Notes:
Since some functionality of the product was realised without using OOP, python module files are displayed as classes with rounded corners

The project entry point is the orange-highlighted file "starter.py" which can be called with .csv files as arguments.

«AdministratorAbstract»
File: administratorAbstract.py

stdErr(string)
stdOut(string)

defines interfact to be implemented by all Administrator classes

cmd.CMD

IOHelper.py

initialize(target::AdministratorAbstract)
stdErr(message::string)
stdOut(message::string)

entry point to the project;
reads system arguments

stores
forwards messages

initializes

charts.py

display_statistic(statistic::{}, parameter::string, group::strin

displays supplied statistics in charts

uses to display statistics

starter.py

start()

entry point to the project;
reads system arguments

starts
administrator.start()

AdministratorCMD
File: administratorCMD.py

_print_welcome()
do_exit(line::string)
do_datasource(line::string)
do_list(line::string)
do_add_employee(line::string)
do_read_csv_file(line::string)
do_update_employee(line::string)
do_delete_employee(line::string)
do_get_info(line::string)
do_get_statistic(line::string)
cmdloop(args::string[])

offers command line interface to user to access all implemented features of the program

uses for data IO

dataHandlerFacade.py

get_all_employees(source::string)
get_employee(emp_id::string, source::string)
employee_exists(emp_id::string, source::string)
save_employees(employees::Employee[], source::string)
save_employee(employee::Employee, source::string)
update_employee(employee::Employee, source::string)
delete_employees(employee_ids::string[], source::string)
get_statistic(statistic::string, group::string, source::string)

acts as a facade to several implementations of DataHandlerAbstract interface

uses to validate input for employee creation/ updating

inputConverter.py

convert_input(input::string, input_type::string):var
convert_employee_id(emp_id::string):string
convert_gender(gender::string):string
convert_age(age::string):int
convert_bmi(bmi::string):string
convert_sales(sales::string):int
convert_salary(salary::string):int
convert_birthday(birthday::string):datetime.date

converts user input/ data read from data handlers to the appropriate type.
uses inputValidator to validate input

abc.ABCMeta

metaclass

offers interface to all implementations

«DataHandlerAbstract»
File: dataHandlerAbstract.py

get_all_employees():Employee[]
save_employees(employees::Employee[])
update_employee(employee::Employee)
delete_employees(emp_ids::string[])
get_statistic(statistic::string, group::string):{}
_get_statistic_default(statistic::string, group::string):{}
save_employee(employee::Employee)
get_employe(emp_id::string):Employee
employee_exists(emp_id::string):boolean
delete_employee(employee:Employee)

defines interface to be implemented by all DataHandler classes

uses to validate data before converting

inputValidator.py

validate_input(input::string, input_type::string):boolean
validate_input_employee_id(empid::string):boolean
validate_input_gender(gender::string):boolean
validate_input_age(age::string):boolean
validate_input_bmi(bmi::string):boolean
validate_input_sales(sales::string):boolean
validate_input_salary(salary::string):boolean
validate_input_birthday(bday::string):boolean

validates user input based on defined rules

creates and displays

Person
File: employee.py

- gender::string
- bmi::string
- birthday::datetime.date
- age::int

get_birthday_string():string

- base class to describe a person

Employee
File: employee.py

- employee_id::string
- sales::int
- salary:: int

equals(other::Employee):boolean
get_csv_line():string

- describes an employee

offers access to

DataHandlerFile
File: dataHandlerFile.py

get_all_employees():Employee[]
save_employees(employees::Employee[])
update_employee(employee::Employee)
delete_employees(emp_ids::string[])

IO to save/ read employees to/ from .csv files

DataHandlerDatabase
File: dataHandlerDatbase.py

get_all_employees():Employee[]
save_employees(employees::Employee[])
update_employee(employee::Employee)
delete_employees(emp_ids::string[])

IO to save/ read employees to/ from mySQL databases

DataHandlerSerial
File: dataHandlerSerial.py

get_all_employees():Employee[]
save_employees(employees::Employee[])
update_employee(employee::Employee)
delete_employees(emp_ids::string[])

IO to save/ read employees to/ from pickle binary files

use to validate gathered data