mercredi 4 janvier 2017

Titre

---

specification general ?
pere général A
spécification fils B
A hérité de B

Question Cours

1) L

2) Quelle erreur commet-on quand on fait hériter la classe Pile de la classe Array ?
Entorse à l'encapsulation, une pile n'est pas un tableau, or en faisant hériter Pile de Array c'est ce qu'on dit, et on donne accès à des modifications non autorisées de la pile via les nombreuses méthodes de Array, alors qu'on ne doit accéder qu'au sommet d'une Pile.

3)Difference surcharge et redefinition:
La surcharge consiste a définir plusieurs fonctions ayant le même nom mais pas le même nombre d'arguments ou dans la même classe, c'est du polymorphisme ad hoc. La redéfinissions consiste à définir une fonction qui a le même nom et prend les mêmes arguments qu'une méthode de la classe mère, c'est du polymorphisme d'héritage.

4) Differents type de polymorphisme :
déjà un peu répondu dans la 3 en fait, faut juste y ajouter le polymorphisme paramétrique, non supporté en ruby, qui consiste à pouvoir rendre des objets génériques.

5)Role opérateur <=> et exemple :
sert à comparer en Ruby, il doit retourner un nombre négatif, 0, ou positif suivant que l'objet est avant, au même niveau ou après l'objet comparé. Pour être utilisé de façon optimale, il faut inclure le module Comparable, qui va ajouter à la classe pleins de méthodes comme < ou >= par exemple facilitant la comparaison.

6)Difference structure de répétition type while/for et un iterateur:
Une structure while for permet de répéter une action en fonction d'une condition ou d'un compteur alors qu'un itérateur permet de répéter une opération sur chaque élément dans un conteneur

7)Principe liaison dynamique :
La liaison dynamique consiste à trouver quelles méthodes exécuter lors durant la compilation

8)Avantages inconvénients TDA en C
Les avantages sont la vitesse d'execution et le fait que l'on peut faire des TDA homogènes, les inconvénients sont que c'est pas fait pour ça, on a pas de généricité et on peut pas faire de TDA hétérogènes

9)Pourquoi représentation par cellule chainées sans entêté n'est pas compatible avec le tda liste du cours ?
La représentation par cellule chainée sans en-tête n'est pas compatible car la représentation d'une liste vide est alors différente d'une liste non vide (NULL contre un pointeur non nul). Ainsi, pour appliquer des opérations à une liste vide on doit passer le pointeur du pointeur ce qui n'est plus compatible.

10)Concept de serialisation.Technique et utilisation ?
Sérialisation : permet d'encoder un objet en mémoire pour l'enregistrer dans un fichier ou l'envoyer sur le réseau. L'objet pourra ensuite être désérialisé, c'est à dire lu et décodé pour être chargé en mémoire. Il existe plusieurs formats de sérialisation dont YAML par exemple. En Ruby on fait YAML::dump(objet, fichier) pour sérialiser, et YAML::load(fichier) pour déserialiser.

11)Instruction Ruby x = {1,2,3,4,5,6}
Cette instruction est équivalente à {1 => 2, 3 => 4, 5 => 6}, elle créé une nouvelle table de hachage et l'affecte à x

12)Difference entre langage statique/dynamique
Un langage à typage statique les types sont résolus lors de la compilation alors que pour les langages à typage dynamiques les types sont résolus à l'exécution.

13)Ruby est un langage non typé :
Les variables locales, variables d'instance, paramètres de fonction, etc. ne sont pas typés en Ruby, on peut donc assigner un nombre à une variable pouis lui assigner un objet d'un tout autre type plus tard par exemple

14)Methode ne comporte pas de return, ruby renvoi le résultat de la derniere expression .. pourquoi ?
Cela peut conduire à des entorses à l'encapsulation si l'on ne fait pas attention. Par exemple le fait de retourner un objet non immuable comme une tableau par mégarde va permettre à n'importe qui de le modifier n'importe comment.

