ECE12: Introduction to Programming
Lecture 14

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 14: Overview

- Object-oriented Programming
  - Introduction
  - Concepts and Terminology
    - Class
    - Object
  - Example: **`class Time`**
    - Class definition
    - Documentation strings
    - Constructor
    - Data members
    - Methods

Object-Oriented Programming

- Introduction
  - Before: *Structured Programming*
    - Literals, identifiers, types, expressions
    - Statements, control flow, functions
    - Procedural programming, *action-oriented*
  - Now: *Object-Oriented Programming (OOP)*
    - Classes
    - Objects
- Background
  - The real world is composed of objects
    - people, animals, plants, cars, planes, buildings, ...
  - An object can be seen as an abstraction of its components
    - we see objects on a screen (not a bunch of pixels)
    - we see a beach (rather than grains of sand)
    - we see a forest (rather than trees)
    - we see buildings (rather than bricks)
  - A class is like a blue-print for an object

Object-Oriented Programming

- Concepts and Terminology
  - Object
    - Abstraction, model of real-world object
    - Has *attributes*
      - name, size, color, weight, ...
    - Exhibits *behavior*
      - people sleep, eat, walk, talk, ...
    - Uses *communication*
      - message passing
  - Class relationship
    - Classes of objects have the same characteristics
      - Class automobile contains
        - » sports car, limousine, pick-up, truck, ...
    - *Inheritance* (multiple inheritance)
      - A convertible is a sports car with a removable roof
      - A convertible is also an automobile
    - Classes of objects are derived from existing classes and add characteristics of their own

Object-Oriented Programming

- Key concepts
  - Hierarchy
  - Encapsulation
    - Attributes: data members
    - Behavior: function members, methods
    - Interfaces: communication attributes and methods
  - Information hiding
  - Reuse
- Terminology
  - Object
    - Instance of a class
    - Instantiation: creation of an object of a class
    - Destruction: deletion of an object
  - Class:
    - Abstract data type (ADT)
      - aka. user-defined type
    - Constructor: creation of objects
    - Destructor: deletion of objects

Object-Oriented Programming

- Example: **`class Time`**
  - Program **`time1.py`** (part 1/2)

```
# time1.py: abstract data type for representation of time
#           (version 1)
# author: Rainer Doemer
# 02/17/04 RD    initial version (similar to figure 7.1)

# class definition
class Time:
    """abstract data type for representation of time"""

    def __init__(self):           # constructor
        """creates a time object initialized to 12am"""
        self.hour   = 0  # 0-23   # data members
        self.minute = 0  # 0-59
        self.second = 0  # 0-59

    def Print(self):              # method
        """prints the value of a time object"""
        print "%02d:%02d:%02d" %  \
                (self.hour,self.minute,self.second)
...
```

Object-Oriented Programming

- Example: **`class Time`**
  - Program **`time1.py`** (part 2/2)

```
...
    def PrintAMPM(self):  # method
        """prints the time in am/pm notation"""
        h = self.hour % 12
        if h == 0:
            h = 12
        if self.hour < 12:
            ampm = "am"
        else:
            ampm = "pm"
        print "%2d:%02d:%02d %s" %\
                (h,self.minute,self.second,ampm)
```

Object-Oriented Programming

- Example: **`class Time`**
  - Notes (1):
    - Class definition consists of
      - Class header (keyword **`class`**, identifier **`Time`**, colon)
      - Class body (indented block of attributes and methods)
        - » contains methods **`__init__`**, **`Print`**, and **`PrintAMPM`**
    - Documentation strings
      - Triple-quoted strings (by convention)
      - Inserted between header and body
      - Optional for modules, functions, classes, methods
      - Available in attribute **`__doc__`** for inspection
    - Class constructor **`__init__`**
      - Special method for object initialization
        - » creates and initializes attributes **`hour`**, **`minute`**, and **`second`**
      - Called implicitly whenever an object of the class is created
      - Must not return any value (**`None`**)

Object-Oriented Programming

- Example: **`class Time`**
  - Notes (2):
    - Object reference **`self`**
      - Aka. *object reference argument* or *class instance object*
      - Called **`self`** by convention
      - First (explicit!) argument of every class method
      - Implicitly supplied when a method of an object is called
      - in C++, **`self`** is called **`this`**
    - Class methods
      - functions that operate on an object
        - » **`Print`**, **`PrintAMPM`**
      - require first argument **`self`** which represents the object
      - **`self`** is used to access the attributes
        - » **`self.hour`**
        - » **`self.minute`**
        - » **`self.second`**

Object-Oriented Programming

- Example: **`class Time`**
  - Notes (3):
    - Class namespace
      - Every class has its own namespace
      - Contains class attributes and class methods (which are shared among all instances of the class)
      - Access by use of dot-operator
        - » from inside the class: through object reference **`self`**
        - » from outside the class: through class name
    - Object namespace
      - Every object has its own namespace
      - Contains object attributes and object methods
      - Is typically populated by the constructor
      - Access by use of dot-operator
        - » from inside the class: through object reference **`self`**
        - » from outside the class: through object name

Object-Oriented Programming

- Example: **`class Time`**
  - Interactive use of module **`time1.py`** (part 1/2)

```
% ls
time1.py
% python
>>> from time1 import Time
>>> type(Time)
<type 'class'>
>>> dir(Time)
['Print', 'PrintAMPM', '__doc__', '__init__', '__module__']
>>> print Time.__doc__
abstract data type for representation of time
>>> print Time.__module__
time1
>>> t1 = Time()
>>> type(t1)
<type 'instance'>
>>> print t1
<time1.Time instance at 0x8178e2c>
>>> dir(t1)
['Print', 'PrintAMPM', '__doc__', '__init__', '__module__',
'hour', 'minute', 'second']
...
```

Object-Oriented Programming

- Example: **`class Time`**
  - Interactive use of module **`time1.py`** (part 2/2)

```
...
>>> t1.hour
0
>>> t1.minute
0
>>> t1.second
0
>>> t1.Print()
00:00:00
>>> t1.PrintAMPM()
12:00:00 am
>>> t1.hour = 15
>>> t1.minute = 30
>>> t1.PrintAMPM()
 3:30:00 pm
```

Object-Oriented Programming

- Example: **`class Time`**
  - Interactive use of module **`time1.py`**
  - Notes:
    - File **`time1.py`** can be used as a module for **`import`**
      - Programs can be split into multiple files
      - Class **`Time`** defined in module **`time1.py`** is imported
    - Class contents can be listed with **`dir()`**
    - Documentation strings are compiled into **`__doc__`**
    - Class instantiation
      - » **`t1 = Time()`**
      - A new object is created by calling the class as a function
      - Implicitly the class constructor will be called
    - Object contents can be listed with **`dir()`**
    - Object members can be accessed with the dot operator
      - » **`t1.hour`**