```
/*
▨ Java Exception Handling Revision File
All ExceptionDemo classes in one place with explanations.
Use this file only for STUDY/REVISION (cannot compile as multiple classes have
same names).
*/

//
================================================================================
// ◈ ExceptionDemo3 (Version 1 – Multiple catch blocks)
// Exceptions used:
// - ArrayIndexOutOfBoundsException
// - ArithmeticException
// - NullPointerException (typo in code: NullpointerException)
// - RuntimeException
// - Exception
// - Throwable
// ✓ Conclusion: Shows multiple catch blocks from specific → broad.
class ExceptionDemo3_V1 {
    public static void main(String[] args) {
        System.out.println("1: start");
        String arr[] = {"12", "2"};
        try {
            String s1 = arr[0];
            String s2 = arr[11]; // ✕ ArrayIndexOutOfBoundsException
            int i = Integer.parseInt(s1);
            int j = Integer.parseInt(s2);
            int k = i / j; // ✕ ArithmeticException
            System.out.println(k);
        } catch (ArrayIndexOutOfBoundsException | ArithmeticException |
NullpointerException e) {
            e.printStackTrace();
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } catch (Throwable e) {
            e.printStackTrace();
        } finally {
            System.out.println("Yes, everything is fine!!!");
        }
        System.out.println("100: Completed");
    }
}

//
================================================================================
// ◈ ExceptionDemo3 (Version 2 – Multi-catch + finally)
// Exceptions used: ArrayIndexOutOfBoundsException, ArithmeticException,
NullPointerException
// ✓ Conclusion: Demonstrates multi-catch and finally block.
```

```
class ExceptionDemo3_V2 {
    public static void main(String[] args) {
        try {
            String arr[] = {"12", "2"};
            String s2 = arr[11]; // ❌ ArrayIndexOutOfBoundsException
        } catch (ArrayIndexOutOfBoundsException | ArithmeticException |
NullpointerException e) {
            e.printStackTrace();
        } finally {
            System.out.println("Yes, everything is fine!!!");
        }
    }
}

//
================================================================================
// 🔷 ExceptionDemo3 (Version 3 - Broad hierarchy)
// Exceptions used: RuntimeException, Exception, Throwable
// ✅ Conclusion: Catches exceptions in a broad hierarchy order.
class ExceptionDemo3_V3 {
    public static void main(String[] args) {
        try {
            String arr[] = {"12", "2"};
            String s2 = arr[11]; // ❌ ArrayIndexOutOfBoundsException
        } catch (RuntimeException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        } catch (Throwable e) {
            e.printStackTrace();
        } finally {
            System.out.println("Yes, everything is fine!!!");
        }
    }
}

//
================================================================================
// 🔷 ExceptionDemo3 (Version 4 - Single catch)
// Exceptions used: Exception
// ✅ Conclusion: Shows single catch block catching all exceptions.
class ExceptionDemo3_V4 {
    public static void main(String[] args) {
        try {
            String arr[] = {"12", "2"};
            String s2 = arr[11]; // ❌ ArrayIndexOutOfBoundsException
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            System.out.println("Yes, everything is fine!!!");
        }
    }
}
```

```
//
================================================================================
// 🔷 ExceptionDemo7
// Exceptions used: Throwable
// ✅ Conclusion: Base class catch – can handle all errors & exceptions.
class ExceptionDemo7 {
    public static void main(String[] args) {
        try {
            String arr[] = {"12", "2"};
            String s2 = arr[11]; // ❌ ArrayIndexOutOfBoundsException
        } catch (Throwable e) {
            e.printStackTrace();
        } finally {
            System.out.println("Yes, everything is fine!!!");
        }
    }
}

//
================================================================================
// 🔷 ExceptionDemo8
// Exceptions used: Exception
// ✅ Conclusion: Uses Exception catch for any runtime error.
class ExceptionDemo8 {
    public static void main(String[] args) {
        try {
            String arr[] = {"12", "2"};
            String s2 = arr[11]; // ❌ ArrayIndexOutOfBoundsException
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            System.out.println("Yes, everything is fine!!!");
        }
    }
}

//
================================================================================
// 🔷 ExceptionDemo9
// Exceptions used: Throwable
// Runtime exception triggered: NullPointerException
// ✅ Conclusion: Demonstrates NPE with null reference and finally block.
class ExceptionDemo9 {
    void m1() {
        System.out.println("m1() : executed");
    }
    public static void main(String[] args) {
        ExceptionDemo9 d1 = null; // ❌ NullPointerException
        try {
            d1.m1();
        } catch (Throwable e) {
            e.printStackTrace();
        } finally {
            System.out.println("Bhai Resources ko release kar do!!!");
        }
    }
}

//
================================================================================
// 🔷 ExceptionDemo10
// Exceptions used: NullPointerException, ArithmeticException
// Also demonstrates: throw keyword
// ✅ Conclusion: Shows manual throwing of exceptions.
