Module 1:

1. EvenOdd

```
public class EvenOdd {

        public static void main(String[] args) {
                int n= Integer.parseInt(args[0]);
                if(n%2==0) {
                        System.out.println("Number "+n+" is even");
                }else {
                        System.out.println("Number "+n+" is odd");
                }
        }

}
```

2. Palindrome

```
import java.util.Scanner;

public class NumberPalindrome {

        public static void main(String[] args) {

                int a=Integer.parseInt(args[0]);
                if(isNumberPalindrome(a)) {
                        System.out.println("Palindrome");
                }else {
                        System.out.println("Not a palindrome");
                }
        }
        public static boolean isNumberPalindrome(int a) {
                int b=a,rev=0;
                while(a!=0) {
                        rev=rev*10+a%10;
                        a=a/10;
                }
                if(rev==b) {
                        return true;
                }
                return false;
        }
}
```

3. Bitwise XOR

```
public class BitwiseXOR {

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

```
                        int a=Integer.parseInt(args[0]);
                        int b=Integer.parseInt(args[1]);
                        System.out.println("XOR Result = "+(a^b));

        }

}
```

4. Min-Max

```
public class minAndMaxArray {

        public static void main(String[] args) {

                int n = Integer.parseInt(args[0]);
        int[] arr = new int[n];

        System.out.println("Array size: " + n);

        for (int i = 0; i < n; i++) {
            arr[i] = Integer.parseInt(args[i + 1]);
        }

        System.out.print("Array elements: ");
        for (int i = 0; i < n; i++) {
            System.out.print(arr[i] +" ");
        }


                int min=arr[0],max=arr[0];
                for(int i=1;i<n;i++) {
                        if(arr[i]>max)
                                max=arr[i];

                        if(arr[i]<min)
                                min=arr[i];

                }
                System.out.println("\nMaximum: "+max+"\nMinimum: "+min);
        }
}
```

5. SortNames

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

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

        Arrays.sort(args);
        System.out.println("Sorted names:");
        for (String name : args) {
            System.out.println(name);
        }
    }
}
```

Module 2:

1. Method Overrideing

```
class A{
        public void print() {
                System.out.println("A method printed");
        }
}
class B extends A{
        public void print() {
                System.out.println("B method printed overriding A");
        }
}
public class MethodOverriding {

        public static void main(String[] args) {

                A obj1=new A();
                obj1.print();
                A obj2=new B();
                obj2.print();
        }

}
```

2. ConstructorDemo

```
class PC{
        int l;
        int b;
        PC(int x,int y){
                l=x;
                b=y;
        }
        public void area() {
                System.out.println("The Area of Rectangle is "+(l*b));
        }
}
```

```
}


public class Constructor {

        public static void main(String[] args) {
                int l=Integer.parseInt(args[0]);
                int b=Integer.parseInt(args[1]);
                PC rect = new PC(l,b);
                rect.area();
        }

}


3. Resistance


class SeriesResistance{
    double r1,r2;
        SeriesResistance(double r1, double r2) {
                this.r1=r1;
        this.r2=r2;
        }

        public void display() {
                System.out.println("Series Resistance: "+(r1+r2));
        }
}
class ParallelResistance{
    double r1,r2;
        ParallelResistance(double r1, double r2) {
                this.r1=r1;
        this.r2=r2;
        }

        public void display() {
                System.out.println("Parallel Resistance: "+(r1*r2)/(r1+r2));
        }
}
public class Resistance {

        public static void main(String[] args) {
                double m=Double.parseDouble(args[0]);
        double n=Double.parseDouble(args[1]);
                SeriesResistance r1=new SeriesResistance(m,n);
                r1.display();
        ParallelResistance r2=new ParallelResistance(m,n);
                r2.display();
        }

}
```

4. AreaOfRect&Triangle

```
class Rectangle{
        double a,b;
        Rectangle(double a,double b){
                this.a=a;
                this.b=b;
        }
        void area(){
                System.out.println("Area of rectangle: "+(a*b));
        }
}
class Triangle{
        double a,b;
        Triangle(double a,double b){
                this.a=a;
                this.b=b;
        }
        void area(){
                System.out.println("Area of triangle: "+(a*b)/2);
        }
}
public class AreaOfRectAndTria{
        public static void main(String[] args){
                double a=Double.parseDouble(args[0]);
                double b=Double.parseDouble(args[1]);
                double c=Double.parseDouble(args[2]);
                double d=Double.parseDouble(args[3]);
                Rectangle r=new Rectangle(a,b);
                r.area();
                Triangle t=new Triangle(c,d);
                t.area();

        }
}
```

