Inheritance
        Stars stars = new Stars();
        stars.display();

        // Multilevel Inheritance
        Planets planets = new Planets();
        planets.display();

        // Hierarchical Inheritance
        Comets comets = new Comets();
        comets.display();
```

```
        // Multiple Inheritance (Simulated through Interface as Java does not support multiple inheritance)
        BlackHoles blackHoles = new BlackHoles();
        blackHoles.display();

        // Hybrid Inheritance
        Supernovas supernovas = new Supernovas();
        supernovas.display();
        }
}
```

OUTPUT:



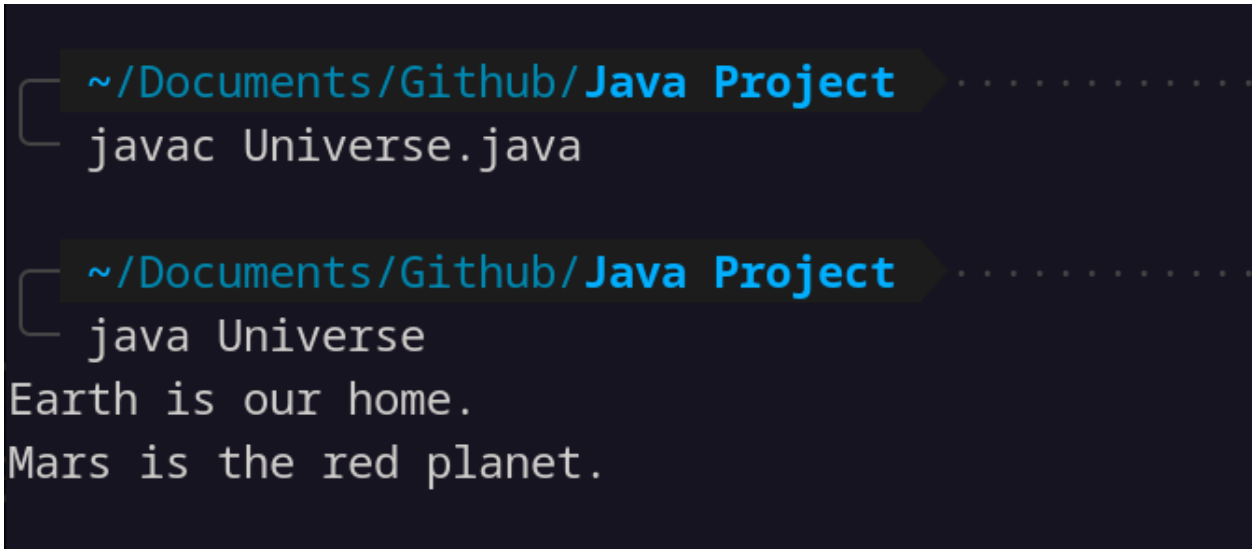
Practical 8

AIM: Demonstrate the use of “interface” using a program.

COMPLETE CODE

```
interface SolarSystem {
        void execute();
}

class Earth implements SolarSystem {
        public void execute() {
        System.out.println("Earth is our home.");
        }
}

class Mars implements SolarSystem {
        public void execute() {
        System.out.println("Mars is the red planet.");
        }
}

class Universe{
        public static void main(String[] args) {
        SolarSystem u = new Earth();
        u.execute();
        u = new Mars();
        u.execute();
        }
}
```

OUTPUT:


~/Documents/Github/Java Project
javac Universe.java
~/Documents/Github/Java Project
java Universe
Earth is our home.
Mars is the red planet.

Practical 9

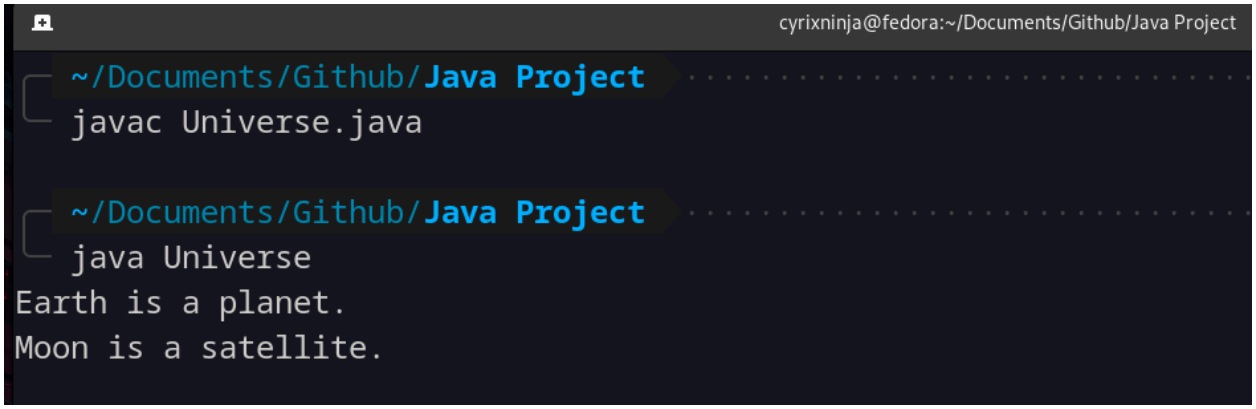
AIM: Demonstrate abstract class using the program.

COMPLETE CODE

// Abstract class is a class that is declared using the abstract keyword. It can have abstract methods(methods without a body) as well as concrete methods(methods with a body). It can't be instantiated, but can be subclassed. It can have constructors and static methods also.

