Experiment 1: Matrix addition

```java
import java.util.Scanner;

public class Matrix {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter dimension of matrix: ");
        int n = sc.nextInt();

        int [][] First = new int[n][n];
        int [][] Second = new int[n][n];
        int [][] Sum = new int[n][n];

        System.out.println("Enter elements of first matrix: ");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                First[i][j] = sc.nextInt();
            }
        }

        System.out.println("Enter elements of second matrix: ");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                Second[i][j] = sc.nextInt();
            }
        }

        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                Sum[i][j] = First[i][j] + Second[i][j];
            }
        }

        System.out.println("Addition of two matrices: ");
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                System.out.print(Sum[i][j] + " ");
            }
            System.out.println();
        }
        sc.close();
    }
}
```

Experiment 2: interface resizable with resizeWidth() and resizeHeight()

```
interface Resizeable {
    void resizeWidth(int width);
    void resizeHeight(int height);
}

class Rectangle implements Resizeable {
    private int width;
    private int height;

    public Rectangle(int width, int height) {
        this.width = width;
        this.height = height;
    }

    public void resizeWidth(int newWidth) {
        this.width = newWidth;
    }

    public void resizeHeight(int newHeight) {
        this.height = newHeight;
    }

    public void display() {
        System.out.println("Rectangle width: " + width + ", Rectangle height: " +
height);
    }

    public static void main(String[] args) {
        Rectangle rectangle = new Rectangle(5, 10);
        rectangle.display();

        rectangle.resizeHeight(20);
        rectangle.resizeWidth(15);
        rectangle.display();
    }
}
```

Experiment 3: class shape with circle, rectangle and square

```
public class Shape {
    public void draw() {
        System.out.println("Drawing a shape");
    }

    public void erase() {
        System.out.println("Erasing a shape");
    }
}

class Circle extends Shape {
    public void draw() {
        System.out.println("Drawing a circle");
    }

    public void erase() {
        System.out.println("Erasing a circle");
    }
}

class Triangle extends Shape {
    public void draw() {
        System.out.println("Drawing a triangle");
    }

    public void erase() {
        System.out.println("Erasing a triangle");
    }
}

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

    public void erase() {
        System.out.println("Erasing a square");
    }
}

class Main {
    public static void main(String[] args) {
        Circle c = new Circle();
```

```
        Triangle t = new Triangle();
        Square s = new Square();

        System.out.println("Using circle object:");
        c.draw();
        c.erase();

        System.out.println("Using triangle object:");
        t.draw();
        t.erase();

        System.out.println("Using square object:");
        s.draw();
        s.erase();
    }
}
```

Experiment 4: outer class and inner class

```
public class Outer {
    public void display() {
        System.out.println("Outer class display method");


    class Inner {
        public void display() {
            System.out.println("Inner class display method");
        }
    }
    Inner a = new Inner();
    a.display();
    }
    public static void main(String[] args) {
        Outer out = new Outer();
        out.display();
    }
}
```

Experiment 5: custom exceptions for zero by division error

```java
import java.util.Scanner;

public class ZeroD extends Exception {
    public ZeroD(String msg) {
        super(msg);
    }
    class Custom {
        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);

            System.out.println("Enter 2 numbers: ");
            int a = sc.nextInt();
            int b = sc.nextInt();

            try {
                if (b == 0) {
                    throw new ZeroD("Zero division not allowed");
                }
                float res = (float) a / b;
                System.out.println("Result: " + res);
            } catch (ZeroD e) {
                System.out.println(e.getMessage());
            } finally {
                sc.close();
            }
        }
    }
}
```

Experiment 6: create a package named mypack

```
//File 1:
package mypack;

public class mypack {

        public void display()
        {
            System.out.println("HELLO FROM MYPACK");
        }
}

//File 2:
package mypack;

public class Testpack {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
            mypack obj = new mypack();
            obj.display();
    }

}
```

Experiment 7: creation of thread

```java
public class Mythread implements Runnable {

    public void run() {
        for (int i = 1; i <= 5; i++) {
            System.out.println(Thread.currentThread().getName() + " i is " + i);
            try {
                Thread.sleep(500);
            } catch (InterruptedException e) {
                System.out.println(e.getMessage());
            }
        }
    }

    public static void main(String[] args) {
        Mythread mythread = new Mythread();

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

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

Experiment 8: main and child thread

```
public class Test{
    public static void main(String[] args)
    {
        new Mythread();
        try
        {
            for (int i = 5; i > 0; i--)
            {
                System.out.println("Running main thread: " + i);
                Thread.sleep(1000);
            }
        } catch (InterruptedException e)
        {
        }
        System.out.println("Exiting main thread");
    }
}
class Mythread extends Thread
{

    Mythread()
    {
        super("Using thread class");
        System.out.println("Child thread: "+ this);
        start();
    }
    public void run ()
    {
        try
        {
            for (int i = 5; i > 0; i--)
            {
                System.out.println("Child thread: " + i);
                Thread.sleep(500);
            }
        } catch (InterruptedException e)
        {
        }
        System.out.println("Exiting child thread");
    }
}
```