5. PreventInheritance

```
import java.util.Scanner;
final class Fig{
        int a;
}
//class Square extends fig{
class Square{
    int a;
        Square(int a){
                this.a=a;
        }
        void Area(){
                System.out.println("Inheritance overcomed");
                System.out.println("Area of Square is "+a*a+".0");
        }
}
public class PrevInheritance{
        public static void main(String[] args){
                Scanner s = new Scanner(System.in);
                System.out.print("Enter the length of Square: ");
                int m=s.nextInt();
                Square b=new Square(m);
            b.Area();
        }
}
```

Module 3:

1. String2Num(NumberFormatException)

```
import java.lang.Exception;

public class String2Num
{
    public static void main(String[] args)
    {
        String input = args[0];
        try
        {
            int number = Integer.parseInt(input);
            System.out.println("" + (2 * number));
        }
        catch(Exception e)
        {
            System.out.println("Error: The input is not a valid integer");

        }
    }
}
```

2. InvalidAgeException

```
import java.lang.Exception;
class InvalidAgeException extends Exception {
    public InvalidAgeException(String message) {
        super(message);
    }
}

class Age {
    public void checkAge(int age) throws InvalidAgeException {
        if (age > 0 && age < 150) {
            System.out.println("Age is: " + age);
        } else {
            throw new InvalidAgeException("Age "+age+" is invalid");
        }
    }
}

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

        int age = Integer.parseInt(args[0]);
        Age a = new Age();
        try {
            a.checkAge(age);
        } catch (InvalidAgeException e) {
            System.out.println("Caught: " + e.getMessage());
        }
    }
}
```

3. ShapeDemo(AreaOfRect&Circle)

```
import java.util.Scanner;
import java.lang.Math;
interface Shape
{
    abstract void getData();
    abstract void Display();
}

class Rectangle implements Shape
{
    int a, b;
    public void getData()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter length");
        a = sc.nextInt();
        System.out.println("Enter width");
        b = sc.nextInt();
    }

    public void Display()
    {
        int area1 = a * b;
        System.out.println("Area of Rectangle is " + area1);
    }
}
```

```
class Circle implements Shape
{
    double r;
    public void getData()
    {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the radius of the circle: ");
        r = sc.nextDouble();
    }

    public void Display()
    {
        double area2 = 2*Math.PI*r;
        System.out.printf("Area of Circle is %.2f%n" ,area2);  // Used to print
Area till 2 decimal places
    }
}

public class ShapeDemo
{
    public static void main(String[] args)
    {
        // Test ectangle
        Rectangle rec = new Rectangle();
            rec.getData();
            rec.Display();

        // Test Circle
        Circle cir = new Circle();
            cir.getData();
            cir.Display();
    }
}
```

4. DrawSquare

```
interface Drawable
{
    void draw();
}

class Square implements Drawable
{
    public void draw()
    {
        System.out.println("Drawing a square");

    }
}
```

```
public class Draw
{
    public static void main(String[] args)
    {
        Square sq = new Square();
        sq.draw();
    }
}
```

Module 4:

