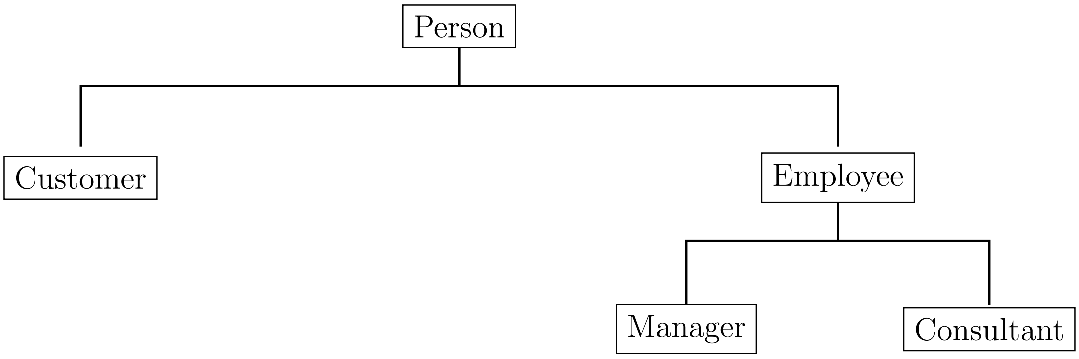
1 Part: Person class hierarchy 50%

In this exercise you will implement a simple hierarchy of people using inheritance. The hierarchy consists of the base class `Person`, the classes `Customer` and `Employee`, which are derived from `Person`, and two other classes, `Manager` and `Consultant`, which are derived from the class `Employee`. The hierarchy is shown in the figure below.



Each person supports two functions: `toString` and `type`. The function `type` should return the name of that class. The `toString` function should return a string with the information of the instance in the format illustrated in the examples below.

- For a `Customer` with name John Smith, customer ID 42 and no orders:

  ```
  [John Smith, 42]
  ```

- For a `Customer` with name John Smith, customer ID 42 and orders with numbers 123 and 456:

  ```
  [John Smith, 42, 123, 456]
  ```

- For a `Manager` with name Alice Jones and salary 50000.5:

  ```
  [Alice Jones, earns:  50000.5]
  ```

- For a `Consultant` Bob Lewis with salary 30000 and manager Alice Jones:

  ```
  [Bob Lewis, earns:  30000, manager:  Alice Jones]
  ```

- For a `Consultant` Bob Lewis with salary 30000 and no manager:

  ```
  [Bob Lewis, earns:  30000]
  ```

`Customer` has an additional feature to place an order. Every employee has the additional feature to get their salary increased. `Consultant` has a manager, which is implemented by a pointer to and object of type `Manager`. Notice that many consultants will share the pointer to the same `Manager`, and that the `Consultant` objects do not own the corresponding `Manager` object. Also, some of the classes have some extra functions.

Your task is to add the necessary member variables and function implementations. You should add the keyword `virtual` where appropriate, and then implement the functions where they are most appropriate. `Person` and `Employee` could be abstract classes, but `Customer`, `Manager` and `Consultant` should be concrete classes. The skeletons of the classes are less complete than in previous assignment, as you now have to choose where to implement each function, where add the `virtual` keyword, etc.

A detailed description of the methods of the classes is given in `Person.h`.