```
abstract class SolarSystem {
        abstract void execute();
}

class Earth extends SolarSystem {
        void execute() {
        System.out.println("Earth is a planet.");
        }
}

class Moon extends SolarSystem {
        void execute() {
        System.out.println("Moon is a satellite.");
        }
}

class Universe {
        public static void main(String[] args) {
        Earth e = new Earth();
        e.execute();
        Moon m = new Moon();
        m.execute();
        }
}
```

OUTPUT:


cyrixninja@fedora:~/Documents/Github/Java Project
~/Documents/Github/Java Project
javac Universe.java
~/Documents/Github/Java Project
java Universe
Earth is a planet.
Moon is a satellite.

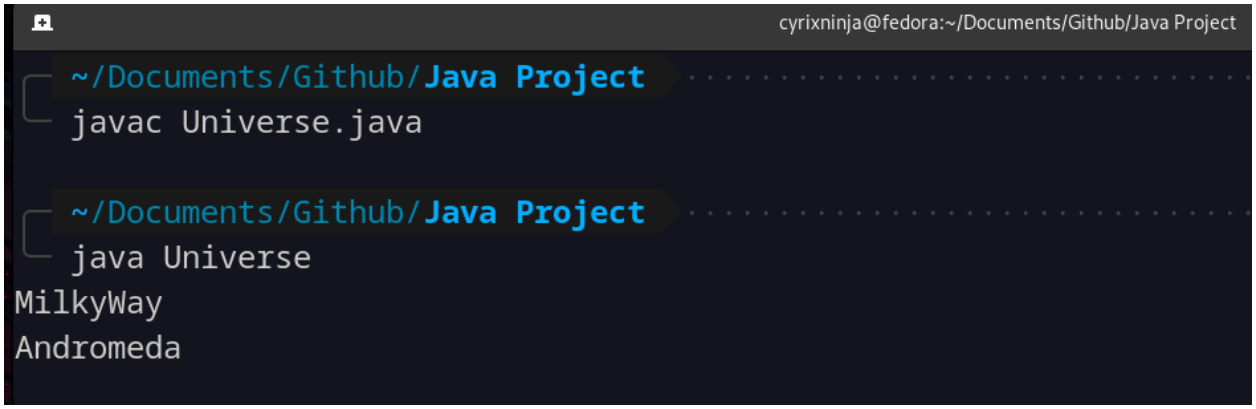
Practical 10

AIM: Write a program to demonstrate static and dynamic binding of objects.

COMPLETE CODE

```
// Static binding: The binding which can be resolved at compile time by the compiler is known as static or early binding.
// Dynamic binding: The binding which can be resolved at runtime is known as dynamic or late binding.

class Galaxy {
        void show() {
        System.out.println("Galaxy");
        }
}

class MilkyWay extends Galaxy {
        void show() {
        System.out.println("MilkyWay");
        }
}

class Andromeda extends Galaxy {
        void show() {
        System.out.println("Andromeda");
        }
}
class Universe {
        public static void main(String[] args) {
        // Static binding
        Galaxy g = new MilkyWay();
        g.show();
        // Dynamic binding
        g = new Andromeda();
        g.show();
        }}
```

OUTPUT:


cyrixninja@fedora:~/Documents/Github/Java Project
~/Documents/Github/Java Project
javac Universe.java
~/Documents/Github/Java Project
java Universe
MilkyWay
Andromeda

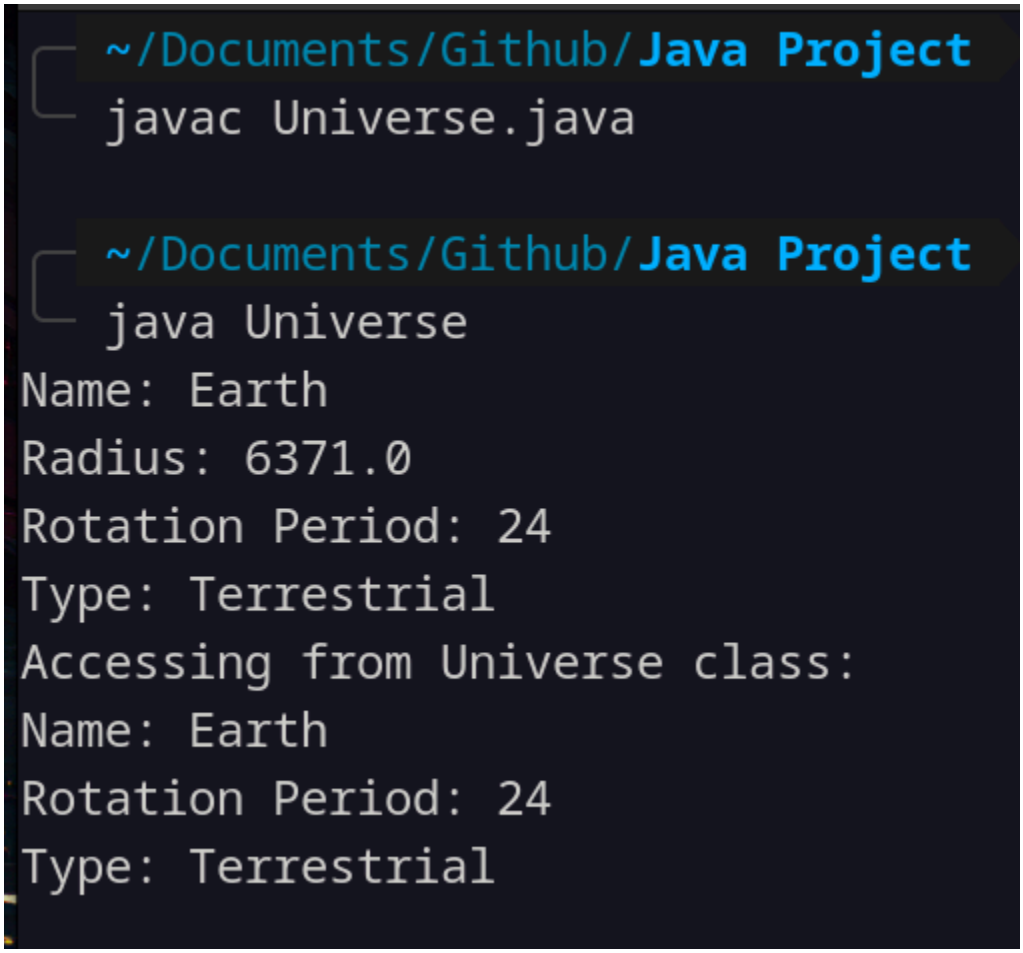
Practical 11

AIM: Write a program to demonstrate the visibility modifiers in java.

COMPLETE CODE

