Introduction

This task is about representation of a bank with personal and clients using inheritance and polymorphism. Basic representation contains one boss, several managers, several staff and some customers.

You are provided with six header files, and one main file to test your functions, which is seven files in total. All explanations will be provided in this .pdf file as well as example output at the bottom. Please, make sure to follow the explanations and check if every case of a single function is considered.

Let us begin!

Person.h

Person.h file contains one enum classes, which represents privileges of a specific person, e.g Manager has privileges Manager. Moreover, there is one usual class, which represent a person. Class Person contains: privileges from enum class Privileges, name and one static integer, which is used to keep track of amount of people.

Task 0 (the beginning)

Please, implement every function in Person.h:

Two different constructors:

1. Person() – default constructor, with default values: priviliges = NONE, name = NO NAME
2. Person(name) – same as default constructor, but now name should be assigned with user input

One copy constructor. While implementing constructors, do not forget to increase amount of people by one.

One destructor, which is virtual. It should decrease amount of people by one, and reset name and privileges.

Additionally, there are four pure virtual functions.

Operator << should be overloaded for printing. Here, operator << should use inherited method info.

Customer.h

Customer.h is inherited from Person.h.

Customer header file contains balance of each customer, which is loan taken from bank, number of days that left to pay back this money to bank, and amount of customers.

Implement four different constructors, two of which is copy constructor:

1. Customer() – default constructor. Hint: use inherited constructors.

2. Customer(name) – same as default constructor, but with name, as user input.
3. Customer(Customer) – copy constructor. Hint: you may use, previously implemented functions.
4. Customer(Customer, cash, day) – copy constructor with cash, which is loan taken from bank, and day, which is number of days for pay back, as user input. We will need this constructor for future implementations in Staff.h

Destructor – should assign everything to default values and decrease number of customers by one.

Moreover, override info method that is used in operator <<

There are also two clone functions. We have implemented similar clone functions during lectures. However, clone function with user input was not discussed during lectures.

> Clone(cash, day) – returns new Customer pointer with user input instead of default values.

Boolean operator== that is used for comparison. Returns true if names of two customers are same.

Additionally, there is general info method. This method will be useful in task 9:

> General_info() – prints name. Idea behind general info – to print each person as a tree of privileges, where boss is at the left and customers are at the right. Example is at the bottom of the file

Method say() - prints “Customer: name”, where name is name of a customer. This is used as an idea for RTTI mechanism (dynamic_cast)

Staff.h

Staff.h file contains dynamically allocated memory of customers, number of customers in a memory, maximum number of customers in a memory, number of staff and total money, which is loan taken from customers.

There are three constructors:

1. Staff() – default constructor. Hint: use inherited constructor. Additionally, please use new for memory allocation in constructors and assign everything to null pointer. Do not forget to change privileges
2. Staff(name) – same as default constructor, but with name as user input.
3. Staff(Staff) – copy constructor. Hint: use previously implemented constructors. Use clone function, to avoid shallow copy.

Destructor: Please, delete assigned memory to avoid memory leak. Moreover, assign everything to default values and do not forget to decrease number of staff.

Implement clone function: functionality is same as from Customer.h.

Overload operator +=: used to append dynamically allocated memory for customers with user input. Please make sure that maximum number of customers is not reached, throw an exception if reached.

Override methods info and general_info:

1. General_info() – prints name with end of line and general info of customers.
2. Info() – prints name and privilege, and prints info of customers.

Method work(customer) – finds specific customer in an array of customers and adds 500 to a balance of a customer and sets day to 50. Does nothing if customer was not found.

Method say() – same as in Customer.h, but instead of Customer prints Staff

Manager.h

Manager.h file contains dynamically allocated memory of staff, number of staff in a memory, maximum number of staff in a memory and number of managers.

There are three constructors. Idea is same as in Staff.h

Destructor – deletes allocated memory and assigns everything to default values. Do not forget to decrease number of mangers.

Clone method – same as in Staff.h

Operator += appends allocated memory with staff. Please make sure that maximum number of staff is not reached, throw an exception if reached.

Override method info and general_info:

1. General_info() – prints name with end of line and general info of staff.
2. Info() - prints name and privilege, and prints info of staff.

Method work(customer) – searches customer in an array of staff. Hint: recursively use work method for staff.

Override method say() - same as in Customer.h, but instead of Customer prints Manager

Boss.h

Boss.h file contains dynamically allocated memory of managers, number of managers in a memory, maximum number of managers in a memory and number of bosses, which should not exceed one.

There are three constructors. Idea is same as in Staff.h. Yet, do not forget to check if number of bosses is not greater than one. Throw exception if exceeded.

Destructor – deletes allocated memory and assigns everything to default values. Do not forget to assign number of bosses to zero.

Clone method – same as in Staff.h

Operator += appends allocated memory with manager. Please make sure that maximum number of managers is not reached, throw an exception if reached.

