# Runtime Exceptions

En Java, las excepciones de tipo **RuntimeException** son aquellas que ocurren durante la ejecución del programa y que no necesitan ser declaradas en un bloque throws o manejadas explícitamente en un try-catch. Estas excepciones son generalmente causadas por errores de lógica del programa que podrían haberse evitado. A continuación, vemos algunas de las más comunes, junto con su propósito:

### 1. `ArithmeticException`

- **Propósito**: Se lanza cuando ocurre un error aritmético, como una división por cero.
- **Ejemplo**:

```java
int result = 10 / 0;  // Lanza ArithmeticException
```

### 2. `ArrayIndexOutOfBoundsException`

- **Propósito**: Ocurre cuando se intenta acceder a una posición inválida de un array, es decir, fuera de sus límites.
- **Ejemplo**:

```java
int[] arr = new int[5];
arr[10] = 25;  // Lanza ArrayIndexOutOfBoundsException
```

### 3. `NullPointerException`

- **Propósito**: Se lanza cuando se intenta acceder a un miembro (método o atributo) de un objeto que es `null`.
- **Ejemplo**:

```java
String s = null;
int length = s.length();  // Lanza NullPointerException
```

### 4. `ClassCastException`

- **Propósito**: Se lanza cuando se intenta realizar un casteo (conversión) de un objeto a un tipo incompatible.
- **Ejemplo**:

```
Object obj = "Hello";
Integer num = (Integer) obj;  // Lanza
ClassCastException
```

### 5. `IllegalArgumentException`

- **Propósito**: Se lanza cuando se pasa un argumento inválido a un método.
- **Ejemplo**:

```
Thread t = new Thread();
t.setPriority(11);  // Lanza IllegalArgumentException
(los valores válidos son 1 a 10)
```

### 6. `NumberFormatException`

- **Propósito**: Se lanza cuando se intenta convertir una cadena que no puede ser interpretada como un número.
- **Ejemplo**:

```
int num = Integer.parseInt("abc");  // Lanza
NumberFormatException
```

### 7. `IndexOutOfBoundsException`

- **Propósito**: Es la clase base para excepciones que indican que un índice está fuera de los límites válidos. Las subclases comunes son `ArrayIndexOutOfBoundsException` y `StringIndexOutOfBoundsException`.
- **Ejemplo**:

```
String s = "Java";
char ch = s.charAt(10);  // Lanza
StringIndexOutOfBoundsException
```

### 8. `IllegalStateException`

- **Propósito**: Ocurre cuando el estado actual de un objeto no es adecuado para la operación que se está intentando realizar.
- **Ejemplo**:

```
Iterator<String> iterator = new
ArrayList<String>().iterator();
iterator.remove();  // Lanza IllegalStateException (el
iterador no tiene elementos)
```

### 9. `UnsupportedOperationException`

- **Propósito**: Se lanza cuando se invoca un método que no está soportado por la implementación actual.
- **Ejemplo**:

```
List<String> list = Arrays.asList("a", "b", "c");
list.add("d");  // Lanza UnsupportedOperationException
```

### 10. `ConcurrentModificationException`

- **Propósito**: Ocurre cuando un objeto es modificado concurrentemente de una manera no permitida, como cuando se modifica una colección mientras se itera sobre ella.
- **Ejemplo**:

```
List<String> list = new ArrayList<>();
list.add("a");
list.add("b");

for (String s : list) {
    list.remove(s);  // Lanza
ConcurrentModificationException
}
```

### 11. `StackOverflowError`

- **Propósito**: Aunque no es una excepción, sino un `Error`, suele clasificarse junto con las excepciones en tiempo de ejecución. Se lanza cuando la pila del programa se llena, generalmente debido a una recursión infinita.
- **Ejemplo**:

```java
public void recursive() {
    recursive();  // Lanza StackOverflowError
}
```

### 12. `NegativeArraySizeException`

- **Propósito**: Se lanza cuando se intenta crear un array con un tamaño negativo.
- **Ejemplo**:

```java
int[] arr = new int[-5];  // Lanza NegativeArraySizeException
```

Las excepciones en tiempo de ejecución en Java indican errores que el compilador no detecta, pero que el programa puede lanzar durante su ejecución debido a fallos en la lógica o en las operaciones permitidas. Cada una de estas excepciones tiene un propósito específico y está diseñada para ayudar a identificar fallos comunes en la ejecución del código.