```
// The visibility modifiers are public, private, protected and default.
class Planet {
	public String name = "Earth"; // Can be accessed from any class
	private double radius = 6371; // Can only be accessed within this class
	protected int rotationPeriod = 24; // Can be accessed within this class, within this package, and in subclasses
	String type = "Terrestrial"; // Can be accessed within this class and within this package (default access)

	public void display() {
	System.out.println("Name: " + name);
	System.out.println("Radius: " + radius);
	System.out.println("Rotation Period: " + rotationPeriod);
	System.out.println("Type: " + type);
	}
}

public class Universe {
	public static void main(String[] args) {
	Planet earth = new Planet();
	earth.display();

	System.out.println("Accessing from Universe class:");
	System.out.println("Name: " + earth.name);
	// System.out.println("Radius: " + earth.radius); // This will give a compile error
	System.out.println("Rotation Period: " + earth.rotationPeriod);
	System.out.println("Type: " + earth.type);
	}}
```

OUTPUT:


~/Documents/Github/Java Project
javac Universe.java
~/Documents/Github/Java Project
java Universe
Name: Earth
Radius: 6371.0
Rotation Period: 24
Type: Terrestrial
Accessing from Universe class:
Name: Earth
Rotation Period: 24
Type: Terrestrial

Practical 12

AIM: Write a program to demonstrate try, catch, finally-exception handling.

COMPLETE CODE

```
import java.util.Scanner;

public class Universe {
	public static void main(String[] args) {
	Scanner sc = new Scanner(System.in);
	System.out.println("Enter the value of a and b: ");
	int a = sc.nextInt();
	int b = sc.nextInt();
	try {
	int c = a / b;
	System.out.println("The value of c is: " + c);
	} catch (ArithmeticException e) {
	System.out.println("The value of b cannot be zero.");
	} finally {
	System.out.println("The program has been executed successfully.");
	}
	}
}
```

OUTPUT:


~/Documents/Github/Java Project
java Universe
Enter the value of a and b:
10 5
The value of c is: 2
The program has been executed successfully.
~/Documents/Github/Java Project
java Universe
Enter the value of a and b:
0 6
The value of c is: 0
The program has been executed successfully.
~/Documents/Github/Java Project
java Universe
Enter the value of a and b:
6 0
The value of b cannot be zero.
The program has been executed successfully.

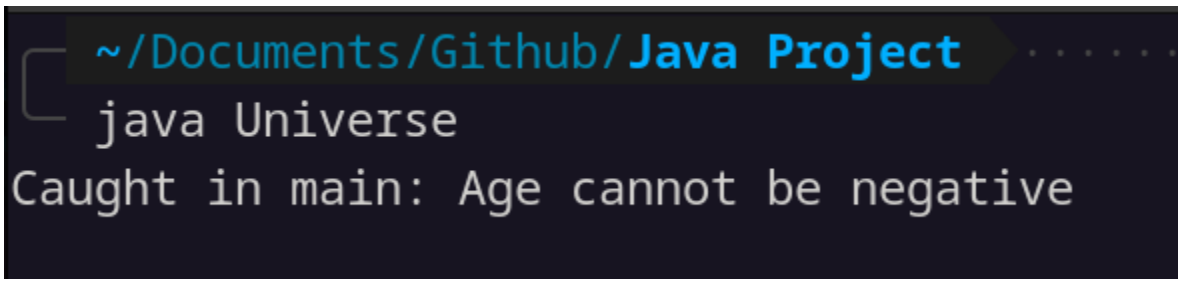
Practical 13

AIM: Write a program to demonstrate the use of throws.

COMPLETE CODE

```
class Universe {
        void ageCheck(int age) throws ArithmeticException {
        if (age < 0) {
        throw new ArithmeticException("Age cannot be negative");
        } else {
        System.out.println("Age is " + age);
        }
        }
        public static void main(String args[]) {
        try {
        Universe universe = new Universe();
        universe.ageCheck(-5);
        } catch (ArithmeticException e) {
        System.out.println("Caught in main: " + e.getMessage());
        }
        }
}
```

OUTPUT:



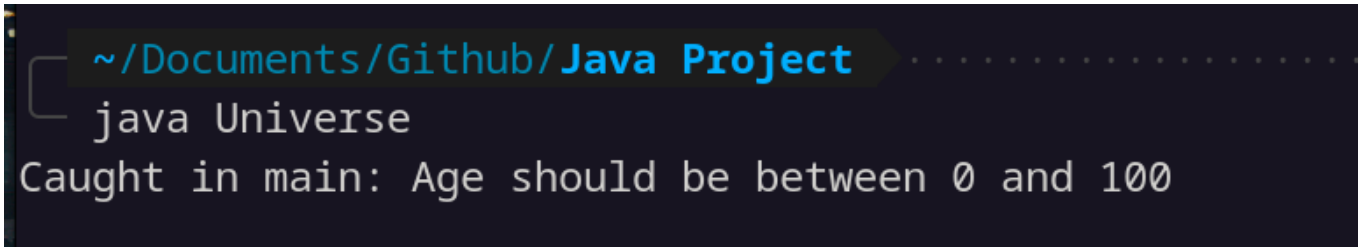
Practical 14

AIM: Write a program to demonstrate user defined exceptions.

COMPLETE CODE

