IT Technology Network
Assignment 14
Inheritance

Lillebaelt Academy of
Professional Higher Education

Authors:
Milan Kristof Vince: mila1025@edu.eal.dk

Sunday,12 February 2017

Introduction

This report is about an guide to show my assignment what I get in Lillebaelt Academy. During of my works I use Notepad++ and Pycharm to convert Python code what I used during create program. The audience of this document will be people in the class who are also handle python programming, but the audience must have done and understand the first 12 chapters in" Starting out with Python 2nd edition" by Tony Gaddis.

Employee and ProductionWorker Class

“Write an Employee class that keeps data attributes for the following pieces of information: • Employee name • Employee number Next, write a class named ProductionWorker that is a subclass of the Employee class. The ProductionWorker class should keep data attributes for the following information: • Shift number (an integer, such as 1, 2, or 3) • Hourly pay rate”

Program Display:

```
import pickle


LOOK_UP = 1
ADD = 2
CHANGE = 3
DELETE = 4
QUIT = 5

FILENAME = 'employees.dat'

class Employee:
    def __init__(self, name, number, department, job):
        self.__name = name
        self.__id = number
        self.__department = department
        self.__job = job
# Set the name
    def set_name(self, name):
        self.__name = name
# Set the number
    def set_number(self, number):
        self.__number = number
# Set the department
    def set_department(self, department):
        self.__department = department
# Set the job
    def set_job(self, job):
        self.__job = job
# Get the name
    def get_name(self):
        return self.__name
# Get the number
    def get_number(self):
        return self.__number
# Get the department
    def get_department(self):
        return self.__department
# Get the job
    def get_job(self):
        return self.__job
# Return string to represent the object
    def __str__(self):
        return "Name: " + self.__name + \
               "\nid: " + self.__number + \
               "\ndepartment: " + self.__department + \
               "\njob: " + self.__job
```

```
class ProductionWorker(Employee):
    def __init__(self, name, number, department, job, shift, hpr):
        Employee.__init__(self, name, number, department, job)
        self.__shift = shift
        self.__hpr = hpr
# Set Shift
    def set_shift(self, shift):
        self.__shift = shift
# Set HPR
    def set_hpr(self, hpr):
        self.__hpr = hpr
# Get Shift
    def get_shift(self):
        return self.__shift
# Get HPR
    def get_hpr(self):
        return self.__hpr
# Return string to represent the object
    def __str__(self):
        return "shift: " + self.__shift + \
               "\nhpr: " + self.__hpr


# Main function


def main():
    mycontacts = load_employee()

    choice = 0


    while choice != QUIT:
# Get the user's menu choice.
        choice = get_menu_choice()

        if choice == LOOK_UP:
            look_up(mycontacts)
        elif choice == ADD:
            add(mycontacts)
        elif choice == CHANGE:
            change(mycontacts)
        elif choice == DELETE:
            delete(mycontacts)

# Save the mycontacts dictionary to a file.
    save_contacts(mycontacts)


def load_employee():
    try:
# Open the contacts.dat file.
        input_file = open(FILENAME, 'rb')

        employees_dct = pickle.load(input_file)

        input_file.close()
    except IOError:
```

