```java
public class NestedIterator implements Iterator<Integer> {
    List<Integer> arr;
    Iterator<Integer> itr;

    public NestedIterator(List<NestedInteger> nestedList) {
        arr = new ArrayList<>();
        flatten(nestedList);
        itr = arr.iterator();
    }

    public void flatten(List<NestedInteger> nestedList){
        for(NestedInteger data: nestedList){
            if(data.isInteger() == true){
                arr.add(data.getInteger());
            } else {
                flatten(data.getList());
            }
        }
    }

    @Override
    public Integer next() {
        return itr.next();
    }

    @Override
    public boolean hasNext() {
        return itr.hasNext();
    }
}
```

extra space flattened list → O(n) space

preprocessing

O(n) time

O(1) time on list

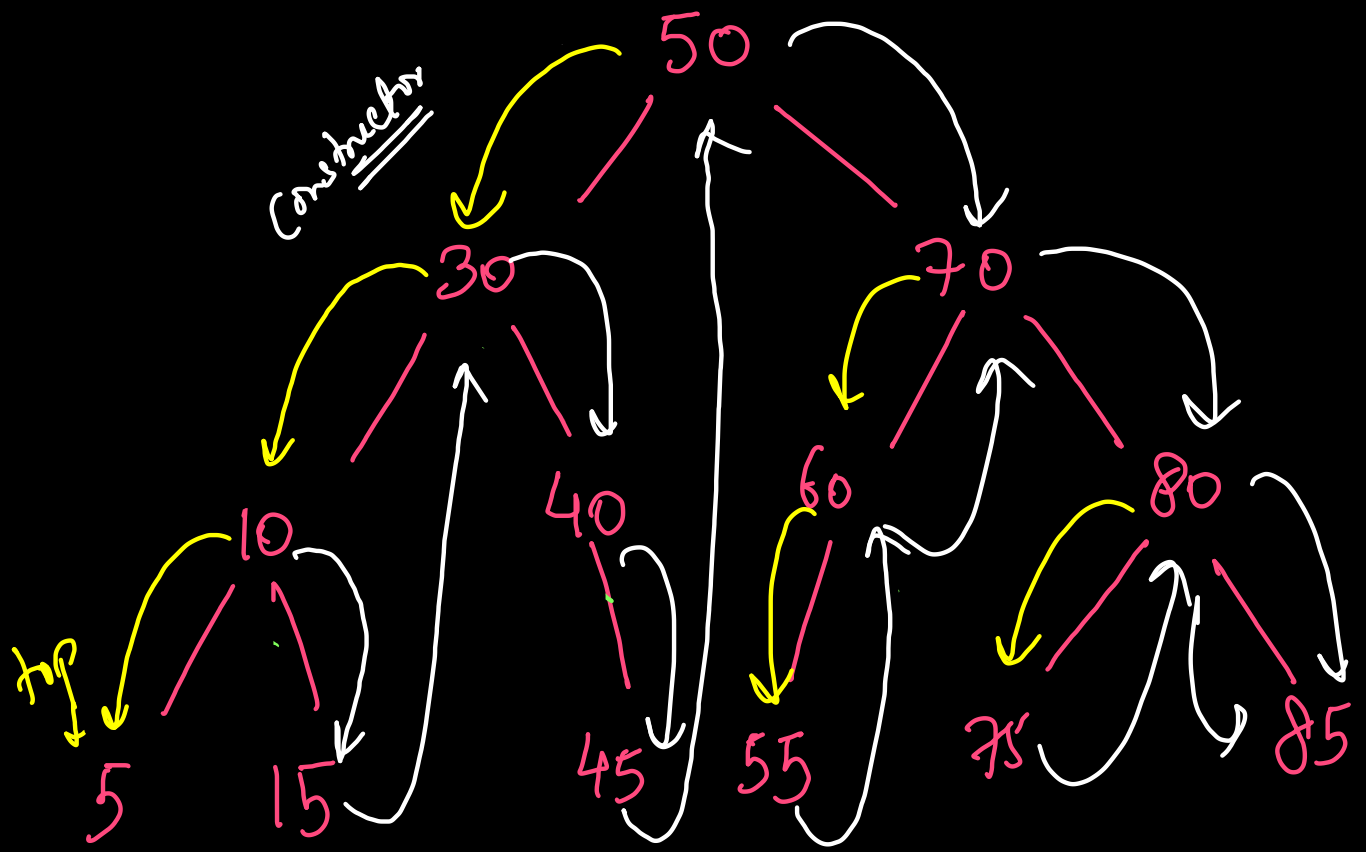
driver code

NestedIterator obj = new NestedIterator(trav);