```
class AgeOutOfRangeException extends Exception {
        public AgeOutOfRangeException(String message) {
        super(message);
        }
}

class Universe {
        void ageCheck(int age) throws AgeOutOfRangeException {
        if (age < 0 || age > 100) {
        throw new AgeOutOfRangeException("Age should be between 0 and 100");
        } else {
        System.out.println("Age is " + age);
        }
        }

        public static void main(String args[]) {
        try {
        Universe universe = new Universe();
        universe.ageCheck(150);
        } catch (AgeOutOfRangeException e) {
        System.out.println("Caught in main: " + e.getMessage());
        }
        }
}
```

OUTPUT:


~/Documents/Github/Java Project
java Universe
Caught in main: Age should be between 0 and 100

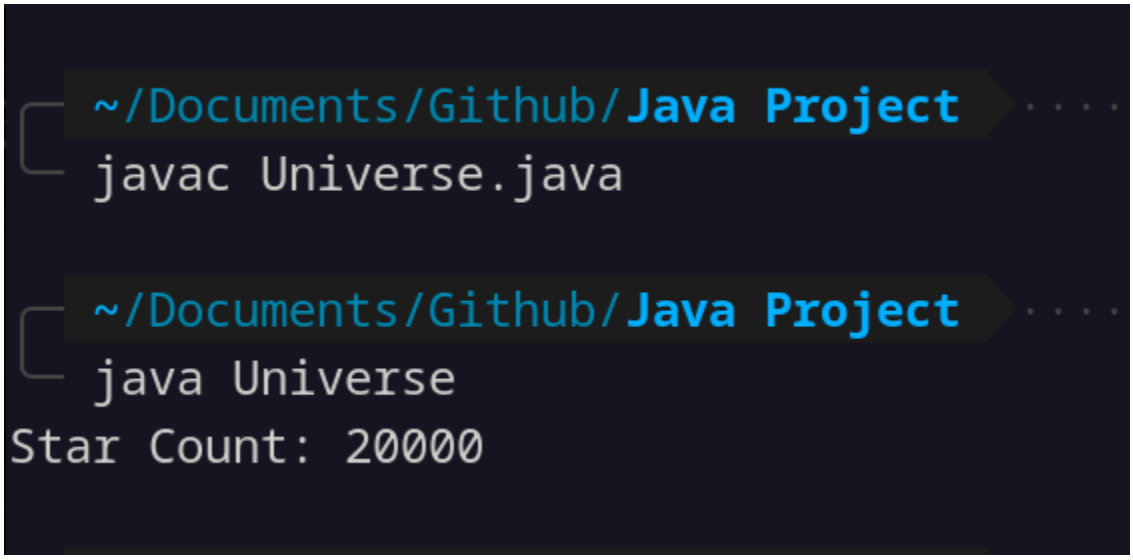
Practical 15

AIM: Demonstrate all the ways to use multithreading and synchronize methods using suitable programs

COMPLETE CODE : 1. Using Thread Class

```
class Universe extends Thread {
        private int starCount = 0;

        public synchronized void incrementStarCount() {
        starCount++;
        }

        public void run() {
        for (int i = 0; i < 10000; i++) {
        incrementStarCount();
        }
        }

        public static void main(String[] args) throws InterruptedException {
        Universe universe1 = new Universe();
        Universe universe2 = new Universe();
        universe1.start();
        universe2.start();
        universe1.join();
        universe2.join();
        System.out.println("Star Count: " + (universe1.starCount + universe2.starCount));
        }
}
```

OUTPUT:

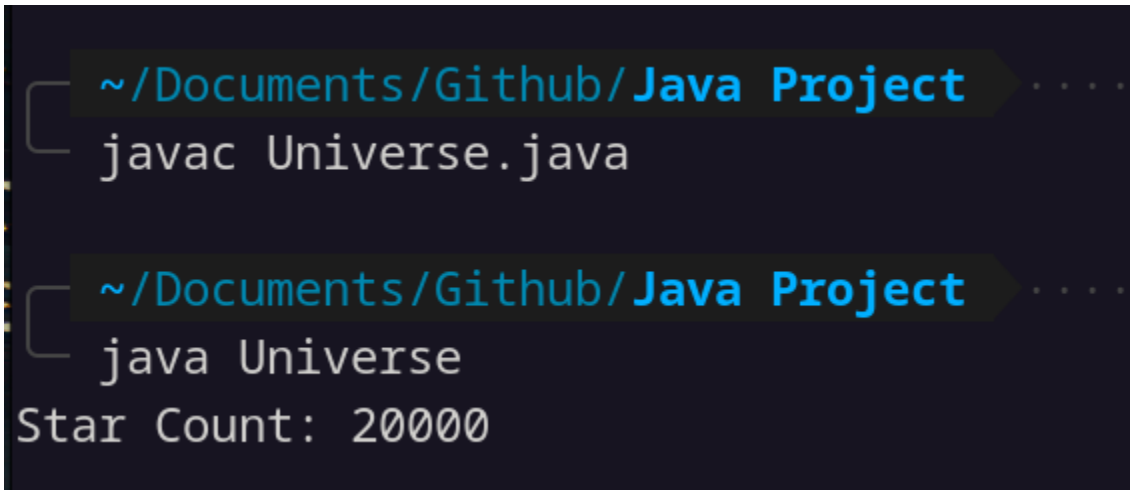


COMPLETE CODE : 2. Using a Runnable Interface

```
class Universe implements Runnable {
        private int starCount = 0;

        public synchronized void incrementStarCount() {
        starCount++;
        }

        public void run() {
        for (int i = 0; i < 10000; i++) {
        incrementStarCount();
        }
        }

        public static void main(String[] args) throws InterruptedException {
        Universe universe = new Universe();
        Thread thread1 = new Thread(universe);
        Thread thread2 = new Thread(universe);
        thread1.start();
        thread2.start();
        thread1.join();
```

