Programação com Interfaces Gráfica

Mario Benevides e Paulo Roma

Universidade Federal do Rio de Janeiro
Rio de Janeiro, Brasil

Projeto 3 - Impressora

Agenda

Aulas Passadas:

- Introdução a OO e Classes
- Classes
- Exceções
- Módulos
- Arquivos

Nesta Aula: Projeto envolvendo estes conceitos

Projeto 3: Controle de Tina e Papel numa Impressora

Escreva um programa para implementar uma classe $impressora$ para controlar a quantidade de tinta e papel na impresora.

- Papel pode ser adicionado à impressora a qualquer momento, e assume-se que não há nenhuma capacidade máxima para papel.
- Uma impressora recém construída tem um cartucho de tinta completo contendo a quantidade de tinta dado pela constante `INK_CAPACITY`.
- A impressora pode imprimir em um lado ou em frente e verso.
- Para cada lado impresso, uma pequena quantidade da tinta é usada, como determinado pela constante `INK_USAGE`.
- O cartucho de tinta pode ser substituído em qualquer altura, restaurando a quantidade da tinta para o valor `INK_CAPACITY`.

Projeto 1: Objeto Printer - Métodos

Métodos

- __init__
- addPaper
- getCurrentPaper
- getTotPaperUsed
- isInkOut
- replaceInk
- printOneSided
- printTwoSided
- __str__

Classe Printer

```
## Models the usage of paper and ink in a printer.
class printer:
##
  # Capacity, in ounces, of a new ink cartridge.
  #
INK_CAPACITY = 2.0 # 56.699 gramas

##
  # Amount of ink, in ounces, used per printed page.
  #
INK_USAGE = 0.0023
```

O Método *__init__*

```
## Printer initially contains a given number of paper sheets
#  and a full ink cartridge.
#
# @param givenNumberOfSheets initial number of paper sheets.

def __init__(self,givenNumberOfSheets):
## number of sheets available
self.__numberOfSheets = abs(givenNumberOfSheets)

## total paper used since construction
self.__totalPaperUsed = 0

## quantity of ink available
self.__inkQuantity = printer.INK_CAPACITY
```

Método *addPaper*

```
## Adds the given number of sheets of paper to this printer.
#  We assume that there is no maximum capacity.
#
#  @param additionalSheets number of sheets to be added
#  to the printer.

def addPaper(self,additionalSheets):
          if additionalSheets > 0:
               self.__numberOfSheets += additionalSheets
```

Os Métodos *getCurrentPaper* e *getTotPaperUsed*

```
## Returns the number of sheets of paper currently in this pri
#
#  @return number of sheets available.

def getCurrentPaper(self):
            return self.__numberOfSheets

## Returns the total number of sheets of paper printed by this
#  Sheets used for two sided printing still count as just one
#
#  @return number of sheets of paper used since construction.

def getTotPaperUsed(self):
            return self.__totalPaperUsed
```

Os Métodos *isInkOut* e *replaceInk*

```
## Check if the ink has run out. Returns true if the amount
#  of ink left is smaller than the quantity INK_USAGE.
#
#  @return True if the ink is over, and False otherwise.

   def isInkOut(self):
      return self.__inkQuantity < printer.INK_USAGE

## Simulates replacement of the ink cartridge, restoring the
#  quantity of ink in the printer to INK_CAPACITY.

   def replaceInk(self):
      self.__inkQuantity = printer.INK_CAPACITY
```

O Método *printOneSided*

Imprimir um Lado da Folha

- Usando a quantidade apropriada de **folhas** e de **tinta**
- Se quantidade de papel é insuficiente ela imprimi até acabar o papel
  - usando a quantidade de tinta necessária;
- Se quantidade de tinta é insuficiente, ela usa a tinta até acabar
  - e imprimi folhas em branco até o final.

O Método *printOneSided*

```
## Simulates printing pages in one-sided mode, using the appropriate number of sheets and a
#  corresponding quantity of ink. If there is not enough paper, the printer will use up all
#  remaining paper and will only use the quantity of ink needed for the sheets actually
#  printed. If there is not enough ink, the printer will use up all the ink, and will still
#  use up the specified number of sheets of paper
#  (i.e., it just prints a bunch of blank pages after the ink runs out).
#

#  @param numberOfPages number of sheets of paper to be printe

def printOneSided(self,numberOfPages):
    np = max(numberOfPages,0)
    np = min(np, self.__numberOfSheets)

    self.__totalPaperUsed += np
    self.__inkQuantity -= printer.INK_USAGE*np
    self.__inkQuantity = max(self.__inkQuantity,0)
    self.__numberOfSheets -= np
```

O Método *printTwoSided*

Imprimir Dois Lado da Folha

- Similar ao *printOneSided*;
- Usando a quantidade apropriada de **folhas** e de **tinta**
- Se quantidade de papel é insuficiente ela imprimi até acabar o papel
  - usando a quantidade de tinta necessária;
- Se quantidade de tinta é insuficiente, ela usa a tinta até acabar
  - e imprimi folhas em branco até o final.
- Precisamos determinar quantos folhas serão necessárias:
  - exemplo: 4 páginas $\rightarrow$ 2 folhas e 5 páginas $\rightarrow$ 3 folhas.
- Precisamos determinar quantidade de tinta necessária:
  - Ou é o número de páginas requisitadas;
  - Ou (talvez duas vezes) o número de folhas disponíveis.
  - Escolhemos o menor.