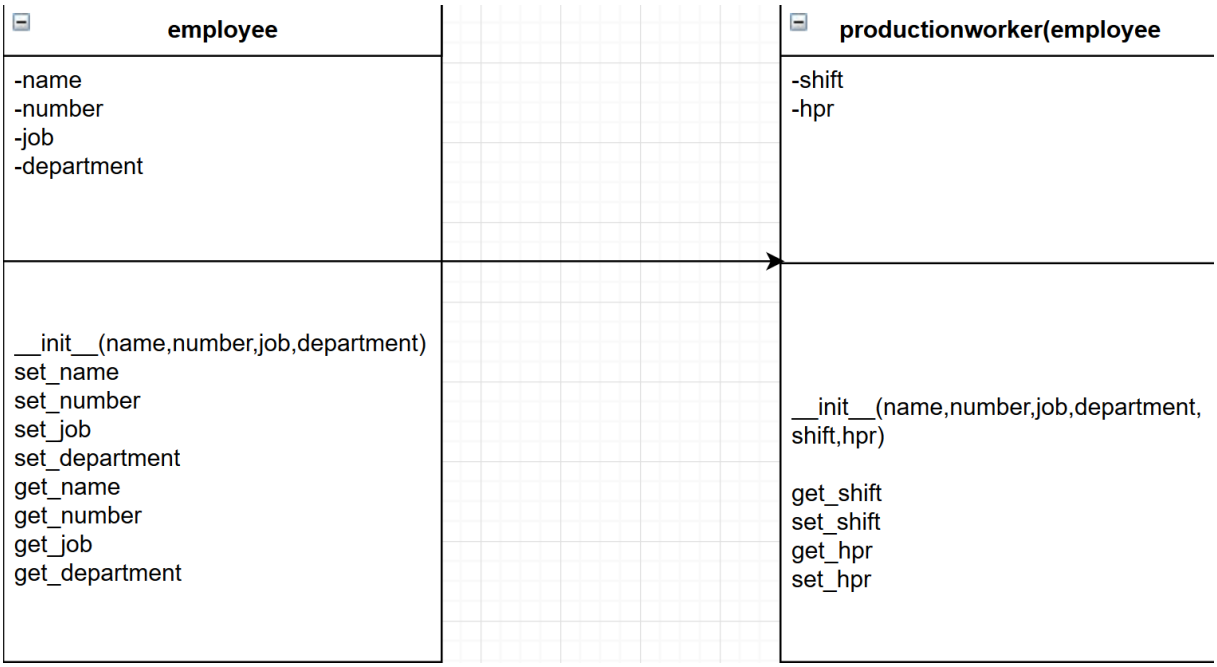
```
# If could not open the file, then create blank dictionary
        employees_dct = {}

    return employees_dct


def get_menu_choice():
# The get menu choice function displays the menu
    print()
    print('Menu')
    print('---------------------------')
    print('1. Look up a contact')
    print('2. Add a new contact')
    print('3. Change an existing contact')
    print('4. Delete a contact')
    print('5. Quit the program')
    print()

    choice = float(input('Enter your choice: '))

    while choice < LOOK_UP or choice > QUIT:
        choice = int(input('Enter a valid choice: '))

    return choice



# The look_up function looks up an item in the
def look_up(mycontacts):

    name = input('Enter a name: ')

    print(mycontacts.get(name, 'That name is not found.'))


def add(mycontacts):
# The add function adds a new employee
    name = input('Name: ')
    number = input('Id number: ')
    department = input('Department: ')
    job = input('Job title: ')
    shift = input('Work shift(1 day/2 afternoon /3 Night): ')
    hpr = input('Hourly pay rate: ')


# Create a Contact object named entry.
    entry = ProductionWorker(name, number, department, job, shift, hpr)

    if name not in mycontacts:
        mycontacts[name] = entry
        print('The entry has been added.')
    else:
        print('That name already exists.')


def change(mycontacts):
# The change function changes an existing employee
    name = input('Enter a name: ')

    if name in mycontacts:
# Update the information on an employee
```

```
        number = input('Enter the new id number: ')
        department = input('Enter the new department: ')
        job = input('Enter the new job title: ')
        shift = input('Enter the new work shift: ')
        hpr = input('Enter the new Hourly pay rate: ')


# Create a contact object named entry.
        entry = ProductionWorker(number, department, job, shift, hpr,'')

        mycontacts[name] = entry
        print('Information updated.')
    else:
        print('That name is not found.')


def delete(mycontacts):

    name = input('Enter a name: ')

    if name in mycontacts:
        del mycontacts[name]
        print('Entry deleted.')
    else:
        print('That name is not found.')


def save_contacts(mycontacts):

    output_file = open(FILENAME, 'wb')

    pickle.dump(mycontacts, output_file)

    output_file.close()
# Call the main function.
main()
```

Program Output:

```
Enter your choice: 2
Name: Milan
Id number: 13123
Department: IT
Job title: Network Specialist
Work shift(1 day/2 afternoon /3 Night): 1
Hourly pay rate: 100
The entry has been added.


Enter your choice: 1
Enter a name: Milan
shift: 1
hpr: 100
```

UML Diagram:


employee
-name
-number
-job
-department
__init__(name,number,job,department)
set_name
set_number
set_job
set_department
get_name
get_number
get_job
get_department
productionworker(employee
-shift
-hpr
__init__(name,number,job,department,
shift,hpr)
get_shift
set_shift
get_hpr
set_hpr

ShiftSupervisor Class

"Write a ShiftSupervisor class that is a subclass of the Employee class you created in Programming Exercise 1. The ShiftSupervisor class should keep a data attribute for the annual salary and a data attribute for the annual production bonus that a shift supervisor has earned. Demonstrate the class by writing a program that uses a ShiftSupervisor object."

