In this project (You must think out of the box; you must be able to decide which access modifier you should use with every data member).

Construct the following 11 classes with their attributes and behaviors as the following description:

1. ***StaffMemebr***
   - ➔ Attributes: First Name (**String**), Last Name (**String**), Date of Birth (**String**), Address (**object** of **address**). Mobile Number (**String**), personal email (**String**). Work email (**String**)
   - ➔ Methods: ***setter and getters for all attributes, toString ().*** Test all attributes with methods described underneath (*parts form a to g*) to ensure that all attributes have correct inputs from the user. Otherwise, the method should print the error message.

   I, e:

   a. Names: Should accept names as characters only. Ahmad is accepted, but ahamd1 is not.
   b. Date birth: Date 30/2/2023 is not accepted, negative numbers are not accepted, and the future date is not accepted. The staff member should be at least 16 years old.
   c. The address should be valid (check the address class underneath).
   d. Mobile No: should contain only numbers and must contain only (059 and 056) and the length is 9 characters.
   e. Personal Emails: The username must start with a character. It should contain only alphabetic (letters A-Z or a-z) followed by numbers (Special characters like white space, $,%,#,..etc., are not allowed). Then, follow with '@' and the server's name and domain.
      Sample: ahamd12@gmail.com is valid, but 1ahamd@gmail.com, or ahamd!@gmail.com is not valid.
   f. Work emails should be created automatically by a private method called *workEmailAuto*(), which will create an email by taking the first character of the first name (lowercase) followed by the last name (lowercase).

   - ➔ Pay method: this method should be public and return double values. (You can leave it empty for this phase of the project (return 0).
   - ➔ toString () should return all instances' values.
   - ➔ One Constructor for all attributes and another one without argument.

2. ***Volunteer***
   - ➔ *Attributes: association name (String)*
   - ➔ One Constructor for all attributes and another one without argument.

➔ *Methods: pay (): return double* (you can leave it empty for this project phase, return 0).

➔ Setters and getters, and toString ().

3. ***Employee:***

➔ *Attributes: empno (String), department (Department)*

Methods:

➔ *: pay (): return double* (you can leave it empty for this project phase.
Setters and getters, and toString ().

➔ One Constructor for all attributes and another one without argument.

4. ***Address:*** *Attributes: State (String), City (String), and Street (String)*

Methods:

➔ Setters and getters, and toString ().

➔ One Constructor for all attributes and another one without argument.

➔ Methods to test if all strings are valid (special characters are not allowed), and an empty string is not allowed.

5. ***HourlyEmployee:***

➔ *Attributes: hours (short), rate (float).*

Methods:

➔ Setters and getters, and toString ().

➔ Only one Constructor for all attributes.

➔ Methods to test if the rate is valid (more than 288 hours per month and less than 1 are not allowed), and a rate of more than 30 NIS and less than 8 is not allowed.

➔ Pay (): return the monthly salary (rate* hours). If the number of hours is more than 120 hours, then the employee will get 50% as extra for every hour.

➔ One Constructor for all attributes and another one without argument.

6. ***weekly employee***

➔ *Attributes: noOfweeks (byte), wage per week (float).*

➔ *Method:* Setters and getters, and toString ().

➔ One Constructor for all attributes and another one without argument.

- ➔ Methods to test if noOfweek is valid (more than 4 weeks and less than 1 are not allowed*) and the wage per week* of more than 700 NIS and less than 300 NIS is not allowed.
- ➔ Pay (): return the monthly salary (*noOfweeks * wage*).

7. Salaried Employee

- ➔ *Attributes: annual salary(double)*

Methods:

- ➔ Setters and getters, and toString ().
- ➔ One Constructor for all attributes and another one without argument.
- ➔ Methods to test if *annual salary* is valid (more than 120,000 NIS per year and less than 21600 NIS are not allowed*).*
- ➔ Pay (): return the monthly salary (*annual salary/12*).

8. Contract Employee

- ➔ *Attributes: rateOfconstract(double), no of months(byte)*

Methods:

- ➔ Setters and getters, and toString ().
- ➔ One Constructor for all attributes and another one without argument.
- ➔ Methods to test if the *contract salary* is valid (more than 10,000 NIS per month is not allowed*). Be aware of negative values.*

- ➔ Pay (): return the monthly salary (*rateOfconstarct/ no of months*).

9. CommessionEmployee

- ➔ *Attributes: The item sold (double) per month. Price: double*

Methods:

- ➔ Setters and getters, and toString ().
- ➔ One Constructor for all attributes and another one without argument.*Be*
- ➔ Pay (): return the monthly salary (*if the income from sold items is between 15,000 NIS and 20,000 NIS, then the percentage is 3%, more than 20,000, then the percentage is 5%, and less than that is 1.5%). aware of negative values.*

10. EmployeeBasedComession

I*n addition to the commission for paid items. The employee-based salary has a basic salary.*

- ➔ *Attributes: based on salary*

  Methods:

- ➔ Setters and getters, and toString ().
- ➔ One Constructor for all attributes and another one without argument.

- Pay (): return the monthly salary (*based on salary + commission). Be aware of negative values.*
-