CS 162P
Programming Lab 6

For this exercise you will be creating a set of people that are employees with jobs.

Program Description

This problem is designed to provide practice at using inheritance, composition, and polymorphism.

For this exericise you will define two distinct classes and three that inherit from one of those two. You need to write your own main program as described below. It is suggested that you test Person, then the derived classes. Note that you cannot create an instance of Employee since it is an abstract class.

Program Requirements

For this, as in all assignments, you are expected to follow the course programming style guidelines and to properly implement both header and source files for your functions. All input should be fully validated.

You should have two class module files, one for the Person class and its methods and another for the Employee class, its methods, and all its descendant classes with their methods. Note that the Employee module must import the Person module.

You should also have a separate helper module for getInteger since it is a utility function called both in main to get the number of Jobs and in createJob to get an age. There may be other functions that should be in your helper module.

Person Class

The Person class contains three private variables, firstName, lastName and age. Its __init__ has default values to set both names to "" and age to 0.

All class variables should be private and have public setters and getters.

Employee Class

The Employee class is an abstract class. It ***includes an instance of a Person***, which is an example of class composition. Inorder to implement Employee, you will need to import the ABC module as noted in the Moodle documents.

- __init__ – parameters are firstName, lastName, and age. Create a Person using those values.
- getEmployeeName – returns a single string; firstName, space, lastName using the Person class methods
- getEmployeeAge – returns the age using the Person class methods
- getJobName – abstract method
- getJobDescription – abstract method

Job Classes

The three different Job classes all inherit from Employee. Each has an overloaded __init__ with parameters firstName, lastName, and age that is used to initialize Person. They each override the getJobName and getJobDescription methods by returning the appropriate strings.

- Class Janitor – “Sanitation Engineer”
- Class Cashier – “Customer Service Specialist”
- Class Stocker – “Merchandise Distribution Associate”

Helper Functions

- getInteger
    - used to get an input value
    - input parameters are min and max value
    - get and validates an integer between min and max
    - return the value

- createEmployee
    - used to dynamically create a new Job class
    - get first and last name and age, age validated as between 5 and 90.
    - get Job class, validated to be in one of the above classes
    - create and return an employee using this data

- displayEmployees
    - Create a CSV document of the employees
    - for each Employee display the full name, age, job, and job description
    - no return values

Main functionality

Your main program should ask the user how many Employees work at the company. It should use getInteger to enter a value between 5 and 10 and have a loop that calls createEmployee that many times to create and fill a list with the returned Employee objects.

Finally, it should call displayEmployees to export the employee information.

Example Output

```
sue jones aged 25 working as a Cashier is a Customer Service Specialist
bob smith aged 30 working as a Janitor is a Sanitation Engineer
mary smith aged 35 working as a Stocker is a Merchandise Distribution Associate
joe jones aged 27 working as a Cashier is a Customer Service Specialist
barb Employee aged 32 working as a Stocker is a Merchandise Distribution Associate
fred flintstone aged 45 working as a Janitor is a Sanitation Engineer
martha flintstone aged 46 working as a Cashier is a Customer Service Specialist
```