15)MIXIN ?
Un MIXIN est un module que l'on utilise pour ajouter des fonctionnalisés de la classe. Le module comparable fournit un grand nombre d'opérateurs de comparaisons (<, <=, >, between?, etc)

```
class Animal
include Comparable
 def <=>(O)
return @membres <=> O.membres() end
end
```

16) YAML ?
YAML:
un module standard lisible et portable qui permet la sauvegarde d'objet lisible et portable qui utilise un format textuel décrivant les structures de données des langages modernes.
require 'yaml'
foo = {1=>1, 2=> » deux», 3=>3,0}
File.open('text.yaml', «w »){ loutl out.puts foo.to_yaml}
roo = YAZML.load_file('test.yaml')

22) Encapsulation :
l'objet est indépendant parce qu'il peut gérer lui même ses données

17)Iterateur ?
Intégrateur permet d'accéder, tout comme la boucle for, à tous les éléments d'un ensemble un à un.
2.upto(7) do l val l
        puts val
end

18) Tag utilisé par rdoc
@abstract description
@api description

19)heritage et composition:
heritage est un mécanisme qui permet de regrouper dans une superclass.
Ce qui commun à plusieurs classe appelé super classe.
externes: ces composants répondent
-soit à des messages différents
-soit différemment à des messages identiques internes: ces composants ont des structures différentes ...

20)Methode destructive
def add!(value)
@res = @res + value
end

@api description

EQUATION ;
1)(8X4 + 3X3 + 5X + 12)

```ruby
class Equation

    def self.valeur(tab)
        coeffs = {}
        tab.reverse.each_with_index { |coeff, puiss|
            coeffs[puiss] = coeff if coeff != 0
        }
        return new(coeffs)
    end

    def initialize(coeffs)
        @coeffs = coeffs
    end
```

2)AFFICHER

```ruby
    def to_s
        str = ""
        premier = true
        @coeffs.each_pair.sort_by{|puiss, coeff| -puiss}.each { |puiss, coeff|
            str += (coeff < 0 ? " - " : " + ") if not premier or coeff < 0
            premier = false
            str += coeff.abs.to_s if coeff != 1 or puiss == 0
            str += "x" if puiss != 0
            str += puiss.to_s if puiss > 1
        }
        return str.strip
    end
```

3)

```
    def +(eq)
        coeffs = self.to_h
        eq.to_h.each_pair { |puiss, coeff|
            coeffs[puiss] = coeffs.fetch(puiss, 0) + coeff
        }
        coeffs.delete_if {|puiss, coeff|
            coeff == 0
        }
        return Equation.new(coeffs)
    end

    def -(eq)
        coeffs = self.to_h
        eq.to_h.each_pair { |puiss, coeff|
            coeffs[puiss] = coeffs.fetch(puiss, 0) - coeff
        }
        coeffs.delete_if {|puiss, coeff|
            coeff == 0
        }
        return Equation.new(coeffs)
    end

    def to_h
        return @coeffs.dup
    end
end
```

TESTTTT

```
monEquation = Equation.valeur([8, 3, 0, 5, 12])
puts monEquation
tonEquation = Equation.valeur([10, 0, 5, 0, 0, 1])
puts tonEquation
sonEquation = monEquation + tonEquation
puts sonEquation
sonEquation = monEquation - tonEquation
puts sonEquation
```