Program Display:

```
import pickle


LOOK_UP = 1
ADD = 2
CHANGE = 3
DELETE = 4
QUIT = 5

FILENAME = 'employees.dat'

class Employee:
    def __init__(self, name, number, department, job):
        self.__name = name
        self.  id = number
        self.__department = department
        self.__job = job
# Set the name
    def set_name(self, name):
        self.__name = name
# Set the number
    def set_number(self, number):
        self.__number = number
# Set the department
    def set_department(self, department):
        self.__department = department
# Set the job
    def set_job(self, job):
        self.__job = job
# Get the name
    def get_name(self):
        return self.__name
# Get the number
    def get_number(self):
        return self.__number
# Get the department
    def get_department(self):
        return self.__department
# Get the job
    def get_job(self):
        return self.__job
# Return string to represent the object
    def __str__(self):
        return "Name: " + self.__name + \
               "\nid: " + self.__number + \
               "\ndepartment: " + self.__department + \
               "\njob: " + self.__job
```

```
class ShiftSupervisor(Employee):
    def __init__(self, name, number, department, job, Annualpay, Paybonus):
        Employee.  init  (self, name, number, department, job)
        self.__annual_pay = Annualpay
        self.__Paybonus = Paybonus
# Set the Annual pay
    def set_Annualpay(self, Annualpay):
        self.__annual_pay = Annualpay
# Set the Paybonus
    def set_Paybonus(self, Paybonus):
        self.__Paybonus = Paybonus
# Get the Annual pay
    def get_Annualpay(self):
        return self.__annual_pay
# Get the Paybonus
    def get_Paybonus(self):
        return self.__Paybonus
# Return string to represent the object
    def __str__(self):
        return "shift: " + self.__annual_pay + \
               "\nhpr: " + self.__Paybonus



def main():
# Main function
    mycontacts = load_employee()

    choice = 0


    while choice != QUIT:
# Get the user's menu choice
        choice = get_menu_choice()

        if choice == LOOK_UP:
            look_up(mycontacts)
        elif choice == ADD:
            add(mycontacts)
        elif choice == CHANGE:
            change(mycontacts)
        elif choice == DELETE:
            delete(mycontacts)

# Save the mycontacts dictionary to a file
    save_contacts(mycontacts)


def load_employee():
    try:
        input_file = open(FILENAME, 'rb')

        employees_dct = pickle.load(input_file)

        input_file.close()
    except IOError:
# If could not open the file, then create blank dictionary
        employees_dct = {}

    return employees_dct
```