① Constructor call

```
while(obj.hasNext())
    Syso(obj.next());
```

Binary Search Tree Iterator



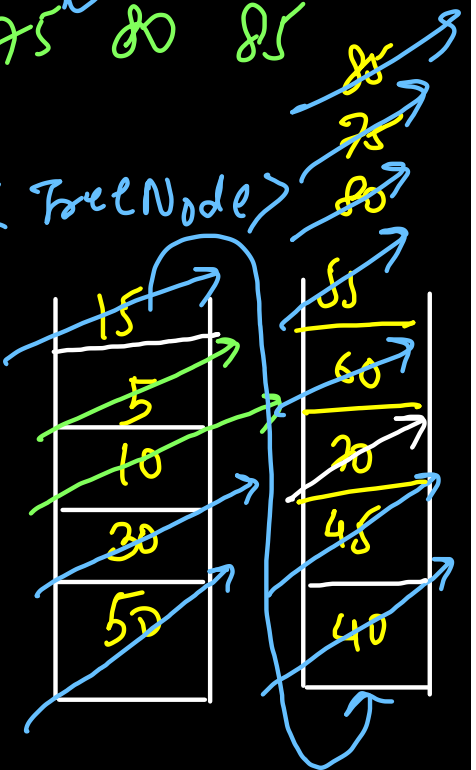
root

5 10 15 30 40

45 50 55 60

70 75 80 85

Stack<TreeNode>



stk.size() == 0 : false

stk.size() > 0 : true

```java
class BSTIterator{
    Stack<TreeNode> stk;

    public BSTIterator(TreeNode root) {
        stk = new Stack<>();
        inorderSucc(root);
    }

    public void inorderSucc(TreeNode curr){
        while(curr != null){
            stk.push(curr);
            curr = curr.left;
        }
    }

    public int next() {
        TreeNode curr = stk.pop();
        inorderSucc(curr.right);
        return curr.val;
    }

    public boolean hasNext() {
        return (stk.size() > 0);
    }
}
```

→ preprocessing → min node is not root
↳ go to leftmost node in BST (min node)

→ $O(\log_2 N) = O(h)$ in worst case

→ current ele return

→ next ele using (ceil/next larger value in stk = top())

Asymptotic
↳ worst $O(\log_2 n)$

Amortized
↳ $O(1)$ avg case

→ $O(1)$

```java
// Single BST Node
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode() {
    }

    TreeNode(int val) {
        this.val = val;
    }
}
```

```java
// Collection of Nodes
class BinarySearchTree {
    private TreeNode root;

    public void insert(int val) {
        root = insert(root, val);
    }

    private TreeNode insert(TreeNode root, int val) {
        if (root == null)
            return new TreeNode(val);

        if (val < root.val)
            root.left = insert(root.left, val);

        else if (val > root.val)
            root.right = insert(root.right, val);

        return root;
    }
}
```

```java
public static void main(String[] args) {
    BinarySearchTree tree = new BinarySearchTree();
    tree.insert(val: 70);
    tree.insert(val: 50);
    tree.insert(val: 90);
    tree.insert(val: 30);
    tree.insert(val: 40);
    tree.insert(val: 80);
    tree.insert(val: 100);
}
```

Custom class
↳ Iterable & Iteral

BST class

```java
// Collection of Nodes
class BinarySearchTree implements Iterable<Integer> {
    TreeNode root;

    public void insert(int val) {
        root = insert(root, val);
    }

    private TreeNode insert(TreeNode root, int val) {
        if (root == null)
            return new TreeNode(val);

        if (val < root.val)
            root.left = insert(root.left, val);

        else if (val > root.val)
            root.right = insert(root.right, val);

        return root;
    }

    @Override
    public Iterator<Integer> iterator() {
        BSTIterator itr = new BSTIterator(root);
        return itr;
    }
}
```

for each method

Iterator class

```java
class BSTIterator implements Iterator<Integer> {
    Stack<TreeNode> stk;

    public BSTIterator(TreeNode root) {
        stk = new Stack<>();
        inorderSucc(root);
    }

    public void inorderSucc(TreeNode curr) {
        while (curr != null) {
            stk.push(curr);
            curr = curr.left;
        }
    }

    @Override
    public Integer next() {
        TreeNode curr = stk.pop();
        inorderSucc(curr.right);
        return curr.val;
    }

    @Override
    public boolean hasNext() {
        return (stk.size() > 0);
    }
}
```

driver code

```java
public class IteratorIterable {
    Run | Debug
    public static void main(String[] args) {
        BinarySearchTree tree = new BinarySearchTree();
        tree.insert(val: 70);
        tree.insert(val: 50);
        tree.insert(val: 90);
        tree.insert(val: 30);
        tree.insert(val: 40);
        tree.insert(val: 80);
        tree.insert(val: 100);

        // Iterable: For Each Loop
        for (Integer data : tree) {
            System.out.print(data + " ");
        }
        System.out.println();

        // Iterator
        BSTIterator itr = new BSTIterator(tree.root);
        while (itr.hasNext() == true) {
            System.out.print(itr.next() + " ");
        }
    }
}
```

root should not be private

Output (Sorted → Inorder)

