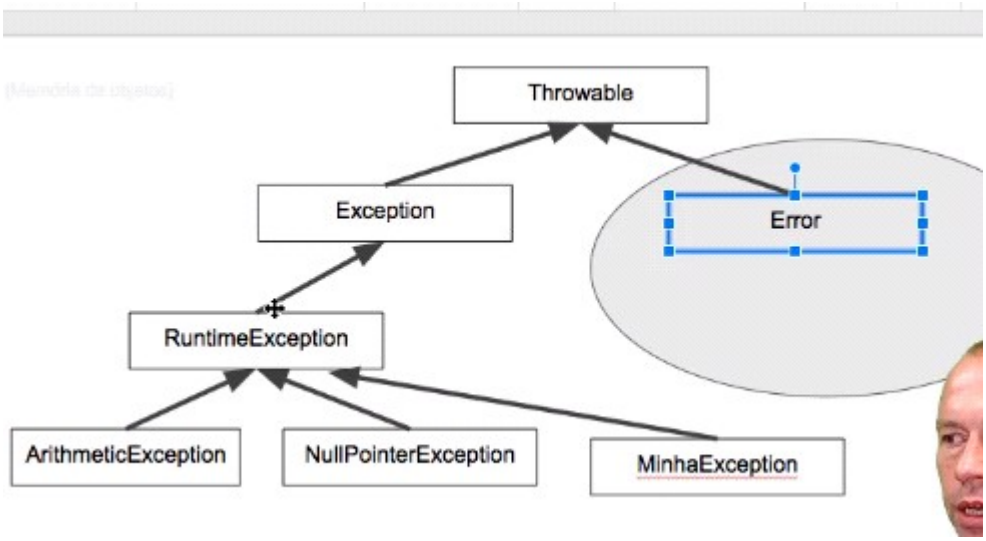
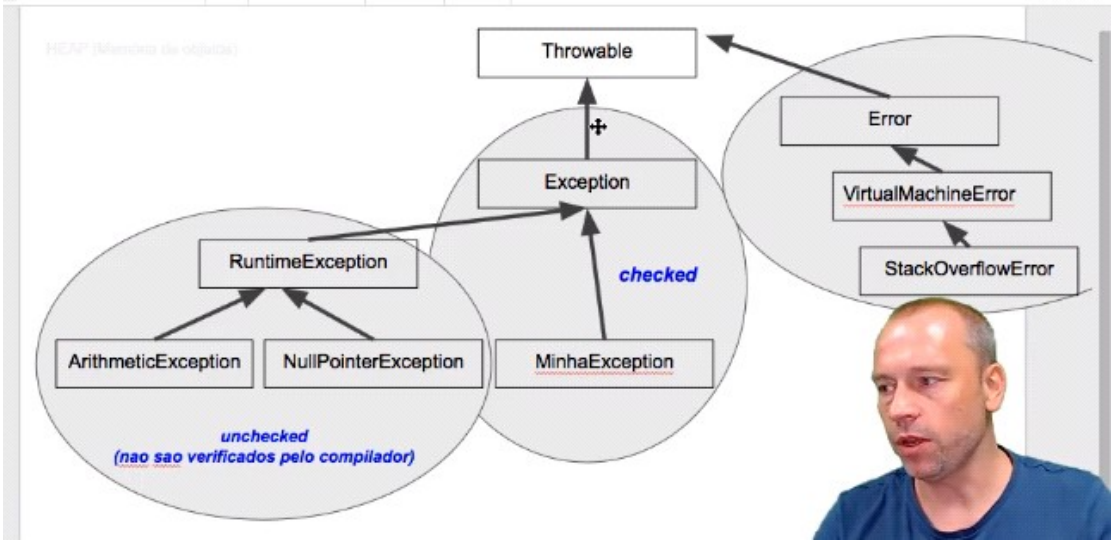
Hierarquia de exceções - com extends é possivel criar sua propria exceção



-Error seriam erros internos do JVM (falta de memória da máquina por exemplo), não temo acesso direto

- Melhor herdar o runtime do que o exception para facilitar (caso tenha herdado no exception, é necessário deixar explictio o throws na assinatura do metodo do nosso programa - compilador precisa verificar métodos que herdam o exception)



-Checker = precisa deixar explicito, obrigatório usar o try catch quando chamar esse método

```
private static void metodo2() throws MinhaExcecao {
    System.out.println("Ini do metodo2");
    throw new MinhaExcecao("deu muito errado");

Conta c = new Conta();
try {
    c.deposita();
} catch(MinhaExcecao ex) {
    System.out.println("tratamento ....");
}
```

-Unchecked = não precisa deixar explicito

Existe uma hierarquia grande de classes que representam exceções. Por exemplo, ArithmeticException é filha de RuntimeException, que herda de Exception, que por sua vez é filha da classe mais ancestral das exceções, Throwable. Conhecer bem essa hierarquia significa saber utilizar exceções em sua aplicação.

Throwable é a classe que precisa ser extendida para que seja possível jogar um objeto na pilha (através da palavra reservada throw)

É na classe Throwable que temos praticamente todo o código relacionado às exceções, inclusive getMessage() e printStackTrace(). Todo o resto da hierarquia apenas possui algumas sobrecargas de construtores para comunicar mensagens específicas

A hierarquia iniciada com a classe Throwable é dividida em exceções e erros. Exceções são usadas em códigos de aplicação. Erros são usados exclusivamente pela máquina virtual.

Classes que herdam de Error são usadas para comunicar erros na máquina virtual. Desenvolvedores de aplicação não devem criar erros que herdam de Error.

StackOverflowError é um erro da máquina virtual para informar que a pilha de execução não tem mais memória.

Exceções são separadas em duas grandes categorias: aquelas que são obrigatoriamente verificadas pelo compilador e as que não são verificadas.

As primeiras são denominadas checked e são criadas através do pertencimento a uma hierarquia que não passe por RuntimeException.

As segundas são as unchecked, e são criadas como descendentes de RuntimeException.