```
        thread2.join();
        System.out.println("Star Count: " + universe.starCount);
        }
}
```

OUTPUT:

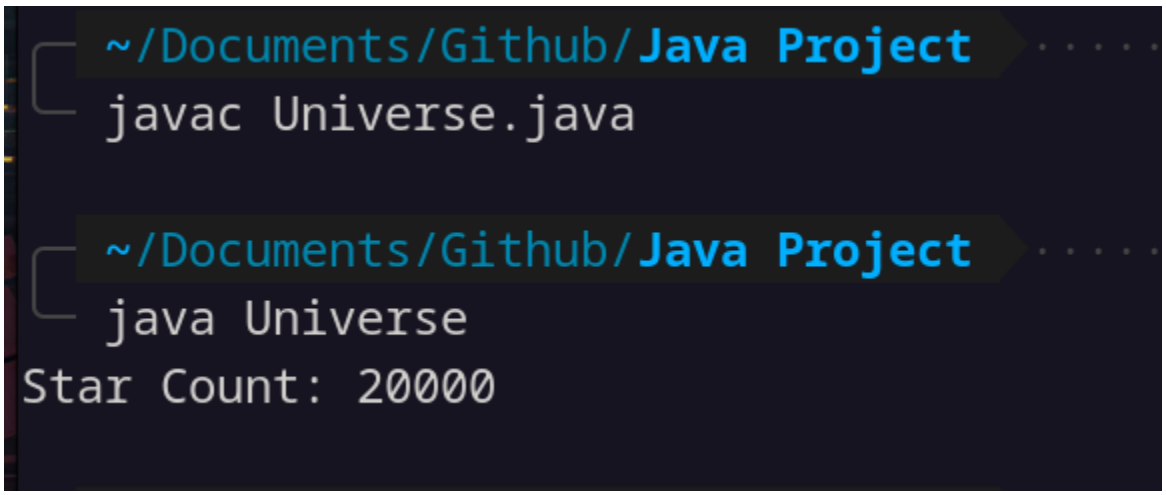


COMPLETE CODE : 3. Using the ExecuterService

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

class Universe {
        private int starCount = 0;

        public synchronized void incrementStarCount() {
        starCount++;
        }

        public void createStars() {
        for (int i = 0; i < 10000; i++) {
        incrementStarCount();
        }
        }
```

```
        public static void main(String[] args) throws InterruptedException {
        Universe universe = new Universe();
        ExecutorService executorService = Executors.newFixedThreadPool(2);
        executorService.submit(universe::createStars);
        executorService.submit(universe::createStars);
        executorService.shutdown();
        while (!executorService.isTerminated()) {
        }
        System.out.println("Star Count: " + universe.starCount);
        }
}
```

OUTPUT:



Practical 16

AIM: Write a program to demonstrate file properties using java.io.File.

COMPLETE CODE

```
import java.io.File;
import java.util.Scanner;

class Universe {
	public static void main(String[] args) {
	Scanner sc = new Scanner(System.in);
	System.out.println("Enter the file name: ");
	String fileName = sc.nextLine();
	File file = new File(fileName);
	if (file.exists()) {
	System.out.println("File name: " + file.getName());
	System.out.println("Absolute path: " + file.getAbsolutePath());
	System.out.println("Size: " + file.length() + " bytes");
	System.out.println("Readable: " + file.canRead());
	System.out.println("Writable: " + file.canWrite());
	System.out.println("Executable: " + file.canExecute());
	} else {
	System.out.println("File does not exist.");
	}
	sc.close();
	}
}
```

OUTPUT:

```
~/Documents/Github/Java Project
javac Universe.java

~/Documents/Github/Java Project
java Universe
Enter the file name:
hello_world.sh
File name: hello_world.sh
Absolute path: /home/cyrixninja/Documents/Github/Java Project/hello_world.sh
Size: 19 bytes
Readable: true
Writable: true
Executable: false
```

Practical 17

AIM: Write a program to demonstrate list and stack operations.

COMPLETE CODE

```
import java.util.*;

class Universe {
	public static void main(String[] args) {
	List<String> planets = new ArrayList<String>();
	planets.add("Mercury");
	planets.add("Venus");
	planets.add("Earth");
	planets.add("Mars");
	planets.add("Jupiter");
	planets.add("Saturn");
	planets.add("Uranus");
	planets.add("Neptune");
	planets.add("Pluto");

	System.out.println("Planets in the Solar System (List): " + planets);

	Stack<String> stack = new Stack<String>();
	stack.push("Mercury");
	stack.push("Venus");
	stack.push("Earth");
	stack.push("Mars");
	stack.push("Jupiter");
	stack.push("Saturn");
	stack.push("Uranus");
	stack.push("Neptune");
	stack.push("Pluto");

	System.out.println("Planets in the Solar System (Stack): " + stack);
```

```
        // List operations
        System.out.println("Number of planets in the Solar System: " + planets.size());
        System.out.println("Is Earth a planet? " + planets.contains("Earth"));
        System.out.println("Index of Mars: " + planets.indexOf("Mars"));
        System.out.println("Get planet at index 3: " + planets.get(3));
        System.out.println("Remove Pluto from the list");
        planets.remove("Pluto");
        System.out.println("Updated list of planets: " + planets);

        // Stack operations
        System.out.println("Top element of the stack: " + stack.peek());
        System.out.println("Pop element from the stack: " + stack.pop());
        System.out.println("Updated stack of planets: " + stack);
        }
}
```

OUTPUT:

```
~/Documents/Github/Java Project ✔ at 16:50:02
javac Universe.java

~/Documents/Github/Java Project ✔ at 16:50:04
java Universe
Planets in the Solar System (List): [Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto]
Planets in the Solar System (Stack): [Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune, Pluto
]
Number of planets in the Solar System: 9
Is Earth a planet? true
Index of Mars: 3
Get planet at index 3: Mars
Remove Pluto from the list
Updated list of planets: [Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune]
Top element of the stack: Pluto
Pop element from the stack: Pluto
Updated stack of planets: [Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune]
```

