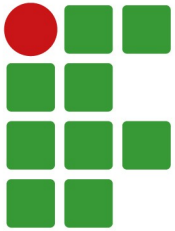
Programação Orientada a Objetos - POOS3

Tecnologia em Análise e Desenvolvimento de Sistemas

Laboratório 2
Implementação de projetos que manipulam datas

2º semestre de 2018



INSTITUTO FEDERAL
São Paulo
Câmpus Araraquara

Problema 1

- A sequência dos 14 algarismos finais de um boleto é uma das mais importantes quando você bate o olho em uma linha digitável: é ela que informa a data de vencimento daquele documento e também o valor (sem descontos) que deve ser pago.
- Os quatro primeiros números informam o fator de vencimento e são, literalmente, o número de dias decorridos desde a data-base (07/10/1997) até a data de pagamento. Por exemplo, se o seu boleto vence em 22/04/2013, o código indicador deve ser 5676. Caso este número seja 0000, o documento não possui uma data de vencimento.
- Já os 10 últimos indicam o valor do documento. Por exemplo, para o pagamento de um boleto de R$ 150,75, o final do boleto será 0000015075. Este valor é mostrado sem descontos e é por isso que, no caixa automático, é pedido para que você digite separadamente o desconto, caso exista.
- Vamos elaborar um sistema que leia os 14 últimos dígitos de um boleto e informe o valor e a data de vencimento.

Projeto

- Crie no Eclipse um novo projeto denominado:
  - “Laboratorio2_Boleto”
- Crie os pacotes:
  - model
  - view

Boleto

- No pacote model, crie a classe Boleto.java
  - Como se manipulará datas, importe os seguintes pacotes:
    - java.time.LocalDate;
    - java.time.Month;
    - java.time.format.DateTimeFormatter;
  - Os atributos da classe são:
    - private long numeros;
      - Armazenará o número (14 digitos) do boleto
    - private LocalDate dataVencimento;
      - Possui a data de vencimento do boleto
    - private double valor;
      - Possui o valor do boleto

Boleto

- Defina o método construtor e os gets conforme descrito a seguir:

```
    public Boleto(long numeros) {
        this.numeros = numeros;
        extrairValor(numeros);
        extrairVencimento(numeros);
    }

    public LocalDate getDataVencimento(){
        return dataVencimento;
    }

    public double getValor() {
        return valor;
    }
```

Boleto

- O método extrair valor deve recuperar os 10 últimos números do boleto e calcular o valor (double) do boleto.
- Para isso, faz uma divisão (resto) pela quantidade de dígitos desejados e depois a conversão para ponto flutuante (divisão por 100).
- Implemente o método conforme apresentado:

```
    private void extrairValor(long nro){
        long value;
        value = (long) (nro % Math.pow(10, 10));
        valor = value / 100.0;
    }
```

Boleto

- O método extrair vencimento deve recuperar a data de vencimento do boleto, contudo, essa informação não está disponível na linha digitável do boleto, será necessário recuperar essa data.
- Inicialmente deve-se extrair os 4 primeiros dígitos do boleto, esses dígitos representam o número de dias, após a data base, para o vencimento do boleto.
- Com a quantidade de dias, instancia-se um objeto com a data base e acrescenta-se o número de dias, com isso, obtém-se a data de vencimento do boleto.
- Implemente o método como abaixo:

```
    private void extrairVencimento(long nro){
        int dias;
        dias = (int) (nro / Math.pow(10,10));
        LocalDate dataBase = LocalDate.of(1997, Month.OCTOBER, 07);
        this.dataVencimento = dataBase.plusDays(dias);
    }
```

Boleto

- Nem sempre é desejado apresentar um objeto LocalDate como retorno para a data de vencimento, desta forma, iremos implementar um método getDataVencimento que retorna a data de vencimento no formato de String como utilizado no Brasil.
- Implemente o método abaixo:

```
    public String getVencimento(){
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("dd/MM/yyyy");
        return dataVencimento.format(formatter);
    }
```

Testando

- Implemente o método main() abaixo para testar o sistema:

```
      public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        long sequencia;
        Boleto boleto;

        System.out.println("Informe a última sequencia numérica do boleto: ");
        sequencia = input.nextLong();

        boleto = new Boleto(sequencia);

        System.out.println("Data de Vencimento: " + boleto.getVencimento());
        System.out.println("Valor: R$ " + boleto.getValor());


      }
```

LocalDate

- As classes do pacote java.time permitem que interpretações do tempo sejam definidas e manipuladas de forma precisa, ao contrário do que acontecia ao usarmos Date ou Calendar.
- Temos, por exemplo, a classe LocalDate que representa uma data, ou seja, um período de 24 horas com dia, mês e ano definidos.
- Um LocalDate serve para representarmos, por exemplo, a data de vencimento de um boleto, em que não nos importa as horas ou minutos, mas o dia todo.
- Podemos criar um LocalDate para uma data específica utilizando o método of.

```
LocalDate hoje = LocalDate.now();
System.out.println(hoje); //2018-11-23
```

```
LocalDate nascimento = LocalDate.of(1500, 4, 21);
```

