

Padrões de projeto, testes automatizados e XML

01. Definições de projeto, datas e constantes



Esp. Márcio Palheta

Gtalk: marcio.palheta@gmail.com

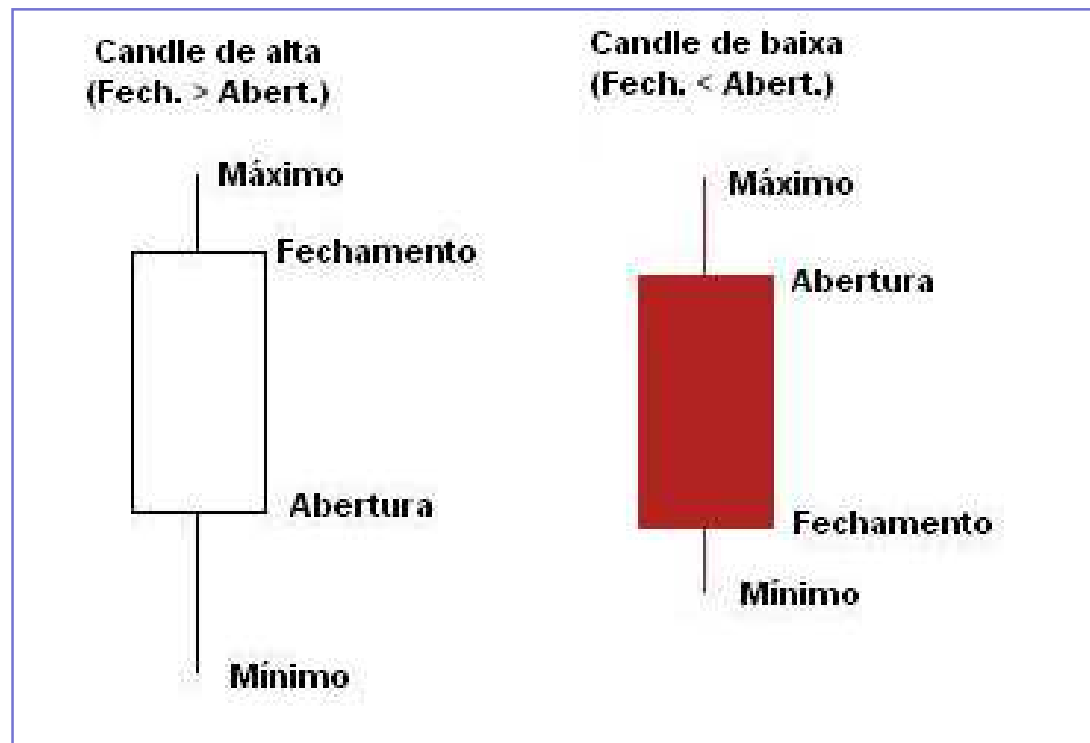


Estudo de caso

- A empresa **Big O**;
- Atua no ramo de bolsas de valores;
- Cada negócio realizado tem preço, quantidade e data
- Está interessada em acompanhar a movimentação diária;

Registros da Big O

- Resumo diário com preços de:
 - Abertura, fechamento, máximo e mínimo



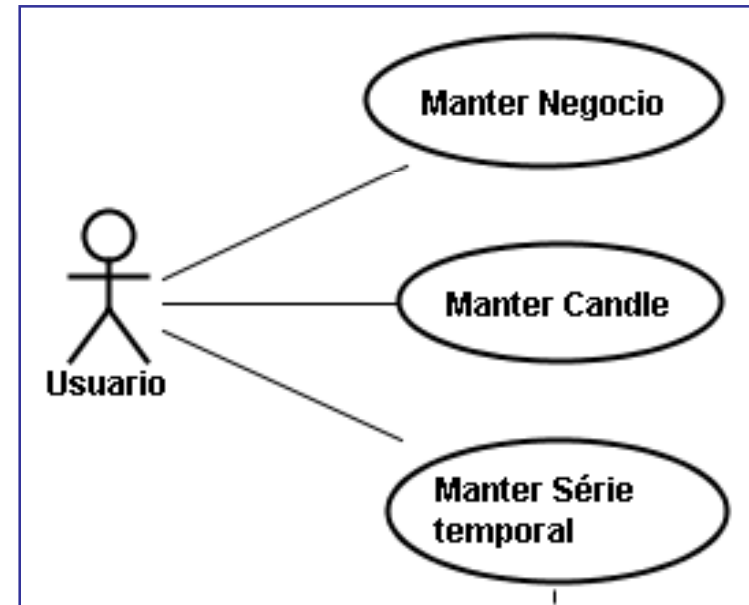
Big O e o Candlestick

- A empresa deseja emitir mensalmente um gráfico de acompanhamento
- Candlestick – Série temporal