Practical 18

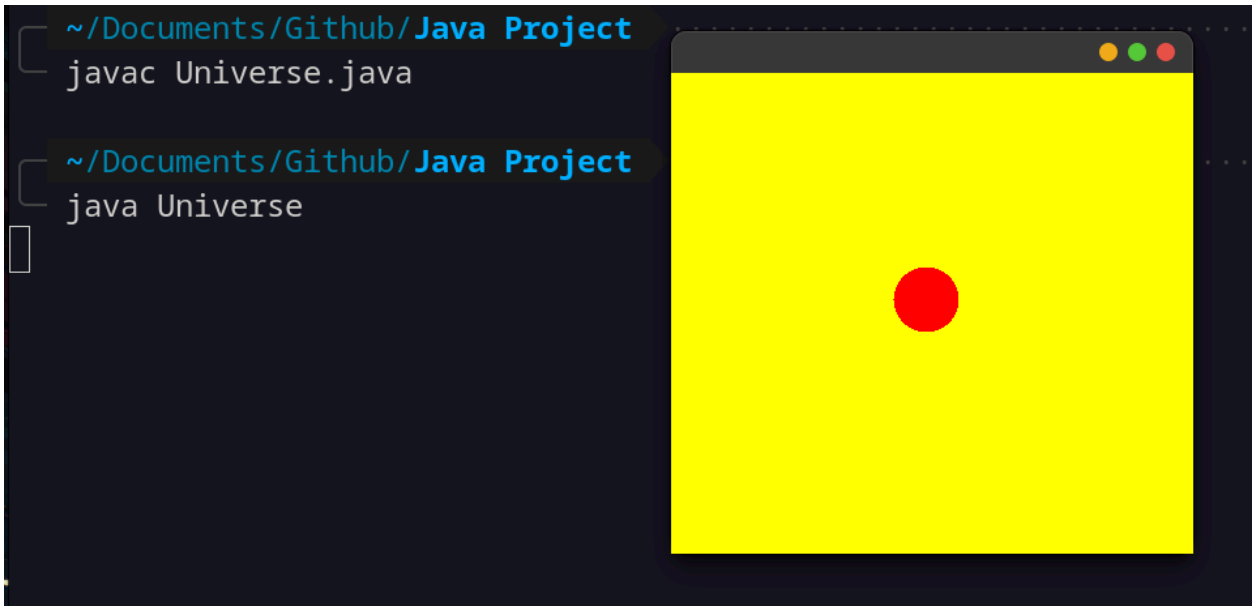
AIM: Write a program that moves a circle up, down, left or right using arrow keys.