LocalDate

- Para calcularmos a duração entre dois LocalDate, devemos utilizar um Period, que já trata anos bissextos e outros detalhes.

```
LocalDate homemNoEspaco = LocalDate.of(1961, Month.APRIL, 12);
LocalDate homemNaLua = LocalDate.of(1969, Month.MAY, 25);

Period periodo = Period.between(homemNoEspaco, homemNaLua);

System.out.printf("%s anos, %s mês e %s dias",
  periodo.getYears() , periodo.getMonths(), periodo.getDays());
  //8 anos, 1 mês e 13 dias
```

LocalTime

- A classe LocalTime serve para representar apenas um horário, sem data específica.
- Pode-se, por exemplo, usá-la para representar o horário de entrada no trabalho.

```
LocalTime horarioDeEntrada = LocalTime.of(9, 0);
System.out.println(horarioDeEntrada); //09:00
```

LocalDateTime

- A classe LocalDateTime serve para representar uma data e hora específicas.
- Podemos representar uma data e hora de uma prova importante ou de uma audiência em um tribunal.

```
LocalDateTime agora = LocalDateTime.now();
LocalDateTime aberturaDaCopa = LocalDateTime.of(2014, Month.JUNE, 12, 17, 0);
System.out.println(aberturaDaCopa); //2014-06-12T17:00 (formato ISO-8601)
```

Problema 1 - Complemento

- Sabe-se que sempre que um boleto é pago após o vencimento há cobrança de multa e juros.
  - Altere o projeto para ler o valor da multa e a porcentagem de juros por dia de atraso.
  - Caso o boleto esteja vencido, apresente o valor do boleto, o valor da multa, o valor dos juros e o valor a pagar de acordo com a data de hoje.

Problema 2

- Implementar um cronômetro regressivo.
  - O cronometro lê a quantidade de minutos desejados e a cada minuto passado é apresentada uma mensagem dizendo o tempo passado e quanto tempo ainda está disponível.

Projeto

- Crie no Eclipse um novo projeto denominado:
  - “Laboratorio2_Cronometro”
- Crie o pacote:
  - view

Cronometro

- Crie a classe Cronometro.java
  - A classe possui apenas o método main().
  - Importe as classes:
    - java.time.Duration;
    - java.time.Instant;
    - java.util.Scanner;

main()

```
public static void main(String[] args) {
  Instant inicio, fim;
  Duration duracao;
  int minutos;
  Scanner input = new Scanner(System.in);
  System.out.println("Quantidade de minutos: ");
  minutos = input.nextInt();
  inicio = Instant.now();
  System.out.println("Inicio: " + inicio.toString());
  do {
      try {
          Thread.sleep(100000);
      } catch (InterruptedException ex) {
          ex.printStackTrace();
      }
      fim = Instant.now();
      duracao = Duration.between(inicio, fim);
      System.out.println("Passaram: " + duracao.toMinutes() + " minutos");
      System.out.println("Você tem: " + (minutos-duracao.toMinutes()) + " minutos para terminar");
  }while (duracao.toMinutes() < minutos);

  System.out.println("Acabou o tempo!!!!");
}
```

Teste

- Execute o programa implementado e entre com 20 minutos.
- Após solicite ao professor que explique o funcionamento do sistema.

Datas para computadores

- Para um computador, o tempo é um número que cresce a cada instante.
- No Java, historicamente era utilizado um long que representava os milissegundos desde 01/01/1970 às 00:00:00.
- Na nova API, a classe Instant é utilizada para representar esse número, agora com precisão de nanossegundos.

```
Instant agora = Instant.now();
System.out.println(agora); //2014-04-08T10:02:52.036Z
```

- Podemos usar um Instant, por exemplo, para medir o tempo de execução de um algoritmo.

```
Instant inicio = Instant.now();
rodaAlgoritmo();
Instant fim = Instant.now();

Duration duracao = Duration.between(inicio, fim);
long duracaoEmMilissegundos = duracao.toMillis();
```

Problema 3

- Como sabe, o horário no mundo depende do fuso horário.
- Implemente um sistema que apresente que horas são agora em:
  - São Paulo (UTC-03:00)
  - Londres (UTC+01:00)
  - Moscou (UTC+03:00)
  - Tóquio (UTC+09:00)
  - Alasca (UTC-08:00)

Projeto

- Crie no Eclipse um novo projeto denominado:
  - “Laboratorio2_FusoHorario”
- Crie o pacote:
  - view

FusoHorario

- Crie a classe FusoHorario.java
- A classe possui apenas o método main().
- Importe as classes:
  - java.time.ZoneId;
  - java.time.ZonedDateTime;