30 40 50 70 80 90 100

```java
// Collection of Nodes
class BinarySearchTree implements Iterable<Integer> {
    private TreeNode root;

    public void insert(int val) {
        root = insert(root, val);
    }

    private TreeNode insert(TreeNode root, int val) {
        if (root == null)
            return new TreeNode(val);

        if (val < root.val)
            root.left = insert(root.left, val);

        else if (val > root.val)
            root.right = insert(root.right, val);

        return root;
    }

    @Override
    public Iterator<Integer> iterator() {
        BSTIterator itr = new BSTIterator(root);
        return itr;
    }
}
```

```java
public static void main(String[] args) {
    BinarySearchTree tree = new BinarySearchTree();
    tree.insert(val: 70);
    tree.insert(val: 50);
    tree.insert(val: 90);
    tree.insert(val: 30);
    tree.insert(val: 40);
    tree.insert(val: 80);
    tree.insert(val: 100);

    // Iterable: For Each Loop
    for (Integer data : tree) {
        System.out.print(data + " ");
    }
    System.out.println();

    // Iterator
    Iterator<Integer> itr = tree.iterator();
    while (itr.hasNext() == true) {
        System.out.print(itr.next() + " ");
    }
}
```

```java
static class ForwardIterator{
    Stack<TreeNode> stk;

    public ForwardIterator(TreeNode root) {
        stk = new Stack<>();
        inorderSucc(root);
    }

    public void inorderSucc(TreeNode curr){
        while(curr != null){
            stk.push(curr);
            curr = curr.left;
        }
    }

    public int peek(){
        if(hasNext() == false) return 0;
        return stk.peek().val;
    }

    public int next() {
        if(hasNext() == false) return 0;
        TreeNode curr = stk.pop();
        inorderSucc(curr.right);
        return curr.val;
    }

    public boolean hasNext() {
        return (stk.size() > 0);
    }
}
```

```java
static class BackWardIterator{
    Stack<TreeNode> stk;

    public BackWardIterator(TreeNode root) {
        stk = new Stack<>();
        inorderPred(root);
    }

    public void inorderPred(TreeNode curr){
        while(curr != null){
            stk.push(curr);
            curr = curr.right;
        }
    }

    public int peek(){
        if(hasPrev() == false) return 0;
        return stk.peek().val;
    }

    public int prev() {
        if(hasPrev() == false) return 0;
        TreeNode curr = stk.pop();
        inorderPred(curr.left);
        return curr.val;
    }

    public boolean hasPrev() {
        return (stk.size() > 0);
    }
}
```

Extra Space → $O(h) = O(\log_2 n)$

Time → $O(n/2 + n/2)$

$= O(n)$

Iterate on inorder

```java
public boolean findTarget(TreeNode root, int target) {
    if(root == null || (root.left == null && root.right == null)) return false;

    ForwardIterator left = new ForwardIterator(root);
    BackWardIterator right = new BackWardIterator(root);

    while(left.hasNext() == true && right.hasPrev() == true && left.peek() < right.peek()){
        if(left.peek() + right.peek() == target) return true;
        if(left.peek() + right.peek() < target) left.next();
        else right.prev();
    }

    return false;
}
```

Exception Handling

1) Except^n vs Error
2) Exception Hierarchy
3) Compile Time (Checked) vs RunTime (Unchecked)
4) Except^n Handling :- What? Why? How?
5) Except^n object & Default Error Handling by JVM
6) 5 keywords: try, catch, finally, throw, throws
7) valid orders of try, catch & finally
8) User Defined / Custom Exceptions
9) Differences: final, finally, finalize
10) Differences: throw vs throws

Exception :→ Any abnormal behavior in your code which occurs at runtime & disturbes the normal flow by abnormal terminat (crashing).

Exception Handling :→ Alternate sequence flow provided using 5 keywords to normally terminate the program is known as exception handling.

WHAT?

```java
public static void main(String[] args) {
    System.out.println(x: "Starting Normally");

    Scanner scn = new Scanner(System.in);
    int a = scn.nextInt();
    int b = scn.nextInt();

    char op = scn.next().charAt(index: 0);

    switch (op) {
        case '+': {
            System.out.println(a + b);
            break;
        }
        case '-': {
            System.out.println(a - b);
            break;
        }
        case '*': {
            System.out.println(a * b);
            break;
        }
        case '/': {
            System.out.println(a / b);
            break;
        }
        default: {
            System.out.println(x: "Invalid Operator");
        }
    }

    System.out.println(x: "Terminating Normally");
}
```

```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
10
2
/
5
Terminating Normally
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
10
0
/
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Solution.main(Solution.java:27)
```

classname

message

Stack Trace

Runtime Except

Handle ✗

Abnormally terminate (Crash)

```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
10
abc
Exception in thread "main" java.util.InputMismatchException
        at java.base/java.util.Scanner.throwFor(Scanner.java:939)
        at java.base/java.util.Scanner.next(Scanner.java:1594)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
        at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
        at Solution.main(Solution.java:9)
```

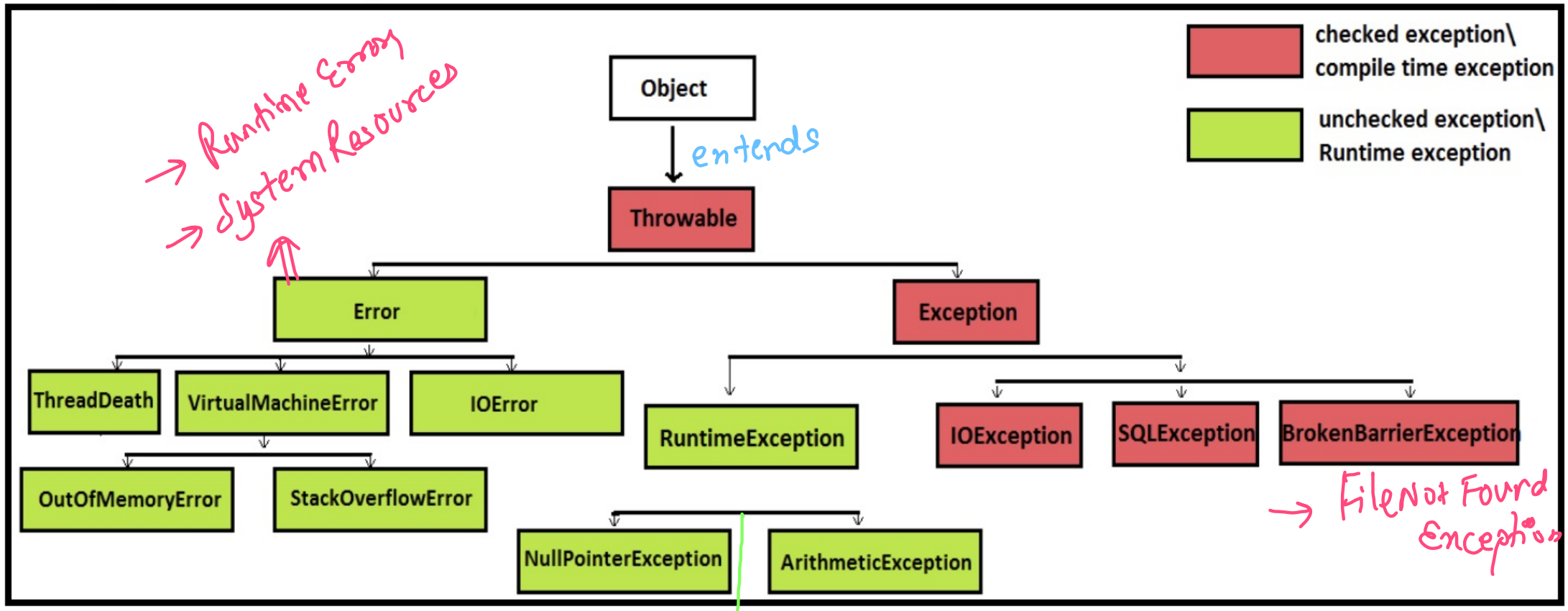
```java
String str = null;
System.out.println(str.charAt(index: 0));

System.out.println(x: "Terminating Normally");
```

```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.charAt(int)" because "<local1>" is null
        at Solution.main(Solution.java:36)
```