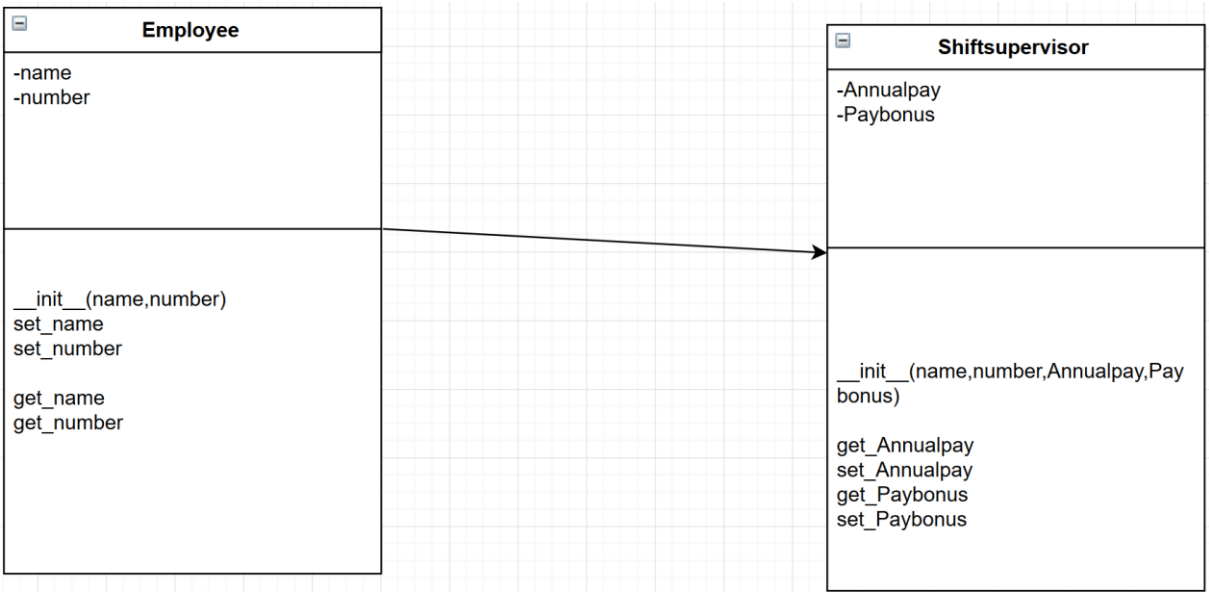
```
def get_menu_choice():
    # The get menu choice function displays the menu
    print()
    print('Menu')
    print('---------------------------')
    print('1. Look up a contact')
    print('2. Add a new contact')
    print('3. Change an existing contact')
    print('4. Delete a contact')
    print('5. Quit the program')
    print()

    choice = float(input('Enter your choice: '))

    while choice < LOOK_UP or choice > QUIT:
        choice = int(input('Enter a valid choice: '))

    return choice

def look_up(mycontacts):

    name = input('Enter a name: ')

    print(mycontacts.get(name, 'That name is not found.'))


def add(mycontacts):
# The add function adds for the new employe
    name = input('Name: ')
    number = input('Id number: ')
    department = input('Department: ')
    job = input('Job title: ')
    Annualpay = input('The annual salary: ')
    Paybonus = input('Production bonus: ')


# Create a Contact object named entry
    entry = ShiftSupervisor(name, number, department, job, Annualpay ,
Paybonus)

    if name not in mycontacts:
        mycontacts[name] = entry
        print('The entry has been added.')
    else:
        print('That name already exists.')


def change(mycontacts):
# The change function changes an existing employee
    name = input('Enter a name: ')

    if name in mycontacts:
# Update the information on an employee
        number = input('Enter the new id number: ')
        department = input('Enter the new department: ')
        job = input('Enter the new job title: ')
        Annualpay = input('Enter the new Annual salary: ')
        Paybonus = input('Enter the new Production bonus: ')
```

```
# Create a contact object named entry
        entry = ShiftSupervisor(number, department, job, Annualpay,
Paybonus,'')

        mycontacts[name] = entry
        print('Information updated.')
    else:
        print('That name is not found.')


def delete(mycontacts):

    name = input('Enter a name: ')

    if name in mycontacts:
        del mycontacts[name]
        print('Entry deleted.')
    else:
        print('That name is not found.')


def save_contacts(mycontacts):

    output_file = open(FILENAME, 'wb')

    pickle.dump(mycontacts, output_file)

    output_file.close()

# Call the main function.
main()
```

Program Output:

```
Enter your choice: 2
Name: Milan
Id number: 242526
Department: IT
Job title: Network Specialist
The annual salary: 90
Production bonus: 50
The entry has been added.


Enter your choice: 1
Enter a name: Milan
shift: 90
hpr: 50
```

UML Diagram:



Person and Customer Classes

"Write a class named Person with data attributes for a person's name, address, and telephone number. Next, write a class named Customer that is a subclass of the Person class. The Customer class should have a data attribute for a customer number and a Boolean data attribute indicating whether the customer wishes to be on a mailing list. Demonstrate an instance of the Customer class in a simple program."

Program Display:

```
import pickle

LOOK_UP = 1
ADD = 2
CHANGE = 3
DELETE = 4
QUIT = 5

FILENAME = 'percustomers.dat'


class Person:
    def __init__(self, name, address, telephone):
        self.__name = name
        self.__address = address
        self.__telephone = telephone

    def set_name(self, name):
        self.__name = name

    def set_address(self, address):
        self.__address = address

    def set_telephone(self, telephone):
```

```
        self.__telephone = telephone

    def get_name(self):
        return self.__name

    def get_address(self):
        return self.__address

    def get_telephone(self):
        return self.__telephone

    def __str__(self):
        return "\nAddress: " + self.__name + \
               "\nTelephone: " + self.__address + \
               "\nMail: " + self.__telephone


class Customer(Person):
    def __init__(self, name, address, telephone, mail):
        Person.__init__(self, name, address, telephone)
        self.__name = name
        self.__address = address
        self.__telephone = telephone
        self.__mail = mail

    def set_mail(self, mail):
        self.__mail = mail

    def get_mail(self):
        return self.__mail


# Main function
def main():
    mycontacts = load_customer()

    choice = 0

    # Get the user's menu choice.
    while choice != QUIT:

        choice = get_menu_choice()

        if choice == LOOK_UP:
            look_up(mycontacts)
        elif choice == ADD:
            add(mycontacts)
        elif choice == CHANGE:
            change(mycontacts)
        elif choice == DELETE:
            delete(mycontacts)

            # Save the mycontacts dictionary to a file.
    save_contacts(mycontacts)


def load_customer():
    try:
        # Open the contacts.dat file.
        input_file = open(FILENAME, 'rb')
```

