//program 1 command line arguments

```
import java.util.*;


class Program1 {
   public static void main(String args[]) {
      int n;
      int sum = 0;
      Scanner sc = new Scanner(System.in);
      System.out.println("Enter integers separated by spaces:");
      String s = sc.nextLine();
      StringTokenizer st = new StringTokenizer(s, " ");
      while (st.hasMoreTokens()) {
         String temp = st.nextToken();
         n = Integer.parseInt(temp);
         System.out.println(n);
         sum = sum + n;
      }
      System.out.println("Sum of the integers is: " + sum);
      sc.close();
   }
}
```

//prog 2 overloading

```
class ClassDemo {
   int sum(int a, int b, int c) {
      return (a + b + c);
   }

   int sum(int a, int b) {
      return (a + b);
   }

   float sum(int a, float b) {
      return (a + b);
   }

   int sum(int a) {
      System.out.println("too few arguments");
      return 0;
   }
}

public class Overloading {
   public static void main(String[] args) {
      int a;
      ClassDemo obj = new ClassDemo();

      System.out.println("Calling sum(int a, int b, int c)");
      a = obj.sum(20, 3, 5);
      System.out.println(a);
```

```
        System.out.println("Calling sum(int a, int b)");
        a = obj.sum(20, 3);
        System.out.println(a);

        System.out.println("Calling sum(int a)");
        a = obj.sum(5);
        System.out.println(a);
    }
}
```

// Program 3 demonstrating single and multilevel inheritance

```
class Employee {
    int basic;

    Employee() {
        basic = 40000;
    }

    public static void main(String[] args) {
        Employee employee = new Employee();
        System.out.println("Basic salary of the employee: " + employee.basic);
    }
}

class Programmer extends Employee {
    int pcomp;

    Programmer() {
        pcomp = 25000;
```

```
    }

    int calculateSalary() {
        return (basic + pcomp);
    }
}

class TechLead extends Programmer {
    int tcomp;

    TechLead() {
        tcomp = 30000;
    }

    int calculateSalary() {
        return (basic + pcomp + tcomp);
    }
}

class InheritanceDemo {
    public static void main(String ar[]) {
        Programmer p = new Programmer();
        // Call a method
        System.out.println("Programmer's salary is " + p.calculateSalary());

        TechLead t = new TechLead();
        System.out.println("TechLead's salary is " + t.calculateSalary());
    }
}
```

//program 4 Polymorphism example

```
class Car {
   public void brand() {
      System.out.println("This is a base class method");
   }
}
class Truck extends Car {
   public void brand() {
      System.out.println("Benz");
   }
}

class SportCar extends Car {
   public void brand() {
      System.out.println("Supra / Jeep");
   }
}

class PolymorphismDemo {
   public static void main(String ar[]) {
      Car c;
      c = new SportCar();
      c.brand();
      c = new Truck();
      c.brand();
      c = new Car();
      c.brand();
   }
}
```

```
// program 5a Abstract methods example
abstract class Car {
   Car() {
      System.out.println("Car's constructor called");
   }

   abstract void run();

   void changeGear() {
      System.out.println("Gear changed");
   }
}

class BMW extends Car {
   void run() {
      System.out.println("BMW's run() called");
   }
}

class MainClass1p5 {
   public static void main(String[] args) {
      Car c = new BMW();
      c.run();
      c.changeGear();
   }
}
```

//program 6a Exception Handling

```
class ExceptionDemo {
   public static void main(String ar[]) {
      try {
         System.out.println("Inside try block");
         int a = 10;
         System.out.println("a=" + a);
         int b;
         b = a / 0;
         System.out.println("b=" + b);
      } catch (Exception e) {
         System.out.println("Inside catch block");
         System.out.println(e.getMessage());
      } finally {
         System.out.println("Inside finally block");
         System.out.println("I am always executed");
      }
   }
}
```

//program 5b Program to create an interface

```
interface Polygon {
   // Abstract method
   void calculateArea(int a, int b);
}

class Rectangle implements Polygon {
   public void calculateArea(int a, int b) {
      int area = a * b;
```

```
        System.out.println("Area of rectangle is " + area);
    }
}

class Triangle implements Polygon {
    public void calculateArea(int a, int b) {
        double area = 0.5 * a * b;
        System.out.println("Area of triangle is " + area);
    }
}

class InterfaceDemo {
    public static void main(String ar[]) {
        Rectangle r = new Rectangle();
        r.calculateArea(10, 20);
        Triangle t = new Triangle();
        t.calculateArea(2, 4);
    }
}
```

```
//program 6b throwdemo
class ThrowDemo {
```

```
    static void checkAge(int age) {
        if (age < 18) {
            throw new ArithmeticException("Not eligible to vote");
        } else {
            System.out.println("Eligible");
        }
    }

    public static void main(String ar[]) {
        checkAge(21);
    }
}
```