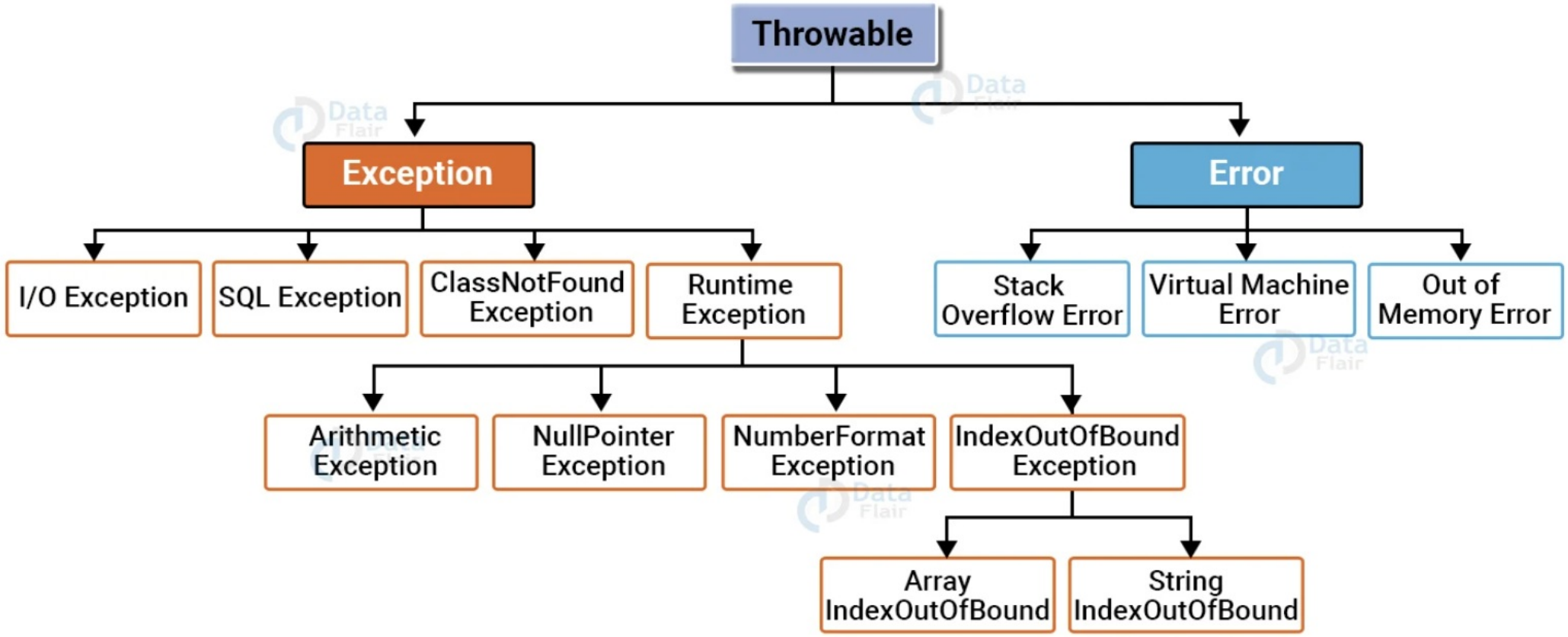
Unchecked (Runtime Excepn)

Compiler was not able to check these Exceptions

Exception hierarchy >



Hierarchy of Java Exceptions



```
FileInputStream fs = new FileInputStream(name: "d:/Archit.txt");
System.out.println(x: "Terminating Normally");
```

```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
Solution.java:39: error: unreported exception FileNotFoundException; must be caught or declared to be thrown
        FileInputStream fs = new FileInputStream("d:/Archit.txt");
                             ^
1 error
```