O Método *printTwoSided*

```
## Simulates printing pages in two-sided mode, using the appropriate...  #
#  This is similar to printOneSided() method, but you first need to determine
#  how many sheets of paper are needed. For 1 or 2 pages, you need 1 sheet;
#  for 3 or 4 pages, you need 2 sheets; and so on. You can use integer division
#  (and/or the modulus operator) for this. Then, you have to figure out how many sheets
#  of paper will actually be used (as in printOneSided()). Finally, to calculate the ink
#  needed, you need to know how many pages will really be printed: this must be either
#  the original number os pages requested, or (maybe twice) the number of sheets of paper
#  available in the printer, whichever is smaller.
```

```
# @param numberOfPages num. sheets printed in double side.

def printTwoSided(self,numberOfPages):
    np = max(numberOfPages,0)
    nf = min(np//2+np%2, self.__numberOfSheets)
    # rnp is real number pages printed
    rnp = min(numberOfPages, 2*self.__numberOfSheets)
    self.__totalPaperUsed += nf
    self.__inkQuantity -= printer.INK_USAGE*rnp
    self.__inkQuantity = max(self.__inkQuantity,0)
    self.__numberOfSheets -= nf
```

O Método *__str__*

```
## Print printer statistics.
#
def __str__(self):
 return "Total paper = %d\nInk quantity = %f\nNum.Sheets = %d\n"% \
 (self.getTotPaperUsed(),self.__inkQuantity,self.getCurrentPaper())
```

Main - Parte 1

```
## Main program for testing.
def main():
   print ("Constructed(50)")
   prt = printer(50)
   print (prt)

   print ("printTwoSided(3)")
   prt.printTwoSided(3)
   print (prt)

   print ("printOneSided(2)")
   prt.printOneSided(2)
   print (prt)

   print ("printOneSided(60)")
   prt.printOneSided(60)
   print (prt)

   print ("addPaper(2000)")
   prt.addPaper(2000)
   print (prt)
```

Main - Parte 2

```
    print ("Sheets used = %d" % prt.getTotPaperUsed())
    print ("Out of ink = %s" % prt.isInkOut())
    print ("Sheets available = %d\n" % prt.getCurrentPaper())

    print ("printOneSided(870)")
    prt.printOneSided(870)
    print (prt)

    print ("Out of ink = %s\n" % prt.isInkOut())

    prt.replaceInk()
    print ("replaceInk()")
    print (prt)

    print ("printTwoSided(101)")
    prt101 = printer(50)
    prt101.printTwoSided(101)
    print (prt101)

if __name__ == "__main__":
sys.exit(main())
```

O Arquivo

```
#!/usr/bin/env python
# coding: UTF-8
#
## @package printer
#
#  The Printer class models the usage of paper and ink in a printer. Paper can be added to the
#  printer at any time, and we assume that there is no maximum capacity for paper. A newly
#  constructed printer has a full ink cartridge containing the quantity of ink given by constant
#  INK_CAPACITY. The printer can print one-sided or two-sided. For each side printed, a small
#  quantity of ink is used, as given by constant INK_USAGE. The ink cartridge can be replaced
#  at any time, restoring the ink quantity to the value INK_CAPACITY.
#
#  Please note that you do not need any conditional statements (which we start next week) to
#  complete this assignment. There will be a few places where you need to choose the smaller of
#  two numbers, which can be done with the method min().
#
#  @author Paulo Roma
#  @since 20/06/2016
#  @see http://radek.io/2011/07/21/private-protected-and-public-in-python/

import sys
```

Classe Printer + Main

Main - Executando 1

```
$python printer.py
Constructed(50)
Total paper = 0
Ink quantity = 2.000000
Num. Sheets = 50

printTwoSided(3)
Total paper = 2
Ink quantity = 1.993100
Num. Sheets = 48

printOneSided(2)
Total paper = 4
Ink quantity = 1.988500
Num. Sheets = 46

printOneSided(60)
Total paper = 50
Ink quantity = 1.882700
Num. Sheets = 0
```

Main - Executando 2

```
addPaper(2000)
Total paper = 50
Ink quantity = 1.882700
Num. Sheets = 2000

Sheets used = 50
Out of ink = False
Sheets available = 2000

printOneSided(870)
Total paper = 920
Ink quantity = 0.000000
Num. Sheets = 1130

Out of ink = True

replaceInk()
Total paper = 920
Ink quantity = 2.000000
Num. Sheets = 1130

printTwoSided(101)
Total paper = 50
Ink quantity = 1.770000
Num. Sheets = 0
```