4. **Inheritance, Abstract Classes and Interfaces**

(a) You are working for a company developing an employment database application. The database will store information about each employee (such as their name, date of birth, contact information, etc.) and will also provide access management for the company's premises.

i. Design an abstract `Employee` class to store the following attributes: forename, surname, date of birth and annual salary. [3]

ii. Add methods to your `Employee` class to modify the attributes and additionally add an abstract method to add overtime to an employee's monthly wage. [3]

iii. Programmers earn overtime at $\frac{1}{1500}$th of their annual salary per hour. Design a concrete `Programmer` class with an additional variable to hold monthly overtime payments (you may assume that this counter is reset automatically). [3]

iv. A salesperson earns overtime at $\frac{1}{1000}$th of their annual salary per hour. Additionally they are contactable via a personal office phone number. Design a concrete `Salesperson` class that holds this additional information and includes a method to retrieve their external and internal contact numbers (where the internal number is the final 5 digits of their 11 digit external phone number). [4]

(b) The access management in the building is controlled by querying an `Employee` object. There are two levels of access in the building: BuildingAccess and SecureAccess. Employees with BuildingAccess should be expected to have a method to retrieve the employee ID; employees with SecureAccess should have a method for verifying a PIN code.

i. Design two interfaces (`BuildingAccess` and `SecureAccess`) that can be used to provide this access management. [2]

ii. Within the company, programmers can access the building and secure resources; salespeople are granted building access only. Show how the class declarations provided previously would be amended to account for this. [2]

iii. Finally, given the following method:

```
public boolean AccessBuilding(Employee e) {
    // does employee have required level of access
        addLog(((BuildingAccess) e).getEmployeeID());
        return true;
    // else
        return false;
}
```

Demonstrate how this method would check if the provided `Employee` has the required level of access for the building. Additionally, write the method `AccessSecure` to check access to secure resources. [3]