

try {

}

catch (ExceptionName

e) {

}

multiple catch

try {

}

catch () {

}

catch () {

}

try {

= doubtful code

}

catch ([↓] _____ e) {

 Sout(" _____ ");

}

try {
 finally {
 }
}
catch {
}
}

The diagram illustrates the mapping of a try-catch-finally block to a try-finally construct. A large curved arrow originates from the 'try' block and points to the 'try' part of the 'try-finally' construct. Another large curved arrow originates from the 'catch' block and points to the 'finally' part of the 'try-finally' construct. The 'finally' block in the original code is circled, and an arrow points from it to the 'finally' block in the 'try-finally' construct.

nested if/for/

nested try & catch
— x —

(try {
= try {
} catch {
}) →

} — ,

try {

try {

} catch () {

}

}

OpPs

Class {

Private,

getter/setter,

→ Const ->

}

mn
(2)

Sum of two no

(oh)

```
class SumOfTwo {  
    private int n1, n2;
```

getter setter

```
void sum() {
```

```
    cout << n1 + n2 << endl;  
}
```

Input

class Main {

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

```
        Scanner sc = new Scanner(System.in);
```

```
        int n1 = sc.nextInt();
```

```
        int n2 = sc.nextInt();
```

```
        SumOfTwo s = new SumOfTwo(n1, n2);  
        s.sum();  
    }
```