Multi-Catch Block.

```
try {
    int a = sc.next()          'i' 'abc'
    a/b     b=0
} catch (ArithmeticException e) {
    — 8 line
} catch (InputMismatchException e) {
    — 8 line
} catch (Exception e) {
    —
}
```

— 1.7 Jdk. → Shorthand

Exception Object: name, description → get

```
try {

} catch ( ArithmeticException | InputMismatchException
        | ArrayIndexOutOfBoundException e) {
    if (e.name == "ArithmeticException"){
    }
}
```

Notes:

→ In multicatch block we can catch multiple exceptions in a single catch back as given in above

→ This approach is usually used when multiple exceptions have the same handling code

→ The exceptions written with in this catch block paranthesis should not have "IS-A" relation

P ⟷ C

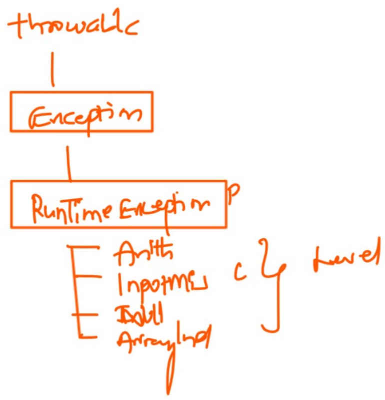
IOException | FileNotFoundException

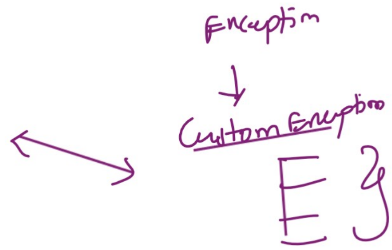
P | C

RuntimeException | ArithmeticException

C | P

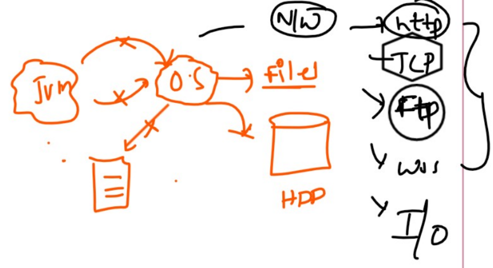
ArithmeticException | Exception



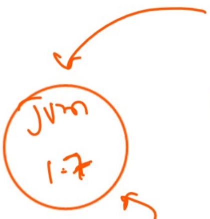


finally

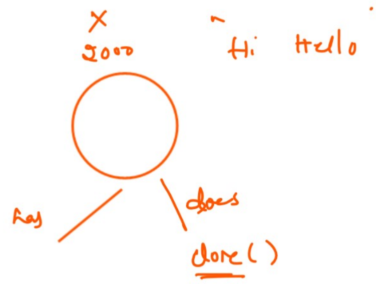
—1.7




```
class Alpha {
    p s v main(String args[]) {
①  →    FileReader reader = null;
        try {
            reader = new FileReader("c:/Desktop/a.txt");
②
        } catch (FileNotFoundException e) {
            ==
        } finally {
            if (reader != null) {
                reader.close()
            }
        }
    }
```



db
htt. req



Try with Resource

>1.7

```
class Alpha {
    p s r main(String args[]) throws FileNotFoundException {
        try( FileReader reader = new FileReader("c: Desktop/a.txt") ) {
            ==
        } catch (Exception e) {
        }
    }
}
```

try(R1){
==
}

try{
==
}

Student implements AutoCloseable
close(){
}}

Notes (interview → Reason)

→ with in try with resource block we can create the object of only such class which have implemented autocloseable interface