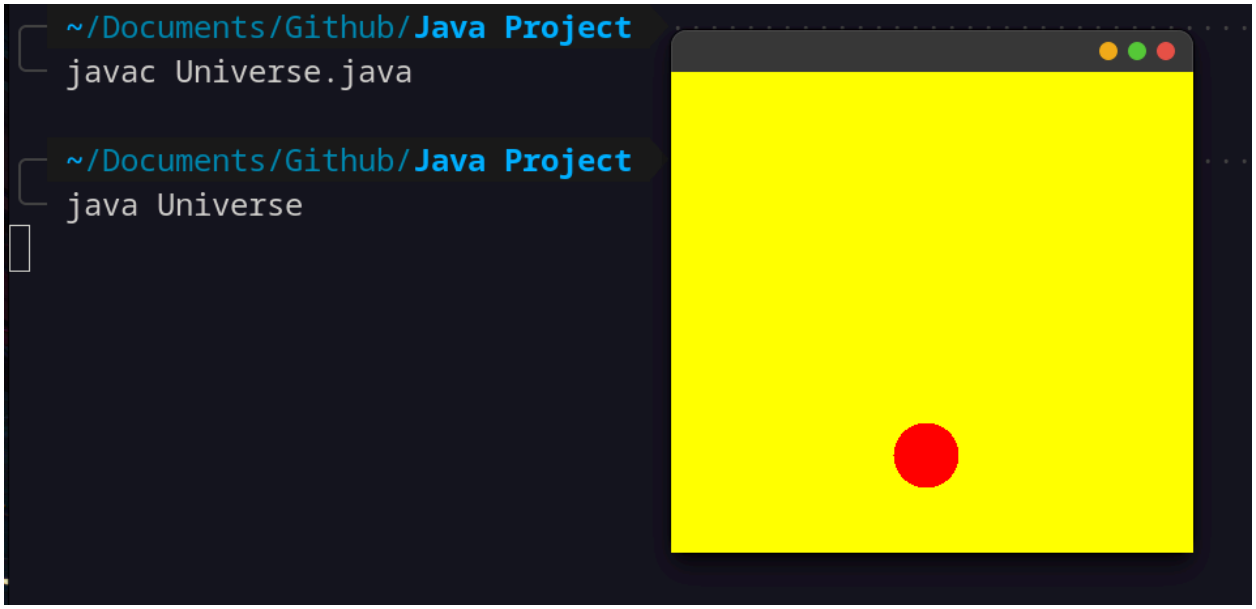
COMPLETE CODE

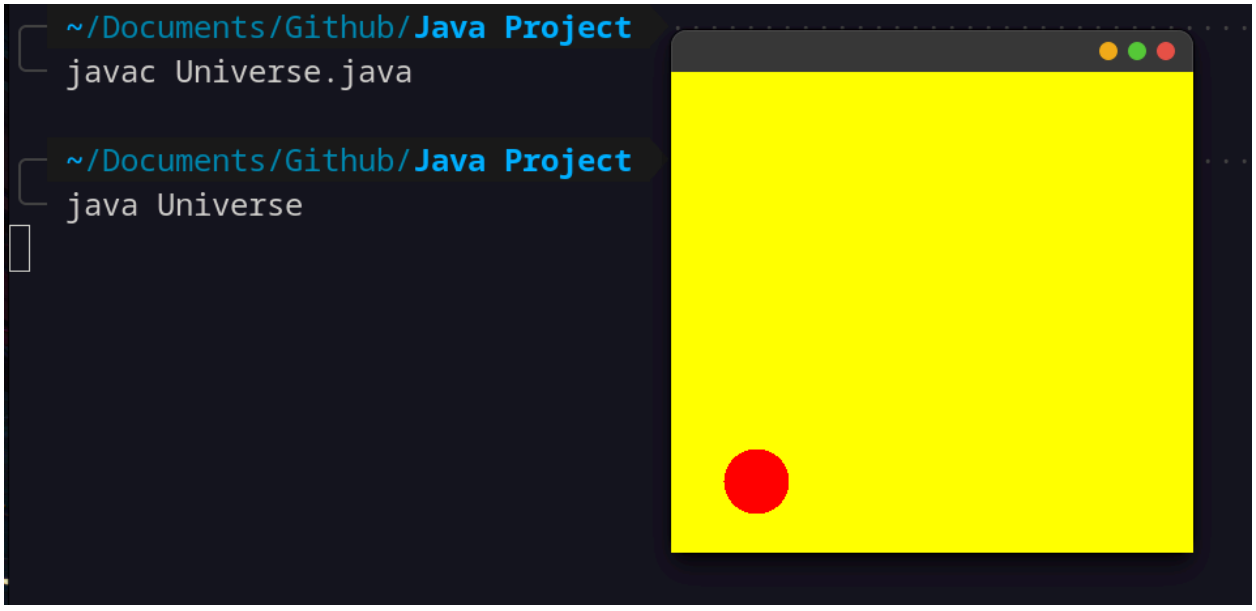
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Universe extends JFrame {
	private int x = 150;
	private int y = 150;

	public Universe() {
	addKeyListener(new KeyAdapter() {
	public void keyPressed(KeyEvent e) {
	if (e.getKeyCode() == KeyEvent.VK_UP) {
		y -= 10;
	} else if (e.getKeyCode() == KeyEvent.VK_DOWN) {
		y += 10;
	} else if (e.getKeyCode() == KeyEvent.VK_LEFT) {
		x -= 10;
	} else if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
		x += 10;
	}
	repaint();
	}
	});
	}
	public void paint(Graphics g) {
	g.setColor(Color.yellow);
	g.fillRect(0, 0, getWidth(), getHeight());
```

```
        g.setColor(Color.red);
        g.fillOval(x, y, 50, 50);
        }

        public static void main(String[] args) {
        Universe universe = new Universe();
        universe.setSize(400, 400);
        universe.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        universe.setVisible(true);
        }
}
```

OUTPUT:




~/Documents/Github/Java Project
javac Universe.java
~/Documents/Github/Java Project
java Universe


~/Documents/Github/Java Project
javac Universe.java
~/Documents/Github/Java Project
java Universe

Practical 19

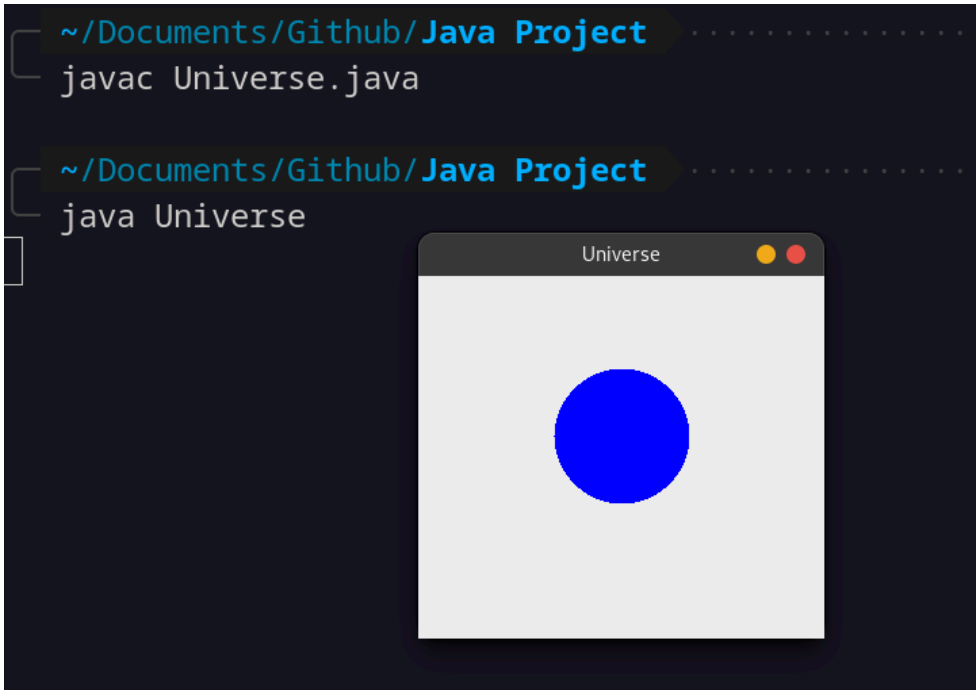
AIM: Write a program that displays the color of a circle as red when the mouse button is pressed and as blue when the mouse button is released.

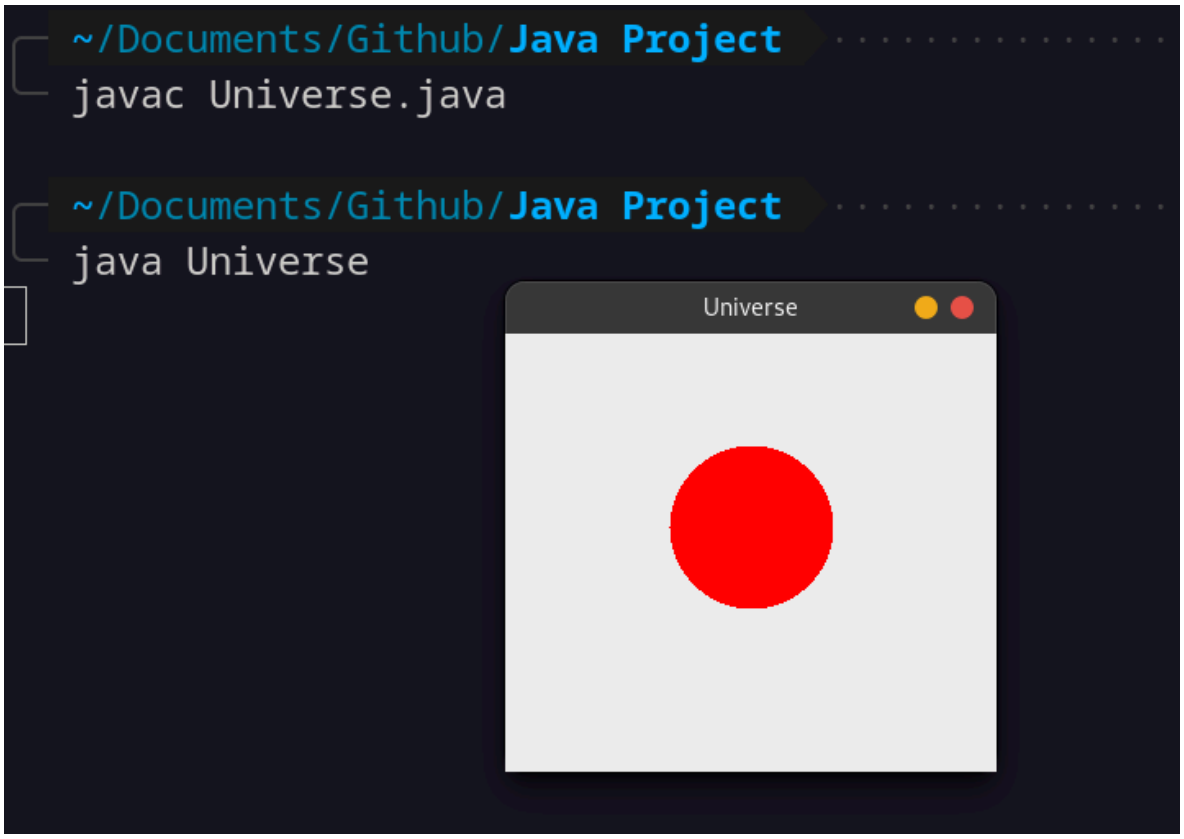
COMPLETE CODE

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Universe extends JFrame {
	private int x = 150;
	private int y = 150;
	private int radius = 50;
	private boolean isPressed = false;

	public Universe() {
	setTitle("Universe");
	setSize(300, 300);
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setLocationRelativeTo(null);
	setResizable(false);
	addMouseListener(new MouseAdapter() {
	public void mousePressed(MouseEvent e) {
	if (Math.pow(e.getX() - x, 2) + Math.pow(e.getY() - y, 2) <= Math.pow(radius, 2)) {
		isPressed = true;
		repaint();
	}
	}

	public void mouseReleased(MouseEvent e) {
	if (isPressed) {
		isPressed = false;
```

```
                repaint();
        }
        }
        });
        }

        public void paint(Graphics g) {
        super.paint(g);
        g.setColor(isPressed ? Color.RED : Color.BLUE);
        g.fillOval(x - radius, y - radius, 2 * radius, 2 * radius);
        }

        public static void main(String[] args) {
        SwingUtilities.invokeLater(new Runnable() {
        public void run() {
        new Universe().setVisible(true);
        }
        });
        }
}
```

