Code Python : Test et Intégration

Classe Book

```
import Atmosphere

class Livre(object) :
    def __init__(self,title,author,summary,epub):
        self.title = title
        self.author = author
        self.summary = summary
        self.text = epub
        self.ambiance
        self.currentText

      def get_items(self):
        # Returns all the datas of the book
        return self.title, self.author, self.summary, self.text

      def get_ambiance(self):
        # Returns the ambiance spread by this text, a piece of the book
        return self.ambiance

    def set_ambiance(self, newAmbiance):
        # Changes the atmosphere
        self.ambiance =  newAmbiance

    def get_currentText(self):
        # Returns the current text that is read by the user
        return self.currentText
```

Classe Classifior

```
import Atmosphere

class Classifior(object) :
    def __init__(self, trainingTime, accuracyRate):
        vectorisor = ""
        self.trainingTime = trainingTime
        self.accuracyRate = accuracyRate

    def getTrainingTime(self):
        # Returns the training time needed for this classifior

        return self.trainingTime

    def getAccuracyRate(self):
        # Returns the accuracy rate reached by this classifior
        return self.accuracyRate

    def predict(self,text):
        # Returns the atmosphere related to the passage
        #TO BE COMPLETED
```

```
        myAtmosphere = Atmosphere()
        return myAtmosphere

    def convertToEpub(self,book):
        # Transfers the text to an ePub
        newText = None
        # TO BE COMPLETED
        return newText
```

Classe Atmosphere :

```
class Ambiance(object) :

    def __init__(self):
        # Takes a list of songs and one smell
        self.song = ["" for x in range(10)]
        self.olfactive = "smell"

    def get_atmosphere(self):
        return self.song,self.smell
```

Classe Test :

```
import Atmosphere
import Book
import Classifior
import time

class Utilisateur(object) :
    def __init__(self):
        self.classifieur
        # The classifior to get the atmospheres

        self.book
        # The current book that is choosen to read by the user

        self.listOfBook
        # The list of books that is saved by the user

        self.currentBookDownload
        # The book that will be downloaded and saved by the user

    def getAllBooks(self):
        # Returns all the books available
        return self.listOfBook

    def getBookDownloaded(self) :
        # Returns the book that the user choosed to download
        return self.currentBookDownload

    def getTextCurrentlyRead(self) :
    # Returns the text that the user is currently reading
        return self.book.get_currentText
```

```
    def lecture_ambiance_sonore(self,i):
        #Plays a song whether the raspberry pi is connected or not
        if (self.is_connected_RP ):
            # TO BE COMPLETED
            self.display_RP(i)
        else :
            # TO BE COMPLETED
            self.display(i)
        return None

    def display_available_books(self):
        # Returns a list of the book available
        # TO BE COMPLETED
        return None

    ### Raspberry Pi

    def lecture_ambiance_olfactive(self, listAmbiance):
        # Sends to the raspberry pi to display the atmosphere
        # TO BE COMPLETED
        return None

    def display(self, name):
        # Plays and audio file without the raspberry pi
        # TO BE COMPLETED
        return None

    def display_RP(self, name):
        #Plays an audio file in the raspberry pi
        # TO BE COMPLETED
        return None

    def is_connected_RP(self):
        # Returns true if it is connected to the raspberry pi
        # TO BE COMPLETED
        return False

    def connection_to_RP(self):
        # Connects to the raspberry Pi
        # TO BE COMPLETED
        return None

    def getDisplayed_RP(self):
        # Returns the atmosphere that is currently displayed bien the
RP
        return None
    ### Classification

    def getClassifieur(self):
        # Initialise the classifior
        # TO BE COMPLETED
        return None

    def getAmbiance(self, text):
```

```
        # Takes a passage and gives the atmosphere
        ambiance = self.classifieur.predict(text)
        return ambiance

    def getAmbianceBook(self, Livre):
        # Take a book and returns two lists
        # The first one the atmosphere
        # The second one the lines of atmosphere associated
        list_ambiance = []
        list_passage = []
        for x in Livre :
            # x is a passage
            list_ambiance.append(self.getAmbiance(x))
            list_passage.append(x)
        return list_ambiance,list_passage

### Tests

###     Tests Classifieur

def testTimeClassifior(user) :
    # It gives the time to get an atmosphere
    start = time.time()
    testToText = ""
    ambianceGiven =  user.getAmbiance(testToText)
    if (ambianceGiven is not None ) :
        end = time.time()
        return abs(end-start)
    else :
        print "The Classifior does not give the atmosphere"

def testTrainingTimeClassifior(user) :
    # It gives the time to get the classifior time training

    return user.classifior.getTrainingTime()

def testAccuracyRateClassifior(user) :
    # It gives the accuracy rate reached by this classifior

    return user.classifior.getAccuracyRate()

def testReturnAmbianceClassifior(user) :
    # It returns a boolean that checks if an ambiance is returned by
the classifior
    a = type(user.classifior.predict(user,
user.getTextCurrentlyRead(user)))
    return (type(a) == 'Ambiance')

###     Tests Raspberry pi

def isGoodConnection(user):
    # Returns if the connection of the user is the same as the RP
displays
    user.connection_to_RP()
```

```
    return (user.is_connected_RP()==True)

def isWellDisplayed(user) :
    # Returns if the song that is displayed by the RP is the good one
    text = user.getTextCurrentlyRead
    ambiance = user.getAmbiance(text)
    user.display_RP(ambiance)
    return (user.getDisplayed_RP == ambiance)

### Tests BDD

def isRespectedFormat(user) :
    # Tests if all the books that are saved by the user
    # have the good format
    formatWanted = ".epub"
    for x in user.getAllBooks :
        if x.getClass() != formatWanted :
            return False
    return True

def tranistionTxtToEpub(user) :
    # Tests if the classifieur transfers a text into an epub
    textToTest = user.getAllBooks[0]
    return (user.getClassifieur.convertToEpub(textToTest).getClass ==
".epub")
def isAssociated(user) :
    # Returns if the classifieur gives a non null atmosphere
    ambiance = user.getClassifieur.predict(user.getCurrentBook)
    return (ambiance != None)
```