Clase Diccionario

Solución utilizando **list** de Python

```
import estructura

# Estructura que almacena una palabra con su significado
# registro: palabra (str) significado (str)
estructura.mutable("registro","palabra significado")

# Clase Diccionario
# Almacena registros de palabras con su significado
# Campos:
#    __D: list (registro)
class Diccionario:
    # __init__ : str -> list(registro)
    #crea diccionario con palabras en archivo x
    #ejs: D=Diccionario("diccionario.txt")
    def __init__(self,x):
        assert type(x)==str
        self.__D=[] #list()
        f=open(x)
        for linea in f:
            i=linea.index(".")
            reg=registro(linea[0:i],linea[i+1:-1])
            self.__D.append(reg)
        f.close()


    #buscar : str -> str
    def buscar(self,palabra):
        i=self.indice(palabra,self.__D)
        if i<0:
            return None
        else:
            return self.__D[i].significado

    #cambiar : str str -> bool
    def cambiar(self,palabra,significado):
        i=self.indice(palabra,self.__D)
        if i<0:
            return False
        else:
            self.__D[i].significado = significado
            return True


    # borrar : str -> bool
```

```
    def borrar(self,palabra):
        i=self.indice(palabra,self.__D)
        if i<0:
            return False
        else:
            self.__D.pop(i)
        return True


    #indice: str list(registro) -> int
    #indice de primer x en L (-1 si no está)
    #L: lista ordenada y sin repeticiones
    #ej:indice('b',[registro('a',1),registro('b',2)])==1
    def indice(self,palabra,L):
        ip=0;
        iu=len(L)-1
        while ip<=iu:
            im=(ip+iu)//2
            if L[im].palabra == palabra:
                return im
            if L[im].palabra<palabra:
                ip=im+1
            else:
                iu=im-1
        return -1 #no está

    #agregar : str str -> bool
    def agregar(self,palabra,significado):
        reg=registro(palabra,significado)
        for i in range(len(self.__D)):
            if self.__D[i].palabra == palabra:
                return False
            if self.__D[i].palabra > palabra:
                #insertar en índice i
                self.__D.insert(i,reg)
                return True
        #agregar al final de la lista
        self.__D.append(reg)
        return True

    # grabar: str -> None
    # efecto: graba diccionario en el archivo indicado, una palabra y significado por
    def grabar(self,archivo):
        assert type(archivo)==str
        f=open(archivo,"w")
        for registro in self.__D:
            linea = registro.palabra+"."+registro.significado
            f.write(linea+"\n")
        f.close()


# Test diccionario
```

```
class TestDiccionario:
    def __init__(self):
        a=open("D.txt","w")
        a.write("hola.hi\n")
        a.close()
        self.__D=Diccionario("D.txt")

    def test(self):
        assert self.__D.buscar("hola")=="hi"
        assert self.__D.cambiar("hola","hello")
        assert self.__D.buscar("hola")=="hello"
        assert self.__D.borrar("hola")
        assert self.__D.buscar("hola")==None
        assert self.__D.agregar("hola","hi")
        assert self.__D.buscar("hola")=="hi"
        self.__D.grabar("D.txt")

        for l in open("D.txt"):
            assert l=="hola.hi\n"

TestDiccionario().test()
```