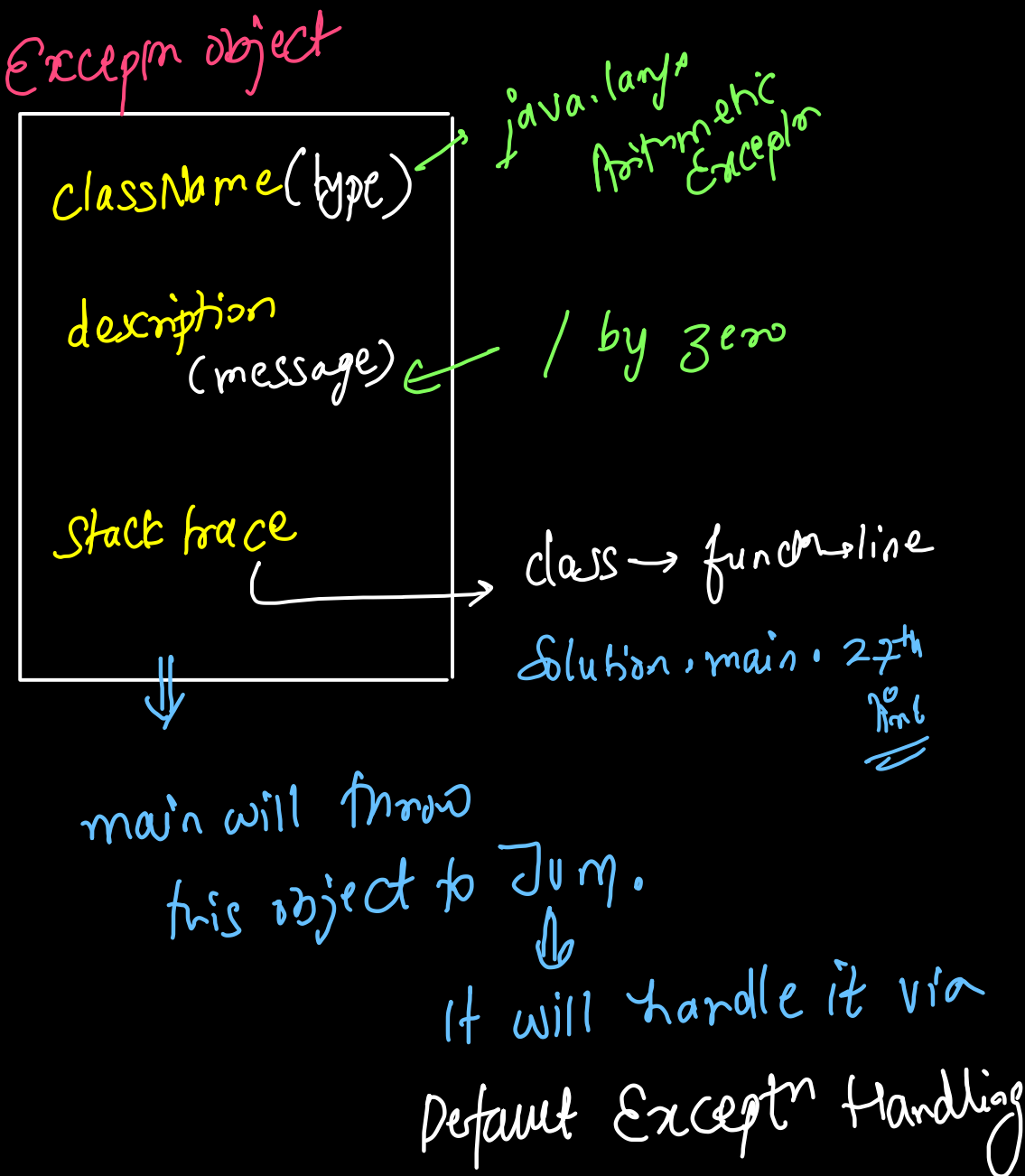
Compiler is complaining that you have not handled the FileNotFound exception so I will not compile it!

Checked / Compile Time Excepn

(Excepn will still happen at run time!)

```java
public static void main(String[] args) {
    System.out.println(x: "Starting Normally");

    Scanner scn = new Scanner(System.in);
    int a = scn.nextInt();
    int b = scn.nextInt();

    char op = scn.next().charAt(index: 0);

    switch (op) {
        case '+': {
            System.out.println(a + b);
            break;
        }
        case '-': {
            System.out.println(a - b);
            break;
        }
        case '*': {
            System.out.println(a * b);
            break;
        }
        case '/': {
            System.out.println(a / b);
            break;
        }
        default: {
            System.out.println(x: "Invalid Operator");
        }
    }

    System.out.println(x: "Terminating Normally");
}
```

← Arithmetic Exception



5 keywords

(1) try ⟶ risky code (chances of Excepn)

(2) catch ⟶ alternate flow (handling the Exception)

(3) finally ⟶ cleanup code (FileInputOutput, I/O Stream close, memory release)

(4) throw ⟶ throw custom Exception from the current fn to the calling fn!
↳ Runtime

