Universidade do Estado de Santa Catarina – UDESC
Curso de Bacharelado em Sistemas de Informação
Disciplina: Programação II
Professor: **Antônio Carlos Tamanini da Silva**
Aluno: _____

Simulado2:

Construa a interface da Figura 2 em PythonQt, onde cada coluna forma um vetor, com exceção do cabeçalho que é formado por objetos simples do tipo QLabel. Cada vetor possui cinco elementos. As colunas são compostas pelos vetores LEd_Quant e LEd_Preco que possuem valores digitados pelo usuário, e a coluna LEd_Total que forma o total de cada item comprado.

A coluna LEd_Total forma o total de cada item de compra após o usuário clicar em **Calculo**. Por fim o objeto LEd_TotalFinal deve conter o valor final a ser pago na nota.

O programa possui o seguinte diagrama de classe:

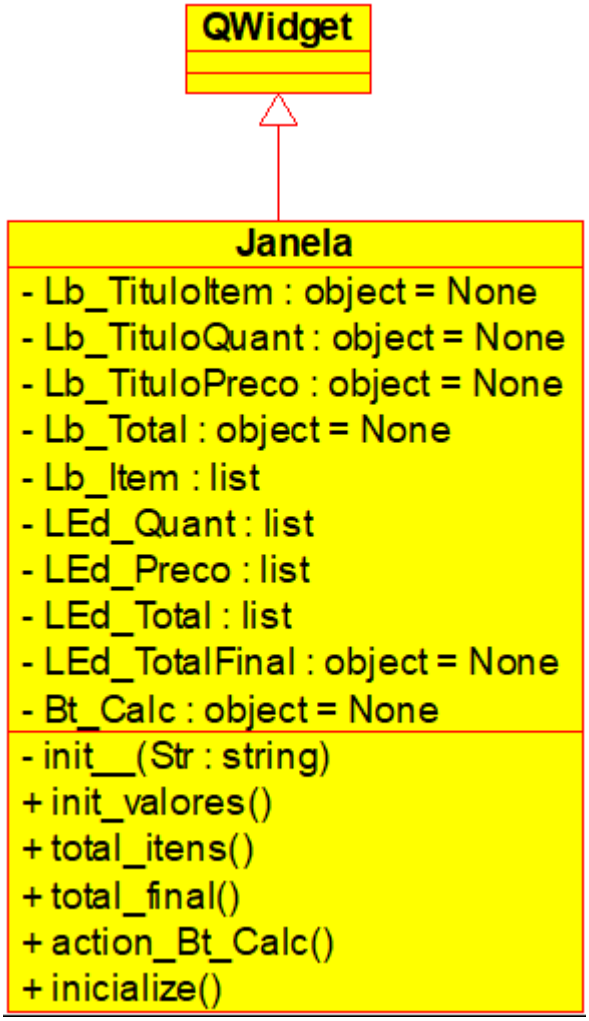

Figura 1: Diagrama de classe
Fonte: Umbrello 2.27.2

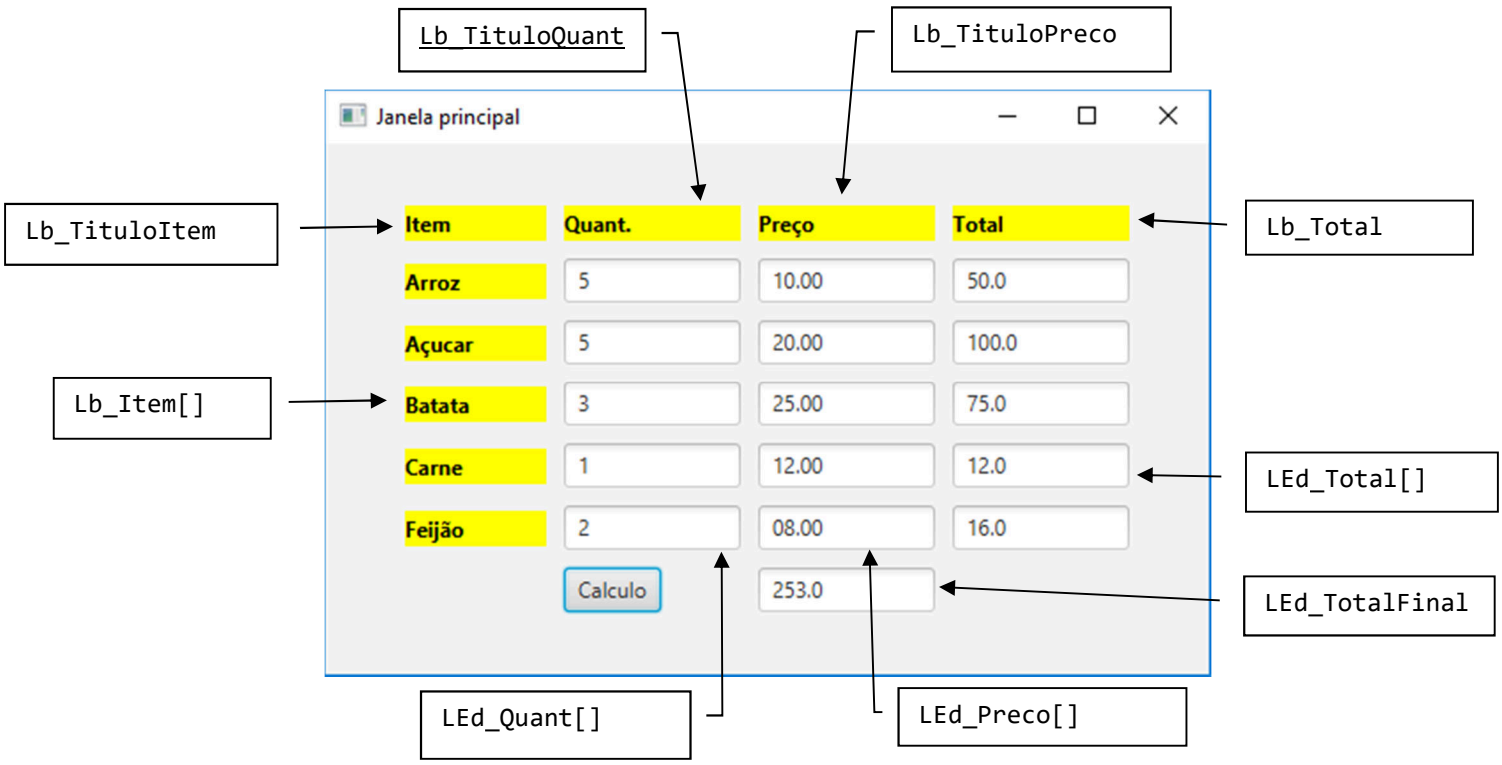

Figura 2: Interface do programa

Neste simulado, parte do programa que constrói a interface da Figura 2 é fornecida e utiliza a biblioteca PythonQt, cabe ao aluno completar as partes faltantes do código.

01 - Complete o código que define a classe Janela: **class Janela** (Vale 0.5)
02 - Criar o programa principal que aloca um objeto da classe Janela e inicia a execução do programa. (Vale 1.0)
03 - Inicializar os valores do vetor Lb_Item[] com o comando inserção de texto de um QLabel. (Valor 1.0)
04 - Criar o evento que calcula o total de cada item descrito na nota fiscal e atribuir na caixa LEd_Total[]. **É obrigatório o tratamento de exceções.** (Vale 2.0)
05 - Criar o evento que calcula o total de **todos** os itens contidos na nota fiscal e atribuir na caixa LEd_TotalFinal. **É obrigatório o tratamento de exceções.** (Vale 2.0)
06 - Chamar os eventos que fazem os cálculos citados em 04, 05 (Vale 0.5)

