Definición de excepciones personalizadas en Kotlin

Definir excepciones personalizadas en Kotlin es un proceso fundamental cuando deseas manejar situaciones excepcionales específicas en tu código de manera más efectiva. Aquí hay una breve teoría sobre la definición de excepciones personalizadas en Kotlin:

1. ¿Qué es una excepción personalizada?

Una excepción personalizada es una clase que extiende la clase Exception o una de sus subclases (por ejemplo, RuntimeException). Estas clases se utilizan para representar situaciones excepcionales específicas que pueden ocurrir en tu aplicación.

2. Por qué crear excepciones personalizadas

- Proporcionan información detallada sobre el error, lo que facilita la depuración.
- Permiten capturar y manejar errores de manera específica en lugar de usar excepciones genéricas.
- Ayudan a mejorar la legibilidad del código al identificar claramente las situaciones excepcionales.

3. Definición de una excepción personalizada en Kotlin

Para crear una excepción personalizada, sigue estos pasos:

1. Define una clase que herede de Exception o una de sus subclases.
2. Puedes agregar propiedades y métodos personalizados para proporcionar información adicional sobre la excepción.
3. Debes proporcionar un constructor que llame al

constructor de la superclase y pase un mensaje descriptivo como parámetro.

```
class MiExcepcion : Exception("Este es un mensaje descriptivo")
```

4. Lanzar una excepción personalizada

Puedes lanzar una excepción personalizada utilizando la palabra clave throw seguida de una instancia de tu excepción personalizada.

```
throw MiExcepcion()
```

5. Capturar y manejar excepciones personalizadas

Para capturar una excepción personalizada, puedes utilizar un bloque try-catch y especificar el tipo de excepción que deseas capturar.

```
try {
   // Código que puede lanzar una excepción
} catch (e: MiExcepcion) {
   // Manejar la excepción personalizada
}
```

En resumen, definir excepciones personalizadas en Kotlin te permite gestionar de manera más efectiva situaciones excepcionales específicas en tu código, proporcionando información detallada sobre los errores y permitiendo un manejo más específico de las excepciones. Esto contribuye a una programación más robusta y mantenible.

6. Ejemplos

Desarrollar una clase llamada MiClase que tenga un método llamado miMetodo que indique que puede lanzar la nueva excepción que hemos creado. Este método devolverá un int. Calculará mediante un random ceros o unos. Si sale un cero, lanzará la excepción. Si sale un uno lo devolverá.

[crayon-6a9d9ac14542e013097165/]

Cree e inicialice un objeto de la clase MiClase. Llame en un bucle que itere 20 veces al método miMetodo del objeto creado anteriormente e imprima el resultado de dicha ejecución: el int que devuelve o el mensaje de la excepción capturada.

[crayon-6a9d9ac145435393636498/]