

1. Which of the following is the **superclass** of all exceptions in Java?

- a) Throwable
 - b) Exception**
 - c) RuntimeException
 - d) Error
-

2. What is the parent class of both `Exception` and `Error`? **a) Throwable**

- b) Object
 - c) RuntimeException
 - d) IOException
-

3. Which of these is a **checked exception**? a) `NullPointerException`

- b) `ArrayIndexOutOfBoundsException`
 - c) `ArithmeticException`
 - d) `IOException`**
-

4. What happens if a checked exception is not handled or declared in a method?

- a) Program crashes at runtime
 - b) Compiler shows an error**
 - c) Method automatically handles it
 - d) JVM ignores it
-

5. Which of the following is an **unchecked exception**? a) `SQLException`

- b) `ClassNotFoundException`
 - c) `FileNotFoundException`
 - d) `NumberFormatException`**
-

6. Errors in Java, like `OutOfMemoryError`, are:

- a) Meant to be caught using try-catch
 - b) Subclasses of `Exception`
 - c) Non-recoverable**
 - d) Checked exceptions
-

7. Which keyword is used to handle exceptions in Java?

- a) `final`
 - b) `catch`
 - c) `throw`
 - d) `try`**
-

8. Which block must always follow a try block?

- a) `catch`**
 - b) `throw`
 - c) `finally`
 - d) None
-

9. How many `catch` blocks can follow a single `try` block?

- a) Only one
 - b) Two
 - c) As many as needed**
 - d) None
-

10. What will happen if an exception is thrown but not caught?

- a) Program continues normally
 - b) Compiler fixes it
 - c) Program terminates abnormally
 - d) JVM ignores it
-

11. What kind of exception is `NullPointerException`? a) Checked

- b) Unchecked
 - c) User-defined
 - d) Compile-time
-

12. Which one of the following is NOT a subclass of `RuntimeException`? a)

`ArithmeticException`

- b) `FileNotFoundException`
 - c) `ArrayIndexOutOfBoundsException`
 - d) `NumberFormatException`
-

13. In Java, errors and exceptions are part of which hierarchy?

- a) `Object`
 - b) `Throwable`
 - c) `Exception`
 - d) `RuntimeException`
-

14. Which of the following is true about `try` and `catch` blocks?

- a) Only `catch` is required
 - b) Only `try` is required
 - c) `try` must be followed by either `catch` or `finally`
 - d) Both `try` and `catch` are optional
-

15. You write a method that reads a file. Which kind of exception must you handle or declare?

- a) `FileNotFoundException`
 - b) `NullPointerException`
 - c) `ArithmeticException`
 - d) `ArrayIndexOutOfBoundsException`
-

16. When multiple `catch` blocks are used, how are exceptions matched?

- a) Bottom-up
 - b) Randomly
 - c) Top-down, from specific to general
 - d) Order does not matter
-

17. What happens if the general exception class is written before a specific one in multiple `catch` blocks?

- a) Compiler error
 - b) Runtime error
 - c) JVM skips the specific one
 - d) No issue
-

18. Can a `try` block be used without a `catch` block?

- a) No
- b) Yes, only with a `finally` block

- c) Yes, always
- d) Only inside a loop

19. Which of the following represents a scenario where you should NOT use try-catch?

- a) Null input
- b) Parsing a file
- c) Dividing two integers
- d) Fixing an `OutOfMemoryError`

20. You have the following code:

```
try {
    int a = 5 / 0;
} catch (ArithmeticException e) {
    System.out.println("Arithmetic error!");
} catch (Exception e) {
    System.out.println("General error!");
}
```

What will be the output?

- a) `Arithmetic error!`
- b) General error!
- c) Compilation error
- d) Runtime exception not caught

Code Based Questions

1.

```
try {
    String str = null;
    System.out.println(str.length());
} catch (ArithmeticException e) {
    System.out.println("Arithmetic error!");
} catch (NullPointerException e) {
    System.out.println("Null pointer error!");
}
```

- a) Arithmetic error!
- b) `Null pointer error!`
- c) Compilation error
- d) No output

2.

```
try {
    int[] arr = new int[3];
    System.out.println(arr[5]);
} catch (ArrayIndexOutOfBoundsException e) {
    System.out.println("Array error!");
} catch (Exception e) {
    System.out.println("General error!");
}
```

- a) Array error! b) General error! c) Compilation error
d) No output
-

3.

```
try {  
    int a = Integer.parseInt("abc");  
} catch (NumberFormatException e) {  
    System.out.println("Number format error!");  
} catch (Exception e) {  
    System.out.println("General error!");  
}
```

- a) Number format error! b) General error! c) Compilation error
d) No output
-

4.

```
try {  
    int result = 10 / 2;  
    System.out.println("Result: " + result);  
} catch (ArithmeticException e) {  
    System.out.println("Divide by zero!");  
}
```

- a) Result: 5
b) Divide by zero! c) Compilation error
d) No output
-

5.

```
try {  
    throw new Exception("Custom exception");  
} catch (RuntimeException e) {  
    System.out.println("Runtime exception caught");  
} catch (Exception e) {  
    System.out.println("General exception caught");  
}
```

- a) Runtime exception caught
b) General exception caught
c) Compilation error
d) No output
-

6.

```
try {  
    int[] nums = null;  
    nums[0] = 10;  
} catch (ArrayIndexOutOfBoundsException e) {  
    System.out.println("Index problem");  
} catch (NullPointerException e) {
```

```
System.out.println("Null reference issue");  
}
```

- a) Index problem
 - b) Null reference issue**
 - c) Compilation error
 - d) No output
-

7.

```
try {  
    String s = "123";  
    int x = Integer.parseInt(s);  
    System.out.println(x / 0);  
} catch (NumberFormatException e) {  
    System.out.println("Invalid number");  
} catch (ArithmeticException e) {  
    System.out.println("Division by zero");  
}
```

- a) Invalid number
 - b) Division by zero**
 - c) Compilation error
 - d) No output
-

8.

```
try {  
    int x = 5 / 0;  
} catch (Exception e) {  
    System.out.println("Exception caught");  
} catch (ArithmeticException e) {  
    System.out.println("Arithmetic caught");  
}
```

- a) Exception caught
 - b) Arithmetic caught
 - c) Compilation error**
 - d) No output
-

9.

```
try {  
    System.out.println("Start");  
} catch (Exception e) {  
    System.out.println("Error occurred");  
}
```

- a) Start**
 - b) Error occurred
 - c) Compilation error
 - d) No output
-

10.

```
try {  
    String s = null;  
    System.out.println("Length: " + s.length());  
} catch (Exception e) {  
    System.out.println("Handled in general catch");  
}
```

- a) Length: 0
 - b) Compilation error
 - c) Handled in general catch**
 - d) NullPointerException
-