(5) throws ⟶ calling fn gets to know called fn might throw Exception!
↳ Checked Exception

```java
public static void main(String[] args) {
    System.out.println(x: "Starting Normally");

    Scanner scn = new Scanner(System.in);
    int a = scn.nextInt();
    int b = scn.nextInt();

    char op = scn.next().charAt(index: 0);

    switch (op) {
        case '+': {
            System.out.println(a + b);
            break;
        }
        case '-': {
            System.out.println(a - b);
            break;
        }
        case '*': {
            System.out.println(a * b);
            break;
        }
        case '/': {
            try {
                System.out.println(a / b);
            } catch (ArithmeticException e) {
                System.out.println(x: "Division by Zero Not Allowed");
            }
            break;
        }
        default: {
            System.out.println(x: "Invalid Operator");
        }
    }

    System.out.println(x: "Terminating Normally");
}
}
```

```
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
10
2
/
5
Terminating Normally
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
10
0
/
Division by Zero Not Allowed
Terminating Normally
```

```java
System.out.println(x: "Starting Normally");

try {
    Scanner scn = new Scanner(System.in);
    int a = scn.nextInt();
    int b = scn.nextInt();
    char op = scn.next().charAt(index: 0);

    switch (op) {
        case '+': {
            System.out.println(a + b);
            break;
        }
        case '-': {
            System.out.println(a - b);
            break;
        }
        case '*': {
            System.out.println(a * b);
            break;
        }
        case '/': {
            System.out.println(a / b);
            break;
        }
        default: {
            System.out.println(x: "Invalid Operator");
        }
    }
} catch (Exception e) {
    System.out.println(e);
}

System.out.println(x: "Terminating Normally");
```

```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
10
abc
java.util.InputMismatchException
Terminating Normally
```

```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
10
0
/
java.lang.ArithmeticException: / by zero
Terminating Normally
```

```java
System.out.println(x: "Starting Normally");

try {
    System.out.println(x: "Inside Try Block Before Input");
    Scanner scn = new Scanner(System.in);
    int a = scn.nextInt();
    int b = scn.nextInt();
    char op = scn.next().charAt(index: 0);
    System.out.println(x: "Inside Try Block After Input");

    switch (op) {
        case '+': {
            System.out.println(a + b);
            break;
        }
        case '-': {
            System.out.println(a - b);
            break;
        }
        case '*': {
            System.out.println(a * b);
            break;
        }
        case '/': {

            System.out.println(x: "Inside Swith Case Before Division");
            System.out.println(a / b);
            System.out.println(x: "Inside Swith Case After Division");
            break;
        }
        default: {
            System.out.println(x: "Invalid Operator");
        }
    }

    System.out.println(x: "Inside Try After Switch Case");
} catch (Exception e) {
    System.out.println(x: "Inside Catch");
    System.out.println(e);
}

System.out.println(x: "Terminating Normally");
```

```
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
Inside Try Block Before Input
10
2
/
Inside Try Block After Input
Inside Swith Case Before Division
5
Inside Swith Case After Division
Inside Try After Switch Case
Terminating Normally
```

No Except | e

```
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
Inside Try Block Before Input
10
abc
Inside Catch
java.util.InputMismatchException
Terminating Normally
```

Input Except | e

```
architaggarwal@Archits-MacBook-Air System Design % java Solution
Starting Normally
Inside Try Block Before Input
10
0
/
Inside Try Block After Input
Inside Swith Case Before Division
Inside Catch
java.lang.ArithmeticException: / by zero
Terminating Normally
```

Division Except | b

```java
public static void main(String[] args) {
    try {
        Integer a = Integer.parseInt(args[0]);
        Integer b = Integer.parseInt(args[1]);

        System.out.println(a / b);
    } catch (Exception e) {
        System.out.println(e);
    }
}
```

```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution 10 2
5
architaggarwal@Archits-MacBook-Air System Design % java Solution 10
java.lang.ArrayIndexOutOfBoundsException: Index 1 out of bounds for length 1
architaggarwal@Archits-MacBook-Air System Design % java Solution 10 abc
java.lang.NumberFormatException: For input string: "abc"
architaggarwal@Archits-MacBook-Air System Design % java Solution 10 0
java.lang.ArithmeticException: / by zero
```

```java
public static void main(String[] args) {
    try {
        Integer a = Integer.parseInt(args[0]);
        Integer b = Integer.parseInt(args[1]);

        System.out.println(a / b);
    } catch (ArithmeticException e) {
        System.out.println(x: "Division by Zero Not Allowed");
    } catch (NumberFormatException e) {
        System.out.println(x: "Please pass integers only");
    } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println(x: "Please pass atleast 2 parameters");
    } catch (Exception e) {
        System.out.println(x: "Some Other Expection Occured");
    }
}
```

multiple catch statements!

```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution 10 2
5
architaggarwal@Archits-MacBook-Air System Design % java Solution 10
Please pass atleast 2 parameters
architaggarwal@Archits-MacBook-Air System Design % java Solution 10 abc
Please pass integers only
architaggarwal@Archits-MacBook-Air System Design % java Solution 10 0
Division by Zero Not Allowed
```

Valid orders of try, catch

```
try { }
```

✗ only try not possible

```
catch { }
```

✗ only catch not possible

```
try {
    try { }
    catch (e1) { }
}
catch { }
```

```
try { }
catch (e1) { }
```