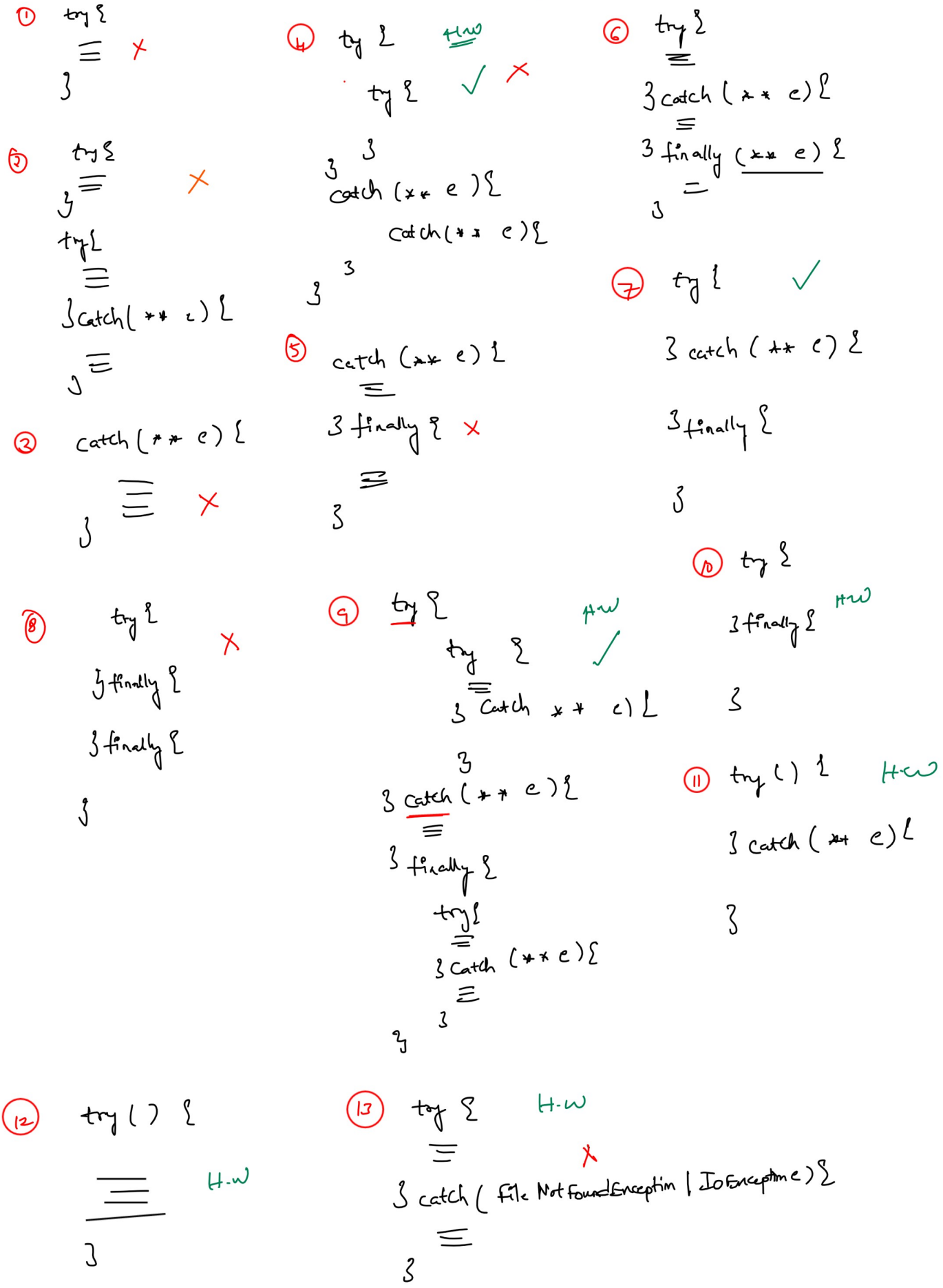
Notes:

→ We can create multiple resources in a given try block

```
try (R1; R2; R3; R4;) {
} catch (Exception e) {
}
```

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

Cheat Sheet — possibilities



Some Common Exception List

(1) InputMismatchException (unchecked)

(2) ArithmeticException (Unchecked)

(3) NullPointerException (Unchecked) — H.W

(4) ArrayIndexOutOfBoundsException (Unchecked)

(5) StringIndexOutOfBoundsException (Unchecked)

(6) NumberFormatException (unchecked)

Integer.parseInt("abc") 10 ← "10"

↑ IllegalArgument

(7) IllegalArgumentException (unchecked) [parent of NumberFormatExcept]

(8) ClassNotFoundException (H.W) java A

(9) ClassCastException (H.W)

Object o = new Object()

String s = (String) o

String s = new String("ABC");

Object o = (Object) s

Parent ← child

child ← Parent

(10) FileNotFoundException

(11) IOException

(12) SQLException

(13) StackOverflowErr

(14) OutOfMemoryError

} (Checked)

Class A {

static void add() {

S.o.pl("add is called")

}

}

HW

A obj = null;

obj.add();