COMPARAISON

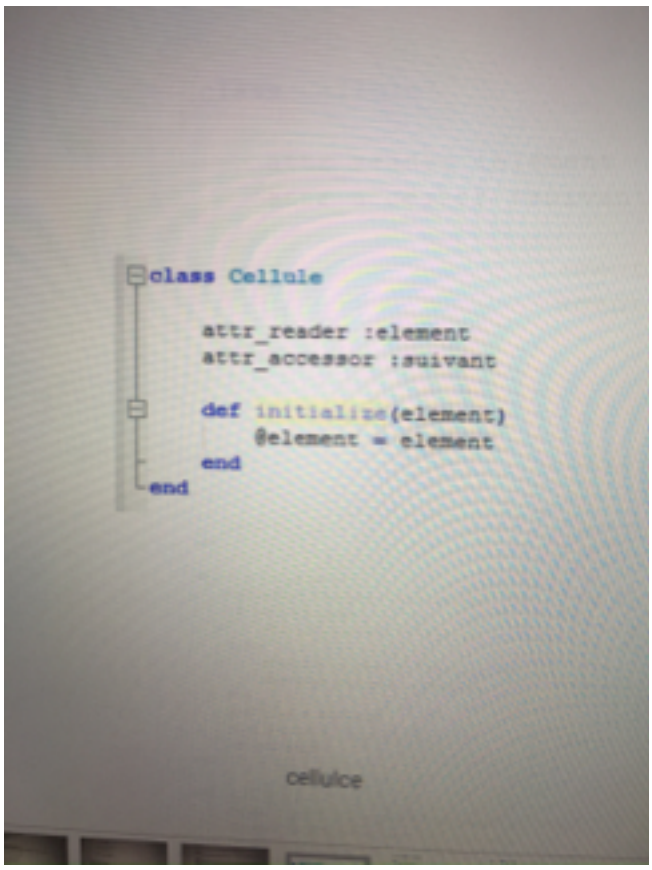
```
    include Comparable

    def <=>(eq)
        m1 = @coeffs.each_value.reduce(:+) / @coeffs.size
        coeffs = eq.to_h
        m2 = coeffs.each_value.reduce(:+) / coeffs.size
