Resumo da aula

- Solução 1 (muito ruim): lógica de validação no programa principal
  - Lógica de validação não delegada à reserva
- Solução 2 (ruim): método retornando string
  - A semântica da operação é prejudicada
    - Retornar string não tem nada a ver com atualização de reserva
    - E se a operação tivesse que retornar um string?
  - Ainda não é possível tratar exceções em construtores
  - Ainda não há auxílio do compilador: o programador deve "lembrar" de verificar se houve erro
  - A lógica fica estruturada em condicionais aninhadas
- Solução 3 (boa): tratamento de exceções

https://github.com/acenelio/exceptions1-java

Resumo da aula

- Cláusula throws: propaga a exceção ao invés de trata-la
- Cláusula throw: lança a exceção / "corta" o método
- Exception: compilador obriga a tratar ou propagar
- RuntimeException: compilador não obriga
- O modelo de tratamento de exceções permite que erros sejam tratados de forma consistente e flexível, usando boas práticas
- Vantagens:
  - Lógica delegada
  - Construtores podem ter tratamento de exceções
  - Possibilidade de auxílio do compilador (Exception)
  - Código mais simples. Não há aninhamento de condicionais: a qualquer momento que uma exceção for disparada, a execução é interrompida e cai no bloco catch correspondente.
  - É possível capturar inclusive outras exceções de sistema