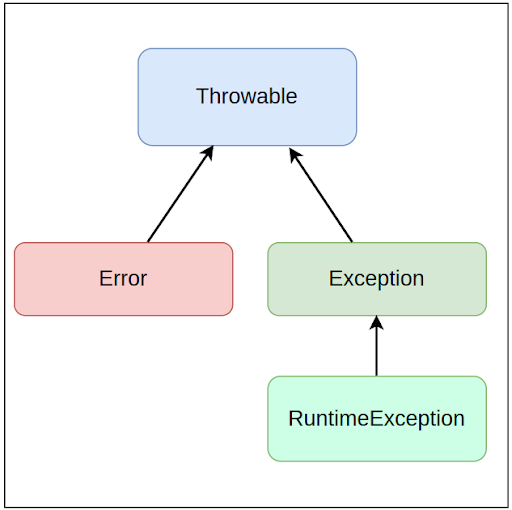
**Hierarquia de excepitons no java**

No Java, as exceções são organizadas em uma hierarquia de classes. Todas as exceções são subclasses da classe **Throwable**, sendo que ela possui duas subclasses principais: **Exception** e **Error**.



As exceções que herdam da classe **Exception** são chamadas de exceções verificadas (checked exceptions). Isso significa que essas exceções devem ser tratadas explicitamente em um bloco try-catch ou declaradas em uma cláusula throws na assinatura do método. Um exemplo é a classe de exceção IOException, que indica algum problema relacionado com leitura/escrita de dados.

As exceções que herdam da classe **Error** representam erros irrecuperáveis pelo sistema, como falta de memória ou falhas internas. Um exemplo é a classe de exceção OutOfMemoryError, que indica que o Java não conseguiu memória suficiente do sistema operacional para executar corretamente a aplicação.

Além disso, existe ainda a classe de exceção **RuntimeException**, que é uma subclasse direta de Exception, e as classes que herdam dela são chamadas de exceções não verificadas (unchecked exception). As exceções não verificadas indicam erros lógicos no código, como a NullPointerException, que indica o acesso a algum atributo ou método de um objeto que é nulo, ou seja, que não foi instanciado ou foi atributo ao valor null.

Ao lidar com exceções em um bloco try-catch, é importante considerar a hierarquia de exceções. É possível capturar exceções de uma classe mãe em um bloco catch que captura exceções de uma classe filha. No entanto, o inverso não é possível. Isso significa que, se um bloco catch captura exceções de uma classe filha, ele não será capaz de capturar exceções de uma classe pai.