```
import sys
from PyQt5.QtWidgets import *
from PyQt5.QtCore import *
from PyQt5.QtGui import *

class Janela("Questão 01"):
    __Lb_TituloItem=None
    __Lb_TituloQuant=None
    __Lb_TituloPreco=None
    __Lb_Total=None

    __Lb_Item=[]
    __LEd_Quant=[]
    __LEd_Preco=[]
    __LEd_Total=[]

    __LEd_TotalFinal=None
    __Bt_Calc=None

    def __init__(self, Str="Janela", x1=0, y1=0, dx=640, dy=480, cor="lightgray"):
        super().__init__()  ## Obrigatório
        self.setWindowTitle(Str)
        self.setGeometry(x1, y1, dx, dy)

        # Set window background color
        self.setAutoFillBackground(True)
        p = self.palette()
        p.setColor(self.backgroundRole(), QColor(cor))
        self.setPalette(p)

        self.inicialize()

    def init_valores(self):
        Questão 03: Inicializar os valores do vetor Lb_Item[]

    def total_itens(self):
        Questão 04: Criar o evento que calcula o total de cada item descrito na nota fiscal e atribuir
        na caixa LEd_Total[]. É obrigatório o tratamento de exceções.

    def total_final(self):
        Questão 05: Criar o evento que calcula o total de todos os itens contidos na nota fiscal e
        atribuir na caixa LEd_TotalFinal. É obrigatório o tratamento de exceções.


    def action_Bt_Calc(self):
        Questão 06: Chamar os eventos que fazem os cálculos citados em 04, 05.

    def inicialize(self):
        Grid=QGridLayout()

        Questão 07: Alocar os componentes (Questão Objetiva)

        self.init_valores()

        Questão 08: Acrescentar no layout os componentes (Questão Objetiva)

        self.setLayout(Grid)
        self.show()

#############################################################################################

        Questão 02: Criar o programa principal
```

