uc3m | Universidad Carlos III de Madrid

Grado en Ciencia e Ingeniería de Datos, 2018-2019

Unit 1. Abstract Data Types

Data Structures and Algorithms (DSA)

Author: Isabel Segura-Bedmar

Index

Introduction

- Programming languages provide **primitive data types**:
  - **simple data types:** cannot be decomposed into smaller parts. Examples: Integer, Float, String, Boolean.
  - **complex data types**: are constructed of multiple components. Examples: List.
- Primitive types may not be sufficient for solving complex problems.
- Programmers can define their own **user-defined types**.

Introduction

```
class CreditCard:
  """A credit card"""
  def __init__ (self, customer, idCard, limit):
    """Creates a new credit card object"""
    self._customer=customer #the name of the customer
    self._idCard=idCard #the id of the credit card
    self._limit=limit   #credit limit
    self._balance=0     #the initial balance is 0
```

Abstraction

- Focus on relevant details for a problem and rule out irrelevant details.

Student
NIA
email
...

Patient
Birthdate
Weight
Allergies
...

Index

What is an abstract data type (ADT)?

- User-defined type that specifies a **set of data values** and a **collection of operations** to manipulate these values.
- An **ADT** is **independent of its implementation**.
- Focus on the **‘what’** must do instead of **‘how’** do it.

Example: Complex Number ADT

A complex number consists of two parts:

- a real part, say a
- an imaginary part, say b

The value of such a complex number is: a+ib, where $i=\sqrt{-1}$

Example: Complex Number ADT

Operations on a complex number:

- *The **negation** of a complex number a+ib is:*

-a + i(-b)

- ***Addition** or **subtraction:***
    - *(a+ib) + (c+id) = (a+c) + (b+d)i*
    - *(a+ib) - (c+id) = (a-c) + (b-d)i*

Example: Complex Number ADT

Operations on a complex number:

- ***multiplication:***

  $(a+ib) * (c+id) = (ac-bd) + (ad+bc)i$

- ***absolute value*** *or* ***module:***

  $|a+ib| = \sqrt{(a^2+b^2)}$

Example: Date ADT

- A date represents a single day in our calendar (e.g December 25, 2018 AC).
- The operations are:
  - **Date**(*day,month,year*): creates a new date instance.
  - **toString**(): returns a string representation in the format '*dd/mm'/yyyy*'.
  - **day**(): returns the day number of this date.
  - **month**(): returns the month number of this date.
  - **year**(): returns the year number of this date.

Example: Date ADT

- More operations:.
  - **monthName**(): returns the month name of this date.
  - **numDays**(other): returns the number of days between this date and the given date.
  - **isLeapYear**(): return True if this date falls in a leap year, and False otherwise.

Example: Date ADT

- More operations:.
  - **compareTo**(other): compares this date to the *other* to determine their logical order.
    - If this date is before to the *other*, returns -1.
    - If both dates are the same, return 0. I
    - If this date is after to the *other*, returns 1.

Index

What is a data structure?

- A data structure is a representation (**implementation**) of an ADT in a programming language.
- In Python, ADT are represented by classes.
- An ADT may have several implementations (data structures).
- Focus on how they store and organize the data and what operations are available for manipulating the data.

Data structure for Complex Number ADT

```
import math

class Complex:

    def __init__(self, re=0, im=0):
        self.re = re
        self.im = im

    def neg(self):
        return Complex(-self.re, -self.im)

    def sum(self, other):
        return Complex(self.re+other.re, self.im+other.im)

    def sub(self, other):
        return Complex(self.re-other.re, self.im-other.im)
```

Data structure for Complex Mumber ADT

```
def mul(self, other):
    return Complex(self.re*other.re - self.im*other.im,
                   self.re*other.im + self.im*other.re)

def abs(self):
    return math.sqrt(math.pow(self.re,2)+math.pow(self.im,2))

def toString(self):
    return 'Complex(%r, %r)' % (self.re, self.im)
```

Data structure for Complex Number ADT

```
c=Complex(3,5)
print(c.toString())
print('Neg of {} = {}'.format(c.toString(),c.neg().toString()))
print('Module of {} = {}'.format(c.toString(),c.abs()))



other=Complex(2,1)
print('{}+{} = {}'.format(c.toString(),other.toString(),c.sum(other).toString()))
print('{}-{} = {}'.format(c.toString(),other.toString(),c.sub(other).toString()))
print('{}*{} = {}'.format(c.toString(),other.toString(),c.mul(other).toString()))
```

Summary

- An ADT defines **what operations**, but not **how to do** (implement) **them**.
- An ADT may have different implementations.
- A data structure is an implementation of an ADT.
- In Python, **classes** allow to implement ADTs.