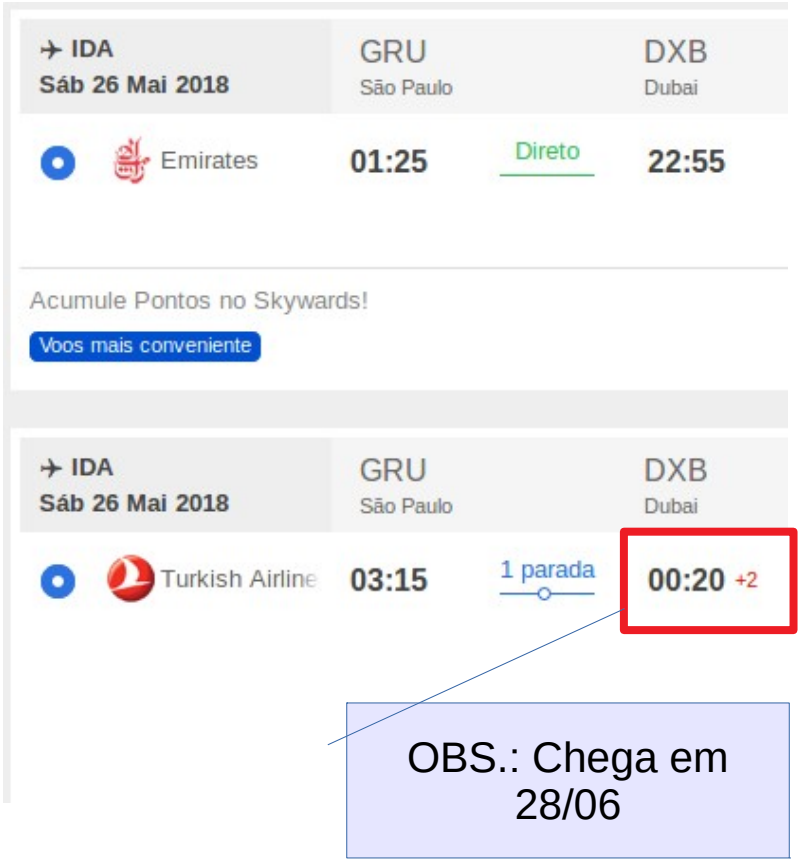
main()

```
public static void main(String[] args) {
     ZoneId saoPaulo = ZoneId.of("America/Sao_Paulo");
     ZoneId londres = ZoneId.of("Europe/London");
     ZoneId moscou = ZoneId.of("Europe/Moscow");
     ZoneId toquio = ZoneId.of("Asia/Tokyo");
     ZoneId alaska = ZoneId.of("US/Alaska");

     System.out.println("São Paulo: " + ZonedDateTime.now(saoPaulo));
     System.out.println("Londres: " + ZonedDateTime.now(londres));
     System.out.println("Moscou: " + ZonedDateTime.now(moscou));
     System.out.println("Tóquio: " + ZonedDateTime.now(toquio));
     System.out.println("Alaska: " + ZonedDateTime.now(alaska));
}
```

Problema 4

- Voos internacionais em geral são longos.
- Observe na imagem abaixo que a viagem para do Brasil para Dubai pode ser longa, mais que 24 horas.
- Implementar um sistema que apresente o tempo de viagem.



Projeto

- Crie no Eclipse um novo projeto denominado:
  - “Laboratorio2_TempoVoo”
- Crie o pacote:
  - view

TempoVoo

- Crie a classe TempoVoo.java
- A classe possui apenas o método main().
- Importe as classes:
  - java.time.Duration;
  - java.time.LocalDateTime;
  - java.time.Month;
  - java.time.ZoneId;
  - java.time.ZonedDateTime;

main()

```
    public static void main(String[] args) {
        ZoneId saoPaulo = ZoneId.of("America/Sao_Paulo");
        ZoneId dubai = ZoneId.of("Asia/Dubai");

        LocalDateTime emSP = LocalDateTime.of(2018, Month.MAY, 26, 1, 25);
        LocalDateTime emDB = LocalDateTime.of(2018, Month.MAY, 26, 22, 55);

        ZonedDateTime saidaSP = ZonedDateTime.of(emSP, saoPaulo);
        ZonedDateTime chegadaDB = ZonedDateTime.of(emDB, dubai);

        Duration duracao = Duration.between(saidaSP, chegadaDB);
        System.out.println("Duração Emirates: " + duracao);

        emSP = LocalDateTime.of(2018, Month.MAY, 26, 3, 15);
        emDB = LocalDateTime.of(2018, Month.MAY, 28, 0, 20);

        saidaSP = ZonedDateTime.of(emSP, saoPaulo);
        chegadaDB = ZonedDateTime.of(emDB, dubai);

        duracao = Duration.between(saidaSP, chegadaDB);
        System.out.println("Duração Emirates: " + duracao);
    }
```

Problema 4 - Complemento

- Pesquise em algum site de venda de passagens uma passagem saindo da Austrália e chegando no Brasil. Implemente um caso de teste que faça o calculo da quantidade de horas dessa viagem.
- Permita que o usuário informe os dados de saída e chegada de uma viagem aérea.

Entrega

- Criar um repositório para cada projeto no github.com.
- O projeto deve ser disponibilizado no github, o último commit deve ser realizado até as 12h (meio dia) do dia 25/11.
- Atenção: não deve-se carregar o projeto compactado no github e sim cada um dos arquivos usando comandos do git.
- Os quatro links dos repositórios devem ser postados no moodle.