class ExceptionDemo10 {
    public static void main(String[] args) {
        try {
            throw new ArithmeticException(); // manual throw
            // throw new NullPointerException();
        } catch (NullPointerException e) {
            e.printStackTrace();
        } catch (ArithmeticException e) {
            e.printStackTrace();
        } finally {
            System.out.println("Bhai Resources ko release kar do!!!");
        }
    }
}

//
================================================================================
// 🔷 ExceptionDemo12
// Exceptions used: Exception, ArithmeticException
// Also demonstrates: re-throwing exceptions
// ✅ Conclusion: Shows nested try-catch and rethrowing.
class ExceptionDemo12 {
    public static void main(String args[]) {
        try {
            int i = 1 / 0; // ❌ ArithmeticException
        } catch (Exception e) {
            try {
                throw e; // re-throwing
            } catch (ArithmeticException e1) {
                e.printStackTrace();
            }
        } finally {
            System.out.println("Release resources");
        }
    }
}

//
================================================================================
// 🔷 ExceptionDemo13
// Exceptions used: ArithmeticException
// ✅ Conclusion: Exception thrown, no handling in chain.
class ExceptionDemo13 {
    static void m3() { throw new ArithmeticException(); }
    static void m2() { m3(); }
    static void m1() { m2(); }
    static void m() { m1(); }
    public static void main(String args[]) { m(); }
}

//
================================================================================
// 🔷 ExceptionDemo14
// Exceptions used: ArithmeticException (handled in m3())
// ✅ Conclusion: Local handling of exception.
class ExceptionDemo14 {
    static void m3() {
        try { int i = 1 / 0; } catch (ArithmeticException e) {
e.printStackTrace(); }
    }
    static void m2() { m3(); }
    static void m1() { m2(); }
    static void m() { m1(); }
    public static void main(String args[]) { m(); }
}

//
================================================================================
// 🔷 ExceptionDemo15
// Exceptions used: ArithmeticException (handled in m1())
// ✅ Conclusion: Exception handled higher up the call stack.
class ExceptionDemo15 {
    static void m3() {}
    static void m2() { m3(); }
    static void m1() {
        m2();
        try { int i = 1 / 0; } catch (ArithmeticException e) {
e.printStackTrace(); }
    }
    static void m() { m1(); }
    public static void main(String args[]) { m(); }
}

//
================================================================================
// 🔷 ExceptionDemo16
// Exceptions used: NullPointerException (explicit),
ArrayIndexOutOfBoundsException, StringIndexOutOfBoundsException
// ✅ Conclusion: Demonstrates multiple runtime exceptions.
class ExceptionDemo16 {
    public static void main(String args[]) {
        try { throw new NullPointerException(); }
        catch (Exception e) { e.printStackTrace(); }

        int arr[] = {1, 2, 3, 4, 5};
        arr[10] = 10; // ❌ ArrayIndexOutOfBoundsException

        String str = "abcd";
        char c = str.charAt(6); // ❌ StringIndexOutOfBoundsException
    }
}

//
================================================================================
// 🔹 ExceptionDemo17
// Exceptions used: NullPointerException, Exception
// ✅ Conclusion: Demonstrates System.exit(0) → finally won’t execute.
class ExceptionDemo17 {
    public static void main(String args[]) {
        try {
            System.exit(0); // JVM exits, finally won’t run
            throw new NullPointerException();
        } catch (Exception e) {
            e.printStackTrace();
        } finally {
            System.out.println("Finally .."); // will not run
        }
    }
}

//
================================================================================
// 🔹 LastExceptionDemo (Custom Exception)
// Exceptions used: SalaryException (extends Exception)
// ✅ Conclusion: Shows custom checked exception and throws/throws handling.
class SalaryException extends Exception {
    SalaryException() { super("Aisa bhi kao salary hota hai kay ?"); }
}
class LastExceptionDemo {
    static void salary(float sal) throws SalaryException {
        if (sal > 10000) System.out.println("Salary = " + sal);
        else throw new SalaryException();
    }
    public static void main(String args[]) throws java.util.InputMismatchException
{
        java.util.Scanner sc = new java.util.Scanner(System.in);
        System.out.println("Enter Salary :");
        float s = sc.nextFloat();
        try { salary(s); }
        catch (SalaryException e) {
            e.printStackTrace();
            System.out.println("Ho gaya Exception!!!! " + e.getMessage());
        }
    }
}

//
================================================================================
// 🔹 Test (Salary validation example)
// Exceptions used: SalaryException (Custom Exception extending Exception)
// ✅ Conclusion: Another custom exception demo.
class SalaryException2 extends Exception {
```

```java
    SalaryException2() { super("What !!!! This is not any salary??????"); }
}
public class Test {
    static void salary(float sal) throws SalaryException2 {
        if (sal > 10000) System.out.println("Salary = " + sal);
        else throw new SalaryException2();
    }
    public static void main(String[] args) {
        java.util.Scanner sc = new java.util.Scanner(System.in);
        System.out.println("Enter salary:");
        float s = sc.nextFloat();
        try { salary(s); }
        catch (SalaryException2 e) { System.out.println(e); }
    }
}
```