        return m1 - m2
    end
id
```

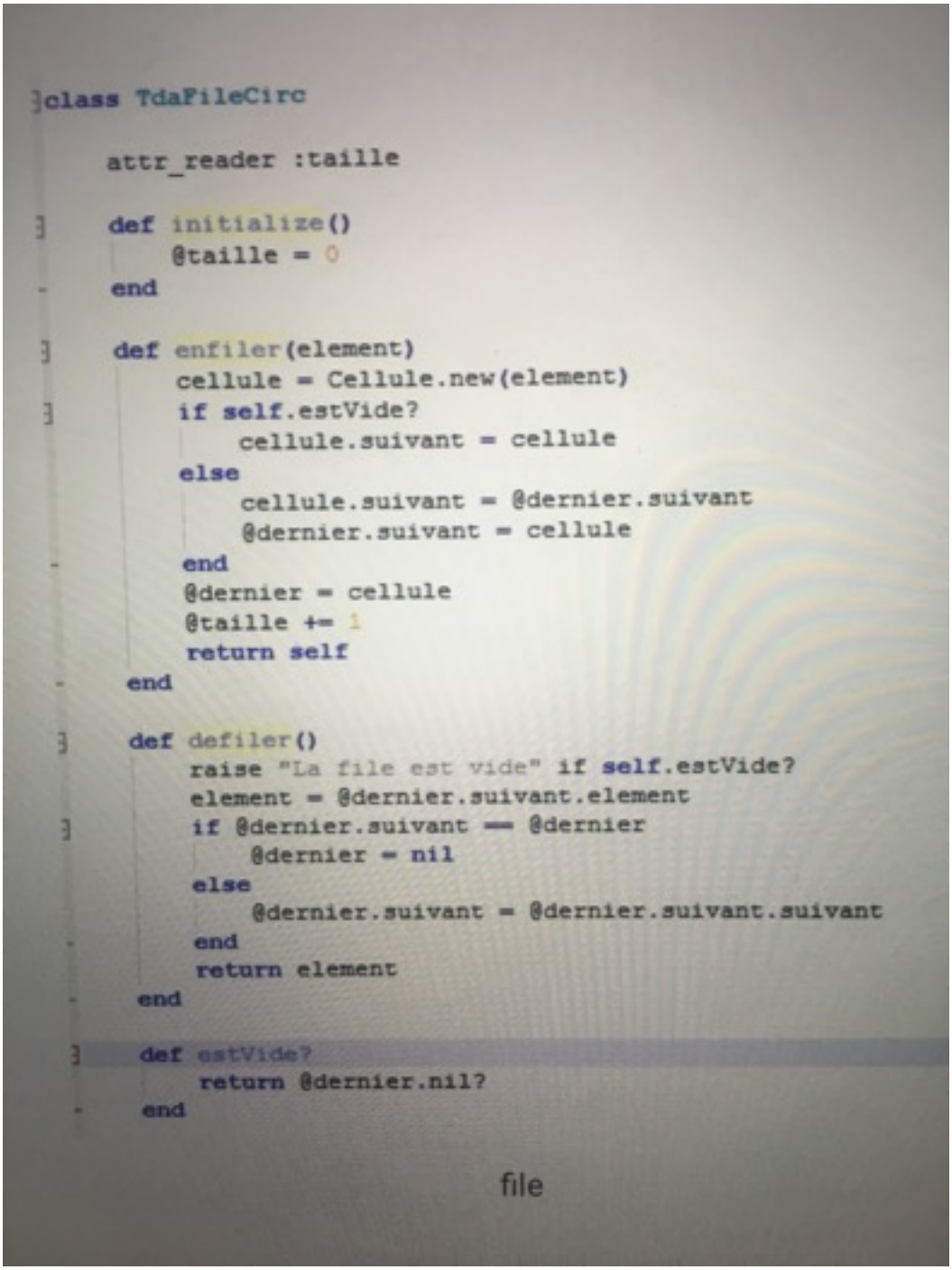
TEST COMPARAISON

```
puts monEquation < tonEquation
puts monEquation > tonEquation
puts monEquation == tonEquation
```