Program 6c

```
import java.io.*;

class Example {
   void fun(int a) throws IOException {
      if (a == 1) {
         throw new IOException("IOException thrown");
      }
      System.out.println("I am in method fun()");
   }
}

class ThrowsDemo {
   public static void main(String[] args) {
      try {
         Example ex = new Example();
         ex.fun(9);
      } catch (IOException e) {
         System.out.println(e);
      }
   }
}
```

Program 7a Runnable

```
// Create a new thread
// Implementing the Runnable interface
// Extending the Thread class
```

```
class NewThread implements Runnable {
   Thread t;

   NewThread() {
      t = new Thread(this, "BCA Thread");
      System.out.println("I am a new thread");
      t.start();
   }

   public void run() {
      try {
         for (int i = 5; i >= 1; i--) {
            System.out.println(i);
            Thread.sleep(1000);
         }
      } catch (InterruptedException ex) {
         System.out.println("Exception caused in thread");
      }
   }
}

class ThreadDemo {
   public static void main(String ar[]) {
      NewThread ob = new NewThread();
      try {
         for (int i = 10; i >= 5; i--) {
            System.out.println("Main thread " + i);
            Thread.sleep(2000);
         }
```

```
        } catch (InterruptedException ex) {
            System.out.println("Exception caused in thread");
        }
        System.out.println("Back to main thread. New thread completes");
    }
}
```

7b extends

```
public class ExtendedThread {
    public static void main(String[] args) {
        new NewThread();
        try {
            for (int i = 5; i > 0; i--) {
                System.out.println("Main Thread: " + i);
                Thread.sleep(1000);
            }
        } catch (InterruptedException e) {
            System.out.println("Main thread interrupted.");
        }
        System.out.println("Main thread exiting.");
    }
}

class NewThread extends Thread {
    NewThread() {
        super("Demo Thread");
        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("Child thread interrupted.");
        }
        System.out.println("Exiting child thread.");
    }
}
```

Program 8 Serialization

```
import java.io.*;

class Student implements Serializable {
    int id;
    String name;

    public Student(int id, String name) {
        this.id = id;
        this.name = name;
    }
}
```

```
class Persist {
   public static void main(String args[]) {
      try {
         // Creating a Student object
         Student s1 = new Student(211, "Sravani");

         // Writing the Student object to a file
         FileOutputStream fout = new FileOutputStream("f.txt");
         ObjectOutputStream out = new ObjectOutputStream(fout);
         out.writeObject(s1);
         out.flush();
         out.close();
         System.out.println("Serialization successful");

         // Reading the serialized object from the file
         ObjectInputStream in = new ObjectInputStream(new FileInputStream("f.txt"));
         Student s = (Student)in.readObject();

         // Printing the data of the deserialized object
         System.out.println("Deserialized Student:");
         System.out.println("ID: " + s.id);
         System.out.println("Name: " + s.name);
         in.close();
      } catch (Exception e) {
         System.out.println(e);
      }
   }
}
```

Program 9 multi-threading

```
class MultithreadDemo {
   public static void main(String args[]) {
      GoodMorning t1 = new GoodMorning();
      Hello t2 = new Hello();
      Welcome t3 = new Welcome();
      t1.start();
      t2.start();
      t3.start();
   }
}

class GoodMorning extends Thread {
   synchronized public void run() {
      try {
         int i = 0;
         while (i < 5) {
            sleep(1000);
            System.out.println("Good morning");
            i++;
         }
```

```
        } catch (InterruptedException e) {
            // Handle InterruptedException if needed
        }
    }
}

class Hello extends Thread {
    synchronized public void run() {
        try {
            int i = 0;
            while (i < 5) {
                sleep(2000);
                System.out.println("Hello");
                i++;
            }
        } catch (InterruptedException e) {
            // Handle InterruptedException if needed
        }
    }
}

class Welcome extends Thread {
    synchronized public void run() {
        try {
            int i = 0;
            while (i < 5) {
                sleep(3000);
                System.out.println("Welcome");
                i++;
```

```
        }
      } catch (InterruptedException e) {
        // Handle InterruptedException if needed
      }
    }
}
```

Program 10 TestSynchronization2

```
class TestSynchronization2 {
   public static void main(String args[]) {
      Table obj = new Table(); // only one object
      MyThread1 t1 = new MyThread1(obj);
      MyThread2 t2 = new MyThread2(obj);
      t1.start();
      t2.start();
   }
}

class Table {
   synchronized void printTable(int n) {
      // synchronized method
      for (int i = 1; i <= 5; i++) {
         System.out.println(n * i);
         try {
            Thread.sleep(400);
         } catch (InterruptedException e) {
            System.out.println(e);
         }
```

```
        }
    }
}

class MyThread1 extends Thread {
    Table t;

    MyThread1(Table t) {
        this.t = t;
    }

    public void run() {
        t.printTable(5);
    }
}

class MyThread2 extends Thread {
    Table t;

    MyThread2(Table t) {
        this.t = t;
    }

    public void run() {
        t.printTable(100);
    }
}
```