both are valid

```
try { }
catch (e1) { }
catch (e2) { }
  :
catch (ek) { }
```

✓ multiple catch

```
try { }
catch {
    try { }
    catch (e) { }
}
```

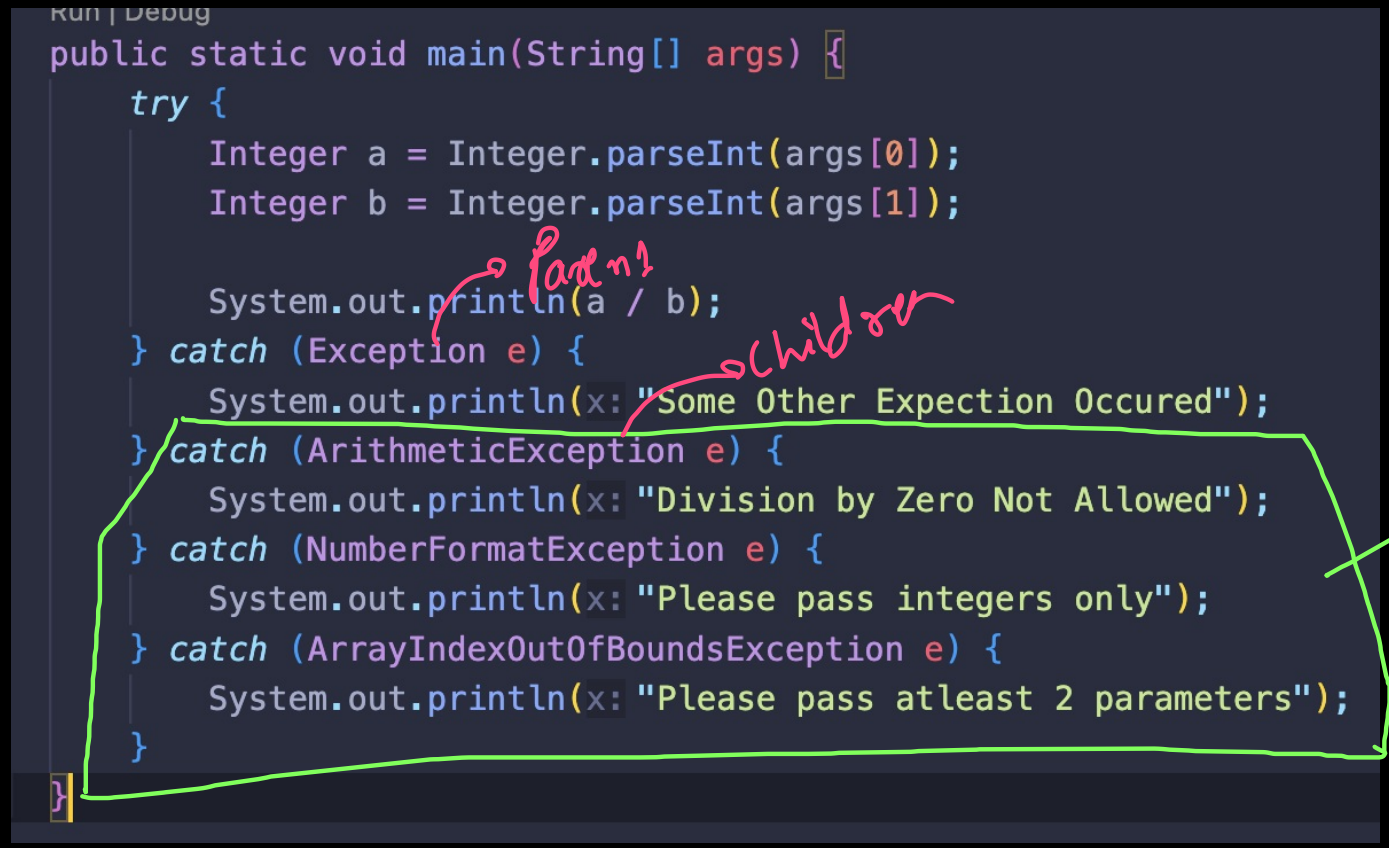
```
catch ( ) { }
    try { }
```

✗ not possible

```
try { ... }
Syso("In b/w try & catch")
catch ( ) { }
```

✗ not possible

✓ Nested Try Loop


Run | Debug
public static void main(String[] args) {
try {
Integer a = Integer.parseInt(args[0]);
Integer b = Integer.parseInt(args[1]);
System.out.println(a / b);
} catch (Exception e) {
System.out.println(x: "Some Other Expection Occured");
} catch (ArithmeticException e) {
System.out.println(x: "Division by Zero Not Allowed");
} catch (NumberFormatException e) {
System.out.println(x: "Please pass integers only");
} catch (ArrayIndexOutOfBoundsException e) {
System.out.println(x: "Please pass atleast 2 parameters");
}
}
Parent
Children
Unreachable code
↓
System/Compilation error

```java
Run | Debug
public static void main(String[] args) {
    try {
        Scanner scn = new Scanner(System.in);
        int a = scn.nextInt();
        int b = scn.nextInt();
        System.out.println(a / b);
        scn.close(); // Scanner Object -> Memory Release, Input Stream Close
    } catch (Exception e) {
        System.out.println("Exception is Handled : " + e);
    }
}
```

→ scn will only close if Except is not occured

```java
try {
    Scanner scn = new Scanner(System.in);
    int a = scn.nextInt();
    int b = scn.nextInt();
    System.out.println(a / b);
    scn.close();
} catch (Exception e) {
    System.out.println("Exception is Handled : " + e);
    scn.close();
}
```

→ Code redundancy

```java
Scanner scn = new Scanner(System.in);
try {
    int a = scn.nextInt();
    int b = scn.nextInt();
    System.out.println(a / b);

} catch (Exception e) {
    System.out.println("Exception is Handled : " + e);
    System.out.println(1 / 0);
} finally {
    System.out.println(x: "Finally Block Executed: Clean up Code");

}

scn.close();
```

This line will not execute if catch will run because catch have excepn (abnormal termination)

```java
Scanner scn = new Scanner(System.in);
try {
    int a = scn.nextInt();
    int b = scn.nextInt();
    System.out.println(a / b);

} finally {
    System.out.println(x: "Finally Block Executed: Clean up Code");
    scn.close();
}
```

```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
10 5
2
Finally Block Executed: Clean up Code
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
10 0
Finally Block Executed: Clean up Code
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at Solution.main(Solution.java:79)
```