Tests comparaison

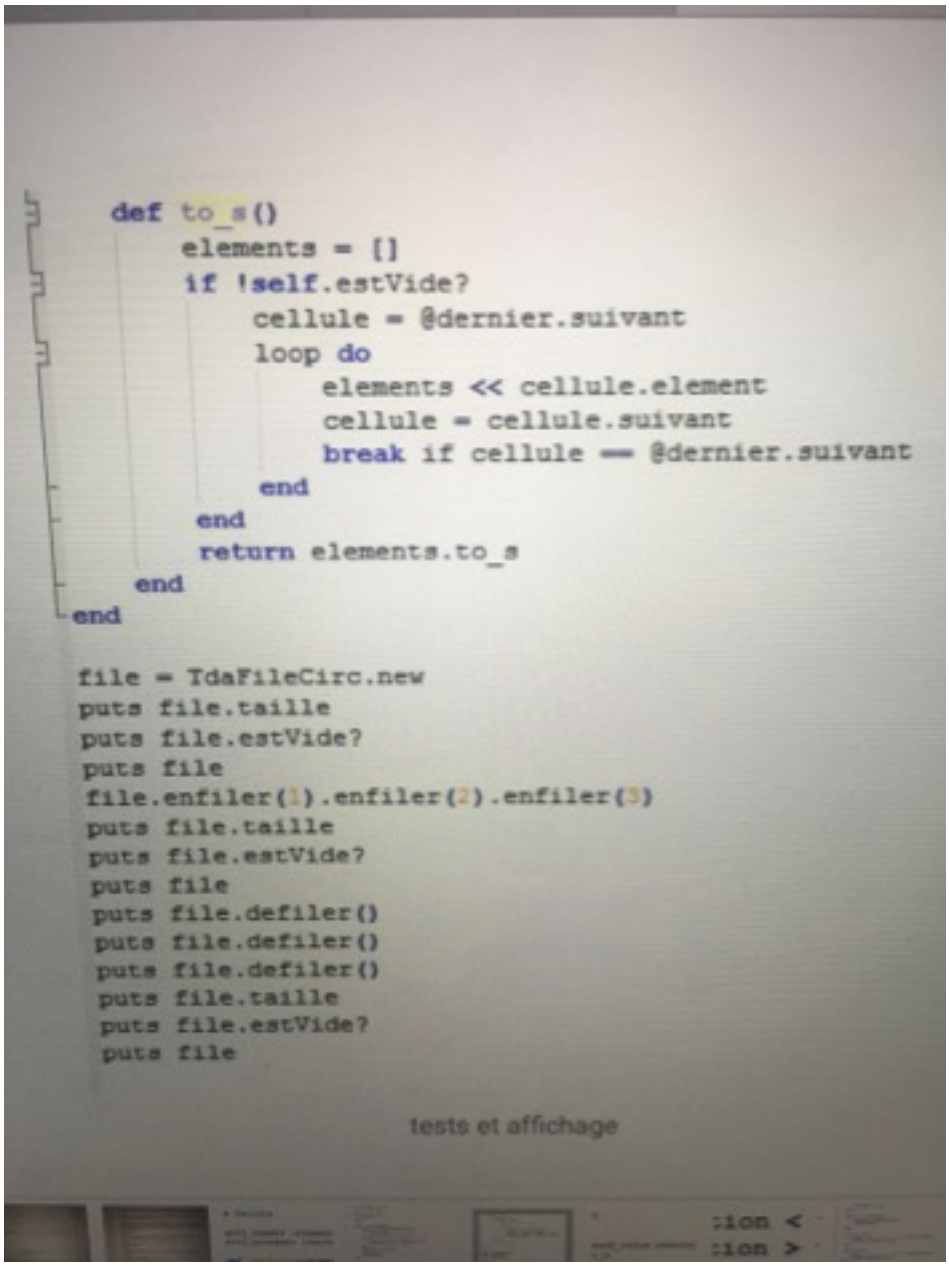
EXERCICE FILE CIRCULAIRE





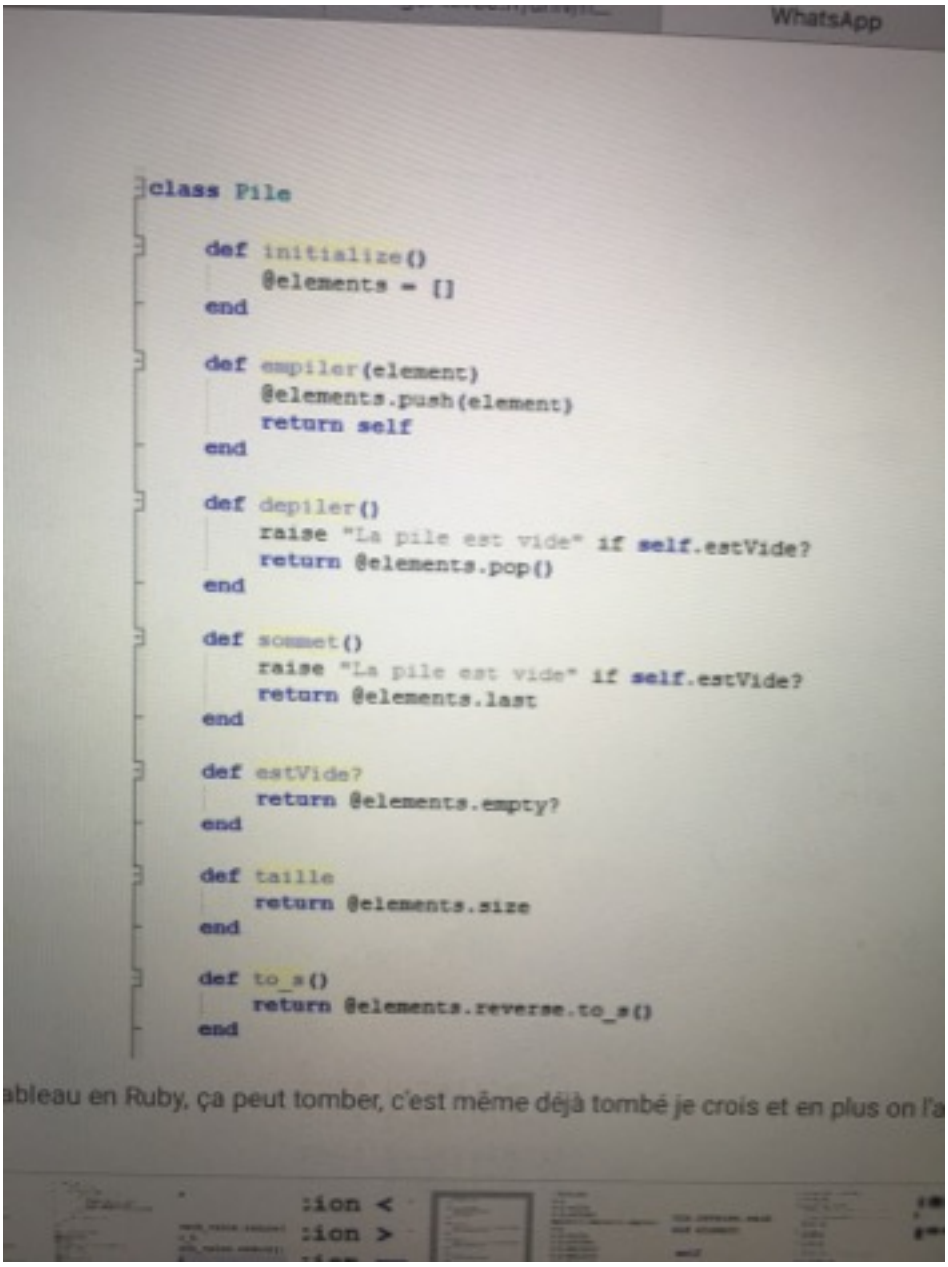
```ruby
class TdaFileCirc

    attr_reader :taille

    def initialize()
        @taille = 0
    end

    def enfiler(element)
        cellule = Cellule.new(element)
        if self.estVide?
            cellule.suivant = cellule
        else
            cellule.suivant = @dernier.suivant
            @dernier.suivant = cellule
        end
        @dernier = cellule
        @taille += 1
        return self
    end

    def defiler()
        raise "La file est vide" if self.estVide?
        element = @dernier.suivant.element
        if @dernier.suivant == @dernier
            @dernier = nil
        else
            @dernier.suivant = @dernier.suivant.suivant
        end
        return element
    end

    def estVide?
        return @dernier.nil?
    end
```