1. Stack Operations(Multithread)

```
import java.util.Stack;

class StackBuffer {
    private Stack<Integer> stack = new Stack<>();

    synchronized void push(int item) {
        stack.push(item);
        System.out.println("Pushed: " + item);
        notify();
    }

    synchronized int pop() {
        while (stack.isEmpty()) {
            try {
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        }
        int item = stack.pop();
        System.out.println("Popped: " + item);
        return item;
    }
}

class StackProducer implements Runnable {
    private StackBuffer stackBuffer;

    StackProducer(StackBuffer stackBuffer) {
        this.stackBuffer = stackBuffer;
    }

    public void run() {
        int i = 0, n = 5;
        while (i < n) {
            stackBuffer.push(i++);
        }
        System.out.println();
    }
}

class StackConsumer implements Runnable {
    private StackBuffer stackBuffer;

    StackConsumer(StackBuffer stackBuffer) {
        this.stackBuffer = stackBuffer;
    }

    public void run() {
        while (true) {
            stackBuffer.pop();
        }
    }
}

public class Stacks{
    public static void main(String args[]) {
        StackBuffer stackBuffer = new StackBuffer();
        StackProducer producer = new StackProducer(stackBuffer);
        StackConsumer consumer = new StackConsumer(stackBuffer);

        Thread producerThread = new Thread(producer);
        Thread consumerThread = new Thread(consumer);

        producerThread.start();
        consumerThread.start();
    }
}
```

2. MultiThreadDemo

```
class Multithread implements Runnable
{
     public synchronized void run()
    {
        Thread t= Thread.currentThread();
        System.out.println("Thread executing: "+t.getName());
        for(int i=0;i<5;i++)
        {
            System.out.println(i);
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
public class multipleThread
{
    public static void main(String[] args) {
        Multithread mt=new Multithread();
        Thread t1=new Thread(mt,"Thread1");
        t1.start();
        Thread t2=new Thread(mt,"Thread2");
        t2.start();

    }
}
```

3. Multiplication Table(multithread)

```
class MTG extends Thread {
    private int startNumber;

    public MTG(int startNumber) {
        this.startNumber = startNumber;
    }

    public void run() {
        System.out.println("Multiplication table for " + startNumber + ":");
        for (int j = 1; j <= 10; j++) {
            System.out.println(startNumber + " * " + j + " = " + (startNumber *
j));
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        System.out.println();
    }
}

public class MultiplicationTables {
    public static void main(String[] args) {
        // Create two threads for generating multiplication tables
        MTG thread1 = new MTG(5);
        MTG thread2 = new MTG(6);

        // Start thread1
        thread1.start();
        try {
            // Wait for thread1 to finish
            thread1.join();
        } catch (InterruptedException e) {
```

```java
            e.printStackTrace();
        }

        // Start thread2 after thread1 finishes
        thread2.start();
        try {
            // Wait for thread2 to finish
            thread2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

4. EvenOddThread(multithread)

```java
class Multithread implements Runnable {
    boolean isEven;

    public Multithread(boolean isEven) {
        this.isEven = isEven;
    }

    public synchronized void run() {

            for (int i = isEven ? 0 : 1; i <= 11; i += 2) {
                System.out.println(i);
                try {
                    Thread.sleep(500);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
            System.out.println();
            notify();

    }
}

public class EvenOddThread {
    public static void main(String[] args) {
        Multithread mt = new Multithread(true);
        Thread t1 = new Thread(mt);
        Thread t2 = new Thread(new Multithread(false));
        t1.start();
        synchronized (mt) {
            try {
                mt.wait(); // main thread waits for t1 to finish
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
        t2.start();
    }
}
```

5. ProducerConsumer

```
class Buffer {
    int item;
    boolean flag = false;
    synchronized void put(int item) {
        while (flag) {
            try {
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        }
        this.item = item;
        flag = true;
        System.out.println("Put: " + item);
        notify();
    }
    synchronized int get() {
        while (!flag) {
            try {
                wait();
            } catch (InterruptedException e) {
                System.out.println("InterruptedException caught");
            }
        }
        System.out.println("Got: " + item);
        flag = false;
        notify();
        return item;
    }
 }
 class Producer implements Runnable {
    Buffer b;
    Producer(Buffer b) {
        this.b = b;
    }
    public void run() {
        int i = 0, n = 25;
        while (i < n) {
            b.put(i++);
        }
    }
 }
```

```
class Consumer implements Runnable {
    Buffer b;
    Consumer(Buffer b) {
        this.b = b;
    }
    public void run() {
        while (true) {
            b.get();
        }
    }
 }
 public class ProducerConsumer{
    public static void main(String args[]) {
        Buffer b = new Buffer();
        Producer p = new Producer(b);
        Consumer c = new Consumer(b);
        Thread pr = new Thread(p);
        Thread con = new Thread(c);
        pr.start();
        con.start();
    }
 }
```