Override method info and general_info:

1. General_info() – prints name with end of line and general info of managers.
2. Info() - prints name and privilege, and prints info of managers.

Method work(customer) – searches customer in an array of managers. Hint: recursively use work method for managers.

Override method say() - same as in Customer.h, but instead of Customer prints Boss.

Database.h

Database.h file contains dynamically allocated memory of Person, number of people in a memory and maximum number of people in a memory.

There is one constructor and one destructor:

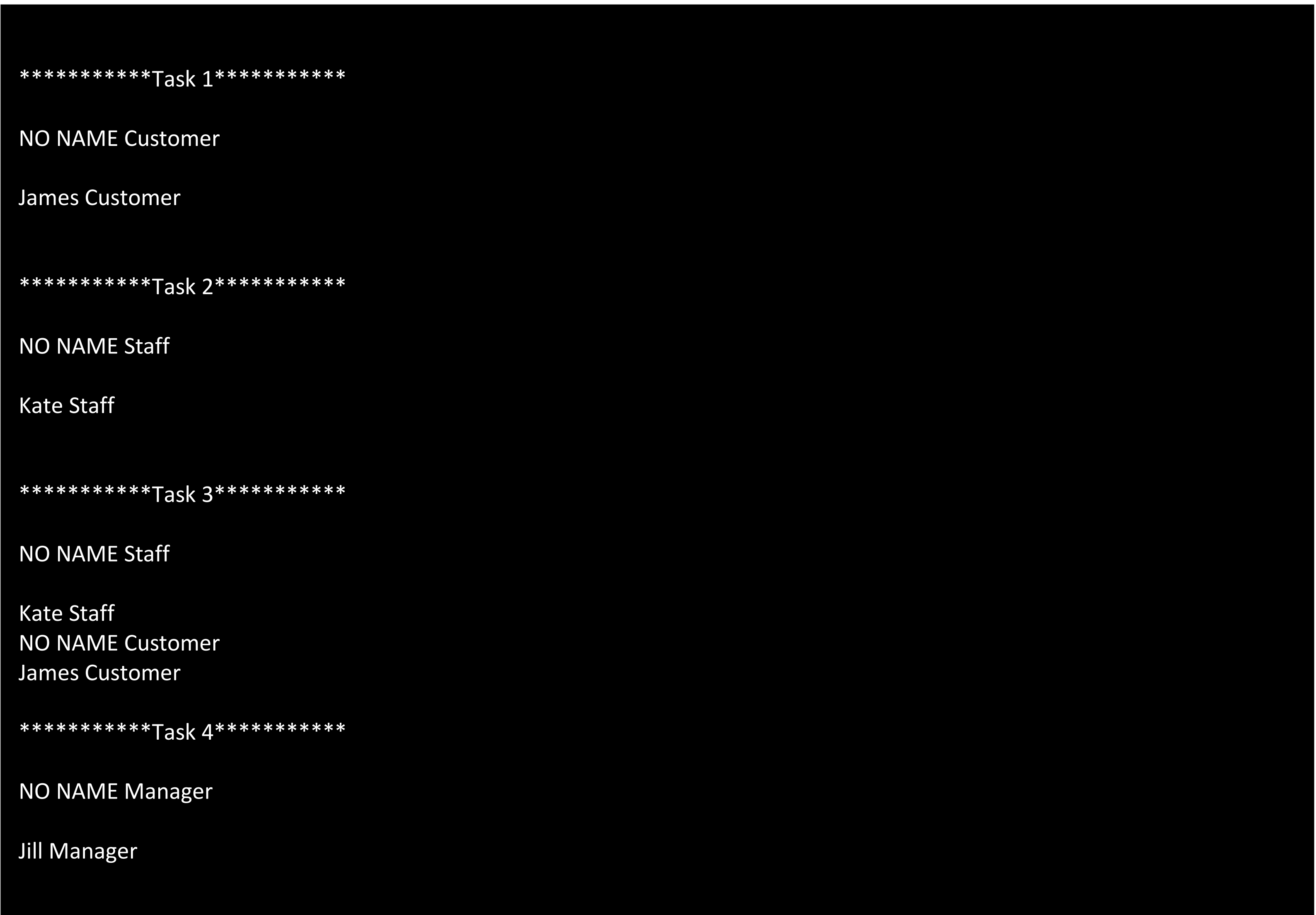
1. Constructor Database() – use new for memory allocation in constructor and assign everything to null pointer.
2. Destructor ~Database() – deletes allocated memory and assigns number of people to 0.

Implement output operator << - prints "Database is empty" if number of people in array of people is zero, otherwise, prints people in array (Hint: use info method implemented before)

Add(person) – appends array of people with user input. Make sure that maximum number of people in an array is not exceeded. Throw exception if exceeded.

