

```
package application;

import java.io.File;
import java.io.IOException;
import java.util.Scanner;

public class Program {

    public static void main(String[] args) {

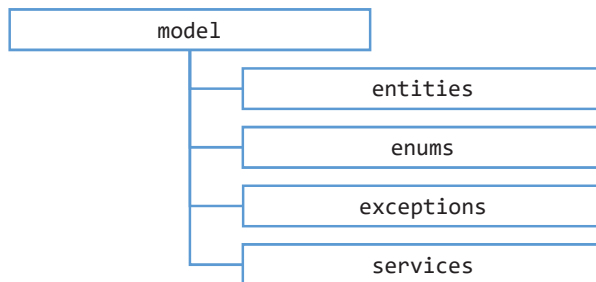
        File file = new File("C:\\temp\\in.txt");
        Scanner sc = null;
        try {
            sc = new Scanner(file);
            while (sc.hasNextLine()) {
                System.out.println(sc.nextLine());
            }
        } catch (IOException e) {
            System.out.println("Error opening file: " + e.getMessage());
        } finally {
            if (sc != null) {
                sc.close();
            }
        }
    }
}
```

Criando exceções personalizadas

<http://educandoweb.com.br>

Prof. Dr. Nelio Alves

Sugestão de pacotes "model"



Problema exemplo

Fazer um programa para ler os dados de uma reserva de hotel (número do quarto, data de entrada e data de saída) e mostrar os dados da reserva, inclusive sua duração em dias. Em seguida, ler novas datas de entrada e saída, atualizar a reserva, e mostrar novamente a reserva com os dados atualizados. O programa não deve aceitar dados inválidos para a reserva, conforme as seguintes regras:

- Alterações de reserva só podem ocorrer para datas futuras
- A data de saída deve ser maior que a data de entrada

Reservation
- roomNumber : Integer - checkin : Date - checkout : Date
+ duration() : Integer + updateDates(checkin : Date, checkout : Date) : void

Examples

Room number: **8021**
 Check-in date (dd/MM/yyyy): **23/09/2019**
 Check-out date (dd/MM/yyyy): **26/09/2019**
 Reservation: Room 8021, check-in: 23/09/2019, check-out: 26/09/2019, 3 nights

Enter data to update the reservation:
 Check-in date (dd/MM/yyyy): **24/09/2019**
 Check-out date (dd/MM/yyyy): **29/09/2019**
 Reservation: Room 8021, check-in: 24/09/2019, check-out: 29/09/2019, 5 nights

Room number: **8021**
 Check-in date (dd/MM/yyyy): **23/09/2019**
 Check-out date (dd/MM/yyyy): **21/09/2019**
 Error in reservation: Check-out date must be after check-in date

Examples

Room number: **8021**
 Check-in date (dd/MM/yyyy): **23/09/2019**
 Check-out date (dd/MM/yyyy): **26/09/2019**
 Reservation: Room 8021, check-in: 23/09/2019, check-out: 26/09/2019, 3 nights

Enter data to update the reservation:
 Check-in date (dd/MM/yyyy): **24/09/2015**
 Check-out date (dd/MM/yyyy): **29/09/2015**
 Error in reservation: Reservation dates for update must be future dates

Room number: **8021**
 Check-in date (dd/MM/yyyy): **23/09/2019**
 Check-out date (dd/MM/yyyy): **26/09/2019**
 Reservation: Room 8021, check-in: 23/09/2019, check-out: 26/09/2019, 3 nights

Enter data to update the reservation:
 Check-in date (dd/MM/yyyy): **24/09/2020**
 Check-out date (dd/MM/yyyy): **22/09/2020**
 Error in reservation: Check-out date must be after check-in date

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