OUTPUT:


~/Documents/Github/Java Project
javac Universe.java
~/Documents/Github/Java Project
java Universe
Universe


~/Documents/Github/Java Project
javac Universe.java
~/Documents/Github/Java Project
java Universe
Universe

Practical 20

AIM: Write a GUI program that uses buttons to move the message to the left and right and uses the radio button to change the color for the message displayed.

COMPLETE CODE

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Universe extends JFrame {
        private JButton left = new JButton("Left");
        private JButton right = new JButton("Right");
        private JLabel label = new JLabel("Welcome to Universe");
        private JRadioButton red = new JRadioButton("Red");
        private JRadioButton yellow = new JRadioButton("Yellow");
        private JRadioButton green = new JRadioButton("Green");
        private JRadioButton blue = new JRadioButton("Blue");
        private ButtonGroup group = new ButtonGroup();

        public Universe() {
        JPanel p1 = new JPanel();
        p1.add(left);
        p1.add(right);
        add(p1, BorderLayout.SOUTH);

        JPanel p2 = new JPanel();
        p2.add(red);
        p2.add(yellow);
        p2.add(green);
        p2.add(blue);
        add(p2, BorderLayout.NORTH);

        group.add(red);
```

```
group.add(yellow);
group.add(green);
group.add(blue);

label.setFont(label.getFont().deriveFont(20f)); // Increase text size to 20

add(label, BorderLayout.CENTER);

left.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
label.setLocation(label.getX() - 10, label.getY());
}
});

right.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
label.setLocation(label.getX() + 10, label.getY());
}
});

red.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
label.setForeground(Color.RED);
}
});

yellow.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
label.setForeground(Color.YELLOW);
}
});

green.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
label.setForeground(Color.GREEN);
}
});
```

```
        blue.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
        label.setForeground(Color.BLUE);
        }
        });
        }

        public static void main(String[] args) {
        Universe frame = new Universe();
        frame.setTitle("Universe");
        frame.setSize(400, 200);
        frame.setLocationRelativeTo(null);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
        }
}
```

OUTPUT:


java Universe
~/Documents/Github/Java Project
javac Universe.java
~/Documents/Github/Java Project
java Universe
Universe
Red
Yellow
Green
Blue
Welcome to Universe
Left
Right


java Universe
~/Documents/Github/Java Project
javac Universe.java
~/Documents/Github/Java Project
java Universe
Universe
Red
Yellow
Green
Blue
Welcome to Universe
Left
Right