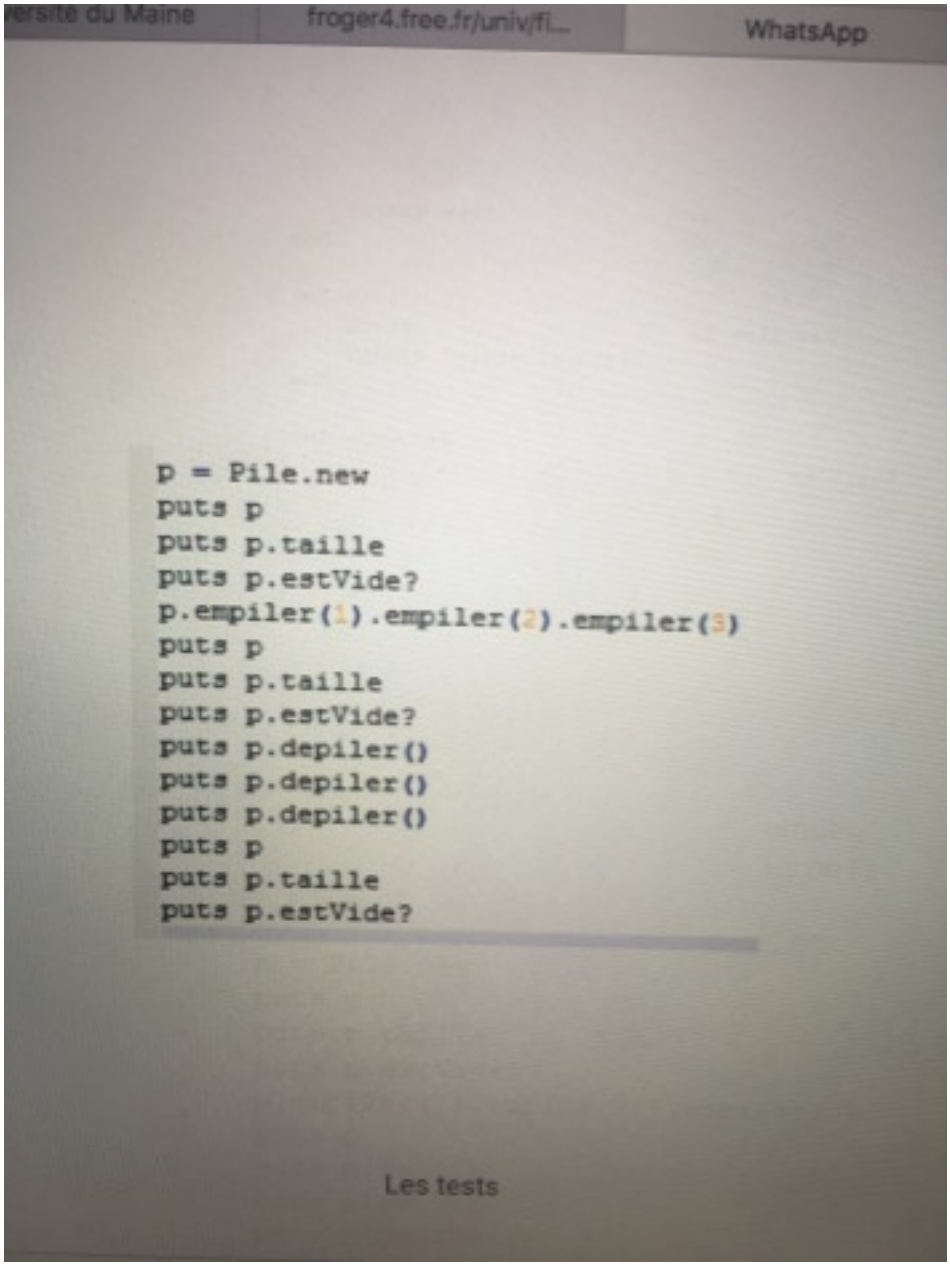
file



```
    def to_s()
        elements = []
        if !self.estVide?
            cellule = @dernier.suivant
            loop do
                elements << cellule.element
                cellule = cellule.suivant
                break if cellule == @dernier.suivant
            end
        end
        return elements.to_s
    end
end

file = TdaFileCirc.new
puts file.taille
puts file.estVide?
puts file
file.enfiler(1).enfiler(2).enfiler(3)
puts file.taille
puts file.estVide?
puts file
puts file.defiler()
puts file.defiler()
puts file.defiler()
puts file.taille
puts file.estVide?
puts file
```

tests et affichage


WhatsApp
class Pile
def initialize()
@elements = []
end
def empiler(element)
@elements.push(element)
return self
end
def depiler()
raise "La pile est vide" if self.estVide?
return @elements.pop()
end
def sommet()
raise "La pile est vide" if self.estVide?
return @elements.last
end
def estVide?
return @elements.empty?
end
def taille
return @elements.size
end
def to_s()
return @elements.reverse.to_s()
end
ableau en Ruby, ça peut tomber, c'est même déjà tombé je crois et en plus on l'a
:ion <
:ion >


ersité du Maine
froger4.free.fr/univ/fi...
WhatsApp
p = Pile.new
puts p
puts p.taille
puts p.estVide?
p.empiler(1).empiler(2).empiler(3)
puts p
puts p.taille
puts p.estVide?
puts p.depiler()
puts p.depiler()
puts p.depiler()
puts p
puts p.taille
puts p.estVide?
Les tests