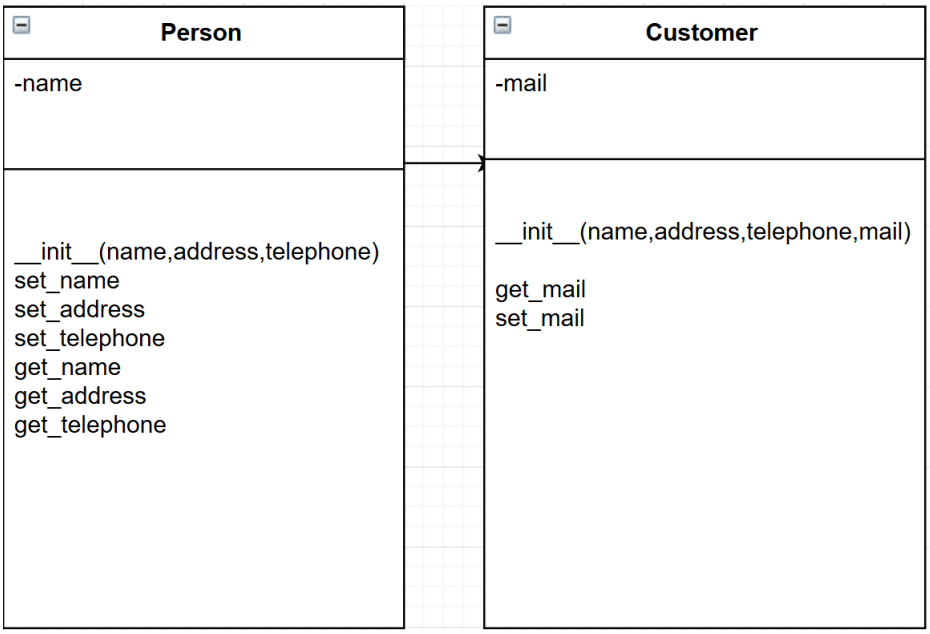
```
        customers_dct = pickle.load(input_file)

        input_file.close()
    except IOError:
        # If could not open the file, then create blank dictionary
        customers_dct = {}

    return customers_dct


def get_menu_choice():
    # The get menu choice function displays the menu
    print()
    print('Menu')
    print('---------------------------')
    print('1. Look up a customer')
    print('2. Add a new customer')
    print('3. Change an existing customer')
    print('4. Delete a customer')
    print('5. Quit the program')
    print()

    choice = float(input('Enter your choice: '))

    while choice < LOOK_UP or choice > QUIT:
        choice = int(input('Enter a valid choice: '))

    return choice


# The look_up function looks up an item in the
def look_up(mycontacts):
    name = input('Enter a name: ')

    print(mycontacts.get(name, 'That name is not found.'))


def add(mycontacts):
    # The add function adds a new employee
    name = input('Name: ')
    address = input('Address: ')
    telephone = input('Telephone number: ')
    mail = input('What is the email?(blank = not interrested): ')

    # Create a Contact object named entry.
    entry = Customer(name, address, telephone, mail)

    if name not in mycontacts:
        mycontacts[name] = entry
        print('The entry has been added.')
    else:
        print('That name already exists.')


def change(mycontacts):
    # The change function changes an existing employee
    name = input('Enter a name: ')

    if name in mycontacts:
        # Update the information on an employee
        address = input('Address: ')
```

```
        telephone = input('Telephone number: ')
        mail = input('What is the new email?(blank = not interrested): ')

        # Create a contact object named entry.
        entry = Customer(name, address, telephone, mail)

        mycontacts[name] = entry
        print('Information updated.')
    else:
        print('That name is not found.')


def delete(mycontacts):
    name = input('Enter a name: ')

    if name in mycontacts:
        del mycontacts[name]
        print('Entry deleted.')
    else:
        print('That name is not found.')


def save_contacts(mycontacts):
    output_file = open(FILENAME, 'wb')

    pickle.dump(mycontacts, output_file)

    output_file.close()


# Call the main function.
main()
```

Output:

```
Name: Milan
Address: Ryttergade
Telephone number: 5224153
What is the email?(blank = not interrested):
The entry has been added.
```

UML:



Conclusion

To be honest this chapter was a bit harder to understand than others. I communicated with other classmates and everything was clear afterwards. I already started to get into chapter 14 with “GUI”s and look forward to practice with them.