Questão 07: Assinale uma ou mais alternativas corretas para alocar os vetores: Lb_Item[], LEd_Quant[], LEd_Preco[], LEd_Total[]: (método init) (valor 1.0)

a) ☐
```
nl=5
for i in range(0, nl, 2):
        Lb = QLabel(self, text="")
        self.__Lb_Item.append(Lb)
        LEd=QLineEdit(self, width=52)
        self.__LEd_Quant.append(LEd)
        LEd = QLineEdit(self, width=52)
        self.__LEd_Quant.append(LEd)
        LEd = QLineEdit(self, width=52)
        self.__LEd_Quant.append(LEd)
```

b) ☐
```
nl=12
for i in range(0, nl, 1):
        Lb = QLabel(self, text="")
        self.__Lb_Item.append(Lb)
        LEd= QLabel(self, width=52)
        self.__LEd_Quant.append(LEd)
        LEd = QLabel(self, width=52)
        self.__LEd_Preco.append(LEd)
        LEd = QLabel(self, width=52)
        self.__LEd_Total.append(LEd)
```

c) ☐
```
nl=5
for i in range(0, nl, 1):
        Lb = QLabel(self, text="")
        self.__Lb_Item.append(Lb)
        LEd=QLineEdit(self, width=52)
        self.__LEd_Quant.append(LEd)
        LEd = QLineEdit(self, width=52)
        self.__LEd_Preco.append(LEd)
        LEd = QLineEdit(self, width=52)
        self.__LEd_Total.append(LEd)
```

d) ☐
```
nl=6
for i in range(0, nl, 2):
        Lb = QLabel(self, text="")
        self.__Lb_Item.append(Lb)
        self.__LEd_Quant.append(LEd)
        self.__LEd_Preco.append(LEd)
        self.__LEd_Total.append(LEd)
```

e) ☐
```
nl=5
for i in range(0, nl, 1):
        LEd=QLineEdit(self, width=52)
        self.__LEd_Quant.append(LEd)
        LEd = QLineEdit(self, width=52)
        self.__LEd_Preco.append(LEd)
        LEd = QLineEdit(self, width=52)
        self.__LEd_Total.append(LEd)
        Lb = QLabel(self, text="")
        self.__Lb_Item.append(Lb)
```

Questão 08: Assinale uma ou mais alternativas corretas para acrescentar no layout os componentes: Lb_Item[], LEd_Quant[], LEd_Preco[], LEd_Total[]: (método start) (valor 1.0)

a)
```
nl=self.__Lb_Item.__len__()
for i in range(0, nl, 1):
        Grid.addWidget(self.__Lb_Item[i], i + 1, 0)
        Grid.addWidget(self.__LEd_Quant[i], i + 1, 1)
        Grid.addWidget(self.__LEd_Preco[i], i + 1, 2)
        Grid.addWidget(self.__LEd_Total[i], i + 1, 3)
```

b)
```
nl=6
for i in range(0, nl, 2):
        Grid.addWidget(self.__Lb_Item[i], i + 1, 0)
        Grid.addWidget(self.__LEd_Quant[i], i + 1, 1)
        Grid.addWidget(self.__LEd_Quant[i], i + 1, 2)
        Grid.addWidget(self.__LEd_Quant[i], i + 1, 3)
```

c)
```
nl=5
for i in range(0, nl, 1):
        Grid.addWidget(__Lb_Item[i], i + 1, 0)
        Grid.addWidget(__LEd_Quant[i], i + 1, 1)
        Grid.addWidget(__LEd_Preco[i], i + 1, 2)
        Grid.addWidget(__LEd_Total[i], i + 1, 3)
```

d)
```
nl=5
for i in range(0, nl, 1):
        Grid.addWidget(self.__Lb_Item[i], i + 1, 0)
        Grid.addWidget(self.__LEd_Quant[i], i + 1, 1)
        Grid.addWidget(self.__LEd_Preco[i], i + 1, 2)
        Grid.addWidget(self.__LEd_Total[i], i + 1, 3)
```

e)
```
nl=5
for i in range(0, nl, 2):
        Grid.addWidget(self.Lb_Item[i], i + 1, 0)
        Grid.addWidget(self.LEd_Quant[i], i + 1, 1)
        Grid.addWidget(self.LEd_Preco[i], i + 1, 2)
        Grid.addWidget(self.LEd_Total[i], i + 1, 3)
```

Questão 09: Verifique se a afirmação abaixo está correta: (valor 1.0)

O comando que serve para abrir uma janela de diálogo para apresentação de erros é:
`QMessageBox.about(self, "Título da Janela", "Mensagem de erro")`
Onde o primeiro parâmetro é a janela pai, o segundo o título da janela e o terceiro a mensagem de erro
Escolha uma opção:

- Verdadeiro
- Falso

Penalidades: Caso não usar tratamento de exceções haverá penalidade de 1 ponto.

A tarefa deverá ser enviada pelo Moodle. Cada dia de atraso no envio sofrerá um desconto na nota de 10%. (Valor máximo da tarefa 2 pontos)

Boa Sorte!!!