→ finally executed even during abnormal terminal!

```java
Scanner scn = new Scanner(System.in);
try {
    int a = scn.nextInt();
    int b = scn.nextInt();
    System.out.println(a / b);

} catch (Exception e) {
    System.out.println("Exception is Handled : " + e);
} finally {
    System.out.println(x: "Finally Block Executed: Clean up Code");
    scn.close();
}
```

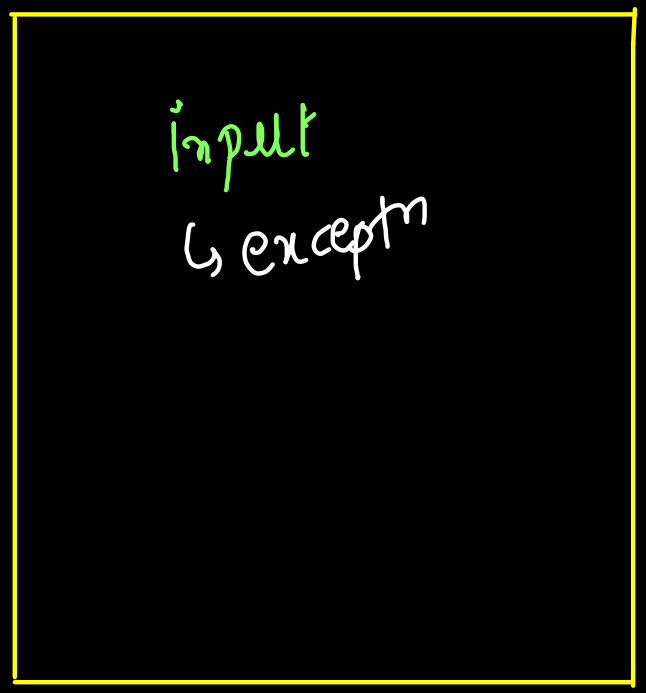
```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
10 2
5
Finally Block Executed: Clean up Code
architaggarwal@Archits-MacBook-Air System Design % java Solution
10 0
Exception is Handled : java.lang.ArithmeticException: / by zero
Finally Block Executed: Clean up Code
```

```java
Run | Debug
public static void main(String[] args) {
    try {
        FileInputStream scn = new FileInputStream(name: "d:/abc.txt");
    } catch (Exception e) {
        System.out.println("Exception Occured: " + e);
    }
}
```

```
architaggarwal@Archits-MacBook-Air System Design % javac Solution.java
architaggarwal@Archits-MacBook-Air System Design % java Solution
Exception Occured: java.io.FileNotFoundException: d:/abc.txt (No such file or directory)
```

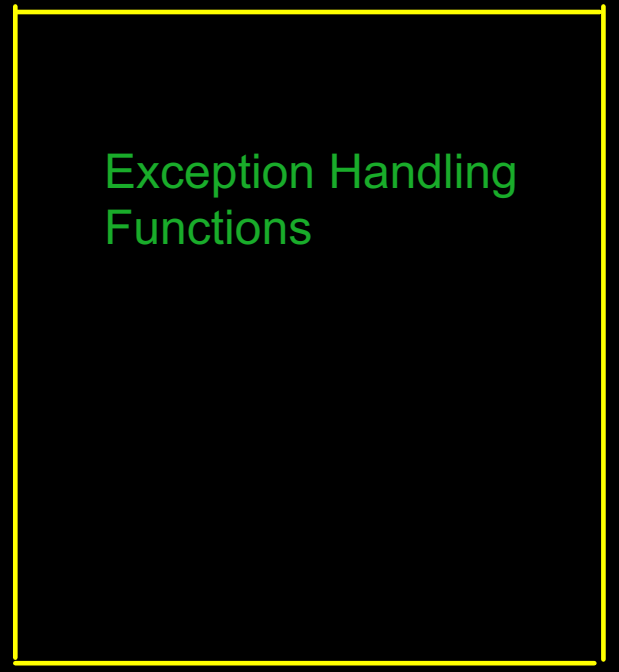
javac : accepts handled using try & catch : no problem

Checked excepn occured at runtime!



Driver
(main)

Calculator



Exception Handler

```java
class Calculator {
    public int add(int a, int b) {
        return a + b;
    }

    public int subtract(int a, int b) {
        return a - b;
    }

    public int divide(int a, int b) {
        return a / b;
    }

    public int multiply(int a, int b) {
        return a * b;
    }
}
```

Business Logic (App)

throws ArithmeticException (Optional)

```java
class Driver {
    Run | Debug
    public static void main(String[] args) {
        Calculator calc = new Calculator();

        Scanner scn = new Scanner(System.in);
        int a = 0, b = 0;
        try {
            a = scn.nextInt();
            b = scn.nextInt();

            try {
                int res = calc.divide(a, b);
                System.out.println(res);
            } catch (ArithmeticException e) {
                ExceptionHandling.calculatorException();
            }

        } catch (Exception e) {
            ExceptionHandling.inputOutputException();
        }

    }
}
```

Driver Code (Client side)

```java
class ExceptionHandling {
    public static void inputOutputException() {
        System.out.println(x: "Please Provide Input Again");
    }

    public static void calculatorException() {
        System.out.println(x: "Division by 0 Not Allowed");
    }
}
```

Except Handler Servic