Modelando o sistema

- Registro dos **negócios** realizados ao longo do dia;
- Gerar o resumo diário - **Candle**
- Manipular e agrupar conjuntos de Candles





O modificador **final**

- Utilidades da palavra reservada **final**:
- **Classe** – não pode ser estendida;
- **Método** – impede a sobrescrita;
- **Atributo** – o conteúdo de uma variável não pode ser alterado
 - Inicializar durante a construção;
 - Erro de compilação, caso não seja inicializada



Definindo uma constante de classe

- A palavra **final** é utilizada para definir uma constante;
- Em java, a definição de constantes é feita usando: **static final**
- Por padrão, o nome da constante fica em letra maiúscula;



Trabalhando com datas

- A classe abstrata `java.util.Calendar` encapsula um dado momento **milissegundos**;
- A classe concreta `GregorianCalendar` – calendário usado em muitos países;
- Recuperando data e hora atual:
 - `Calendar` agora = `Calendar.getInstance()`;



Usando um java.util.Calendar

```
Calendar data = Calendar.getInstance();
data.get(Calendar.DAY_OF_MONTH);
data.get(Calendar.DAY_OF_WEEK); // domingo == 1
System.out.println(data.get(Calendar.MONTH)); // jan == 0

data.set(Calendar.HOUR, 10); // data.hora = 10
data.set(Calendar.MINUTE, 30); // data.minuto = 30
data.set(2012, 11, 25); // data == Natal de 2012

data.add(Calendar.YEAR, 1); // adiciona 1 ao ano
data.add(Calendar.YEAR, -1); // diminui 1 ano

//Comparando datas
Calendar c1 = new GregorianCalendar(2012, Calendar.OCTOBER, 12);
Calendar c2 = new GregorianCalendar(2012, Calendar.DECEMBER, 25);
System.out.println(c1.after(c2));
```



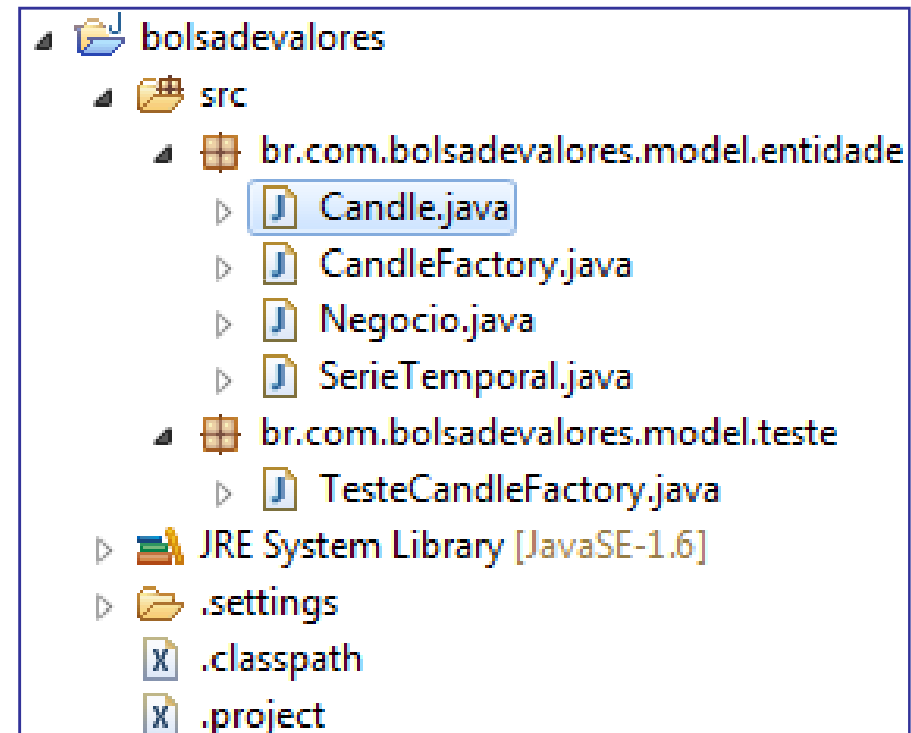
java.util.Date

- Não recomendada
 - Métodos marcados como **deprecated**;
- Substituída por **Calendar** em Java 1.1;
- Conversão entre objetos de data:

```
//Definicao da variavel Calendar
Calendar c = new GregorianCalendar(2005, Calendar.OCTOBER, 12);
//Transformacao de Calendar em Date
Date d = c.getTime();
//Transformacao de Date em Calendar
c.setTime(d);
```

Exercício 01

- Criação de um novo projeto java: **bolsadevalores**;
- Criação das classes entidade;
- Criação das classes de Teste;
- **OBS:** siga a definição de pacotes





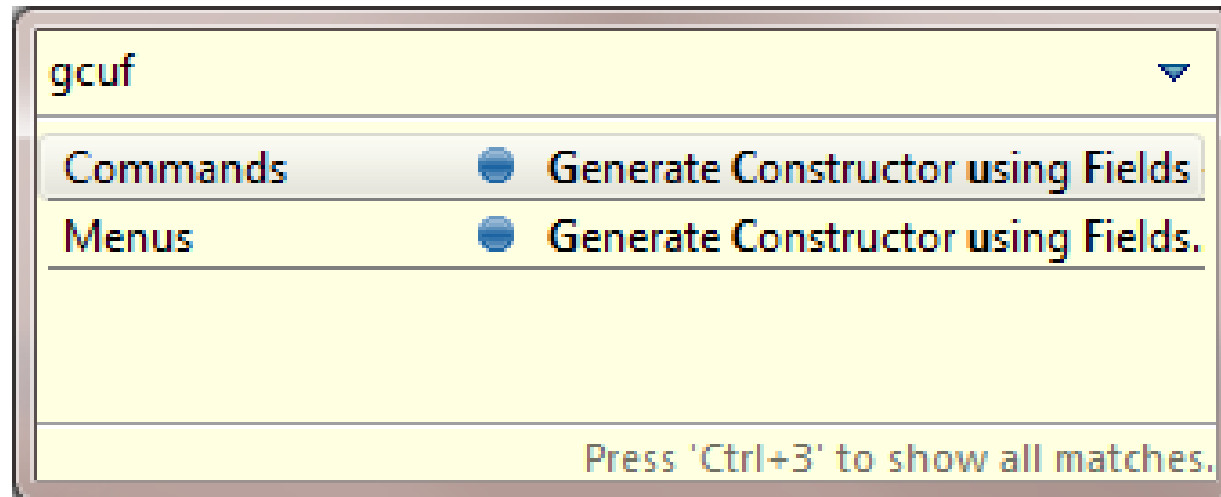
Classe Negocio.java

- Definição de atributos e métodos

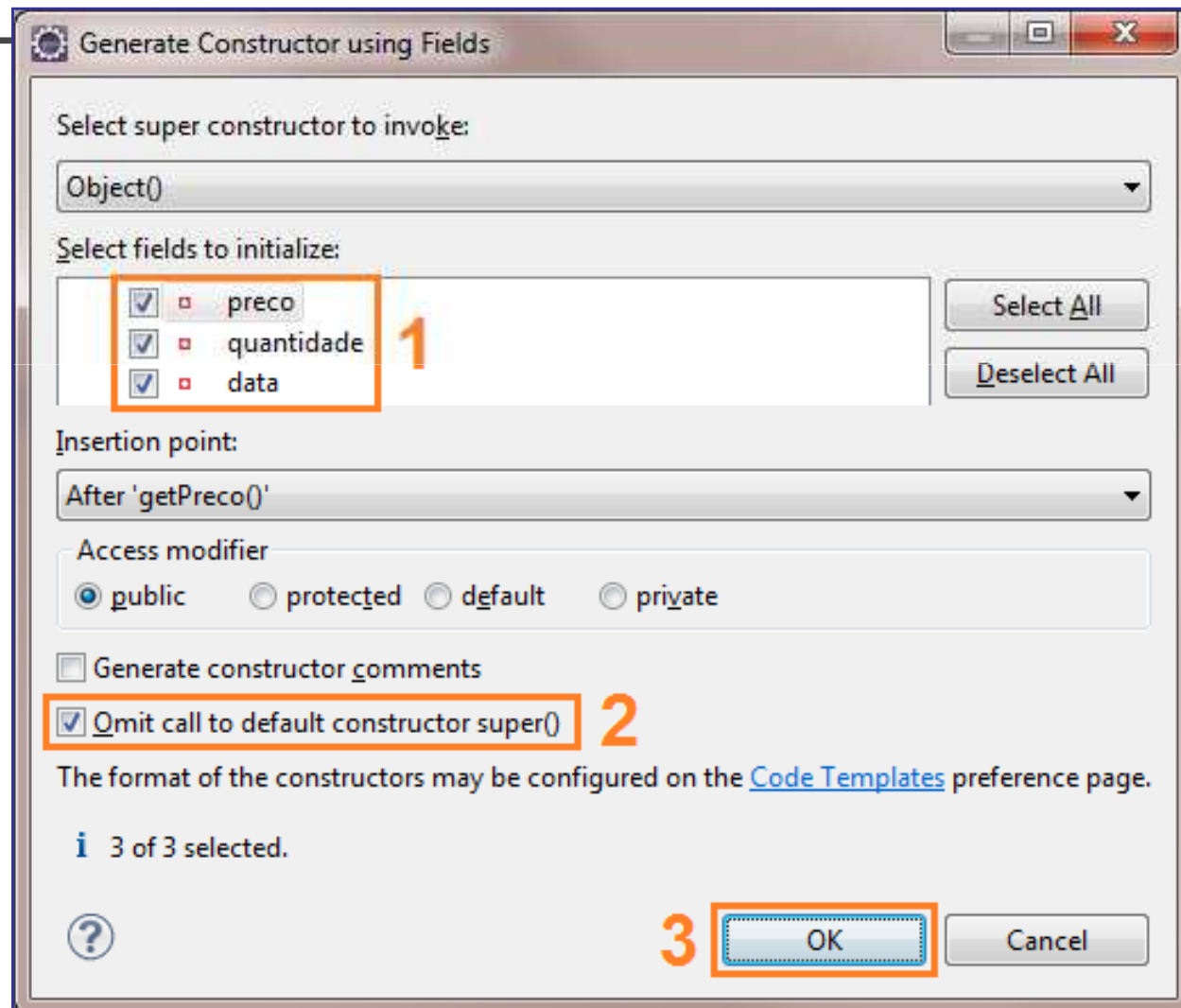
```
package br.com.bolsadevalores.model.entidade;  
  
import java.util.Calendar;  
  
public class Negocio {  
  
    //Atributos  
    private double preco;  
    private int quantidade;  
    private Calendar data;  
  
    //Metodos de get e set
```

Definição do método de inicialização

- No menu, selecione:
 - Source/Generate Constructors using Fields
- Ou: Tecle **Ctrl+3** e digite **gcuf**



Selecionando os atributos





Classe Negocio atualizada

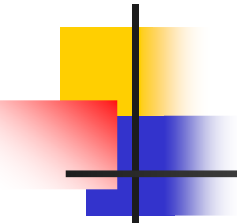
```
public class Negocio {  
  
    // Atributos  
    private double preco;  
    private int quantidade;  
    private Calendar data;  
  
    // Metodo construtor gerado  
    public Negocio(double preco,  
                    int quantidade,  
                    Calendar data) {  
        this.preco = preco;  
        this.quantidade = quantidade;  
        this.data = data;  
    }  
}
```



Novo método getVolume()

```
public class Negocio {  
    // Atributos  
    private double preco;  
    private int quantidade;  
    private Calendar data;  
    // Metodo construtor gerado  
    public Negocio(double preco,  
                    int quantidade,  
                    Calendar data) {  
        this.preco = preco;  
        this.quantidade = quantidade;  
        this.data = data;  
    }  
    //Metodo que retorna o volume  
    public double getVolume(){  
        return preco * quantidade;  
    }  
}
```


Uma classe imutável



```
package br.com.bolsadevalores.model.entidade;
import java.util.Calendar;
public final class Negocio {
    private final double preco;
    private final int quantidade;
    private final Calendar data;
    public Negocio(double preco, int quantidade, Calendar data) {
        super();
        this.preco = preco;
        this.quantidade = quantidade;
        this.data = data;
    }
    public double getVolume(){
        return quantidade * preco;
    }
    public double getPreco() {
        return preco;
    }
    public int getQuantidade() {
        return quantidade;
    }
    public Calendar getData() {
        return data;
    }
}
```



Definição da classe Candle

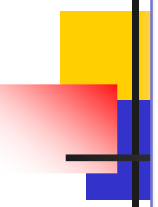
```
public final class Candle {  
  
    private final double abertura;  
    private final double fechamento;  
    private final double minimo;  
    private final double maximo;  
    private final double volume;  
    private final Calendar date;  
  
    public Candle(double abertura, double fechamento, double minimo,  
                  double maximo, double volume, Calendar date) {  
        super();  
        this.abertura = abertura;  
        this.fechamento = fechamento;  
        this.minimo = minimo;  
        this.maximo = maximo;  
        this.volume = volume;  
        this.date = date;  
    }  
    //Metodos de GET()  
}
```

Definição da CandleFactory

```
package br.com.bolsadevalores.model.entidade;
import java.util.Calendar;
import java.util.List;

public class CandleFactory {
    public Candle constroiCandlePorData(Calendar data, List<Negocio> negocios)
        //Definicao de variaveis
        double maximo = negocios.get(0).getPreco();
        double minimo = negocios.get(0).getPreco();
        double volume = 0;
        double abertura = negocios.get(0).getPreco();
        double fechamento = negocios.get(negocios.size() - 1).getPreco();
        //Digite: foreach
        for (Negocio negocio : negocios) {
            volume += negocio.getVolume();
            if (negocio.getPreco() > maximo) {
                maximo = negocio.getPreco();
            } else if (negocio.getPreco() < minimo) {
                minimo = negocio.getPreco();
            }
        }
        return new Candle(abertura, fechamento, minimo, maximo, volume, data);
    }
}
```

Teste da fábrica de Candles



```
public class TesteCandleFactory {  
    public static void main(String[] args) {  
        Calendar hoje = Calendar.getInstance();  
        //Definicao de objetos de negocio  
        Negocio negocio1 = new Negocio(40.5, 100, hoje);  
        Negocio negocio2 = new Negocio(45.0, 100, hoje);  
        Negocio negocio3 = new Negocio(39.8, 100, hoje);  
        Negocio negocio4 = new Negocio(42.3, 100, hoje);  
  
        //Montagem da lista de negocios  
        List<Negocio> lista = Arrays.asList(negocio1, negocio2,  
            negocio3, negocio4);  
  
        //Definicao da fabrica de Candles  
        CandleFactory fabrica = new CandleFactory();  
        //Criacao do objeto Candle  
        Candle candle = fabrica.constroiCandlePorData(hoje, lista);  
        System.out.println(candle.getAbertura());  
        System.out.println(candle.getFechamento());  
        System.out.println(candle.getMinimo());  
        System.out.println(candle.getMaximo());  
        System.out.println(candle.getVolume());  
    }  
}
```



java.util.Arrays

- Classe utilitária para arrays;
- `asList()` – cria uma lista a partir de um array;
- Mas não passamos um array. O que houve?
- O método `asList()` aceita **varargs**
- Parece autoboxing de arrays



Bibliografia

- Java - Como programar, de Harvey M. Deitel
- Use a cabeça! - Java, de Bert Bates e Kathy Sierra
- (Avançado) Effective Java Programming Language Guide, de Josh Bloch



Referências WEB

- **Site oficial:**

- SUN: www.java.sun.com

- **Fóruns e listas:**

- Javaranch: www.javaranch.com
- GUJ: www.guj.com.br

- **Apostilas:**

- Argonavis: www.argonavis.com.br
- Caelum: www.caelum.com.br

Padrões de projeto, testes automatizados e XML

01. Definições de projeto, datas e constantes



Esp. Márcio Palheta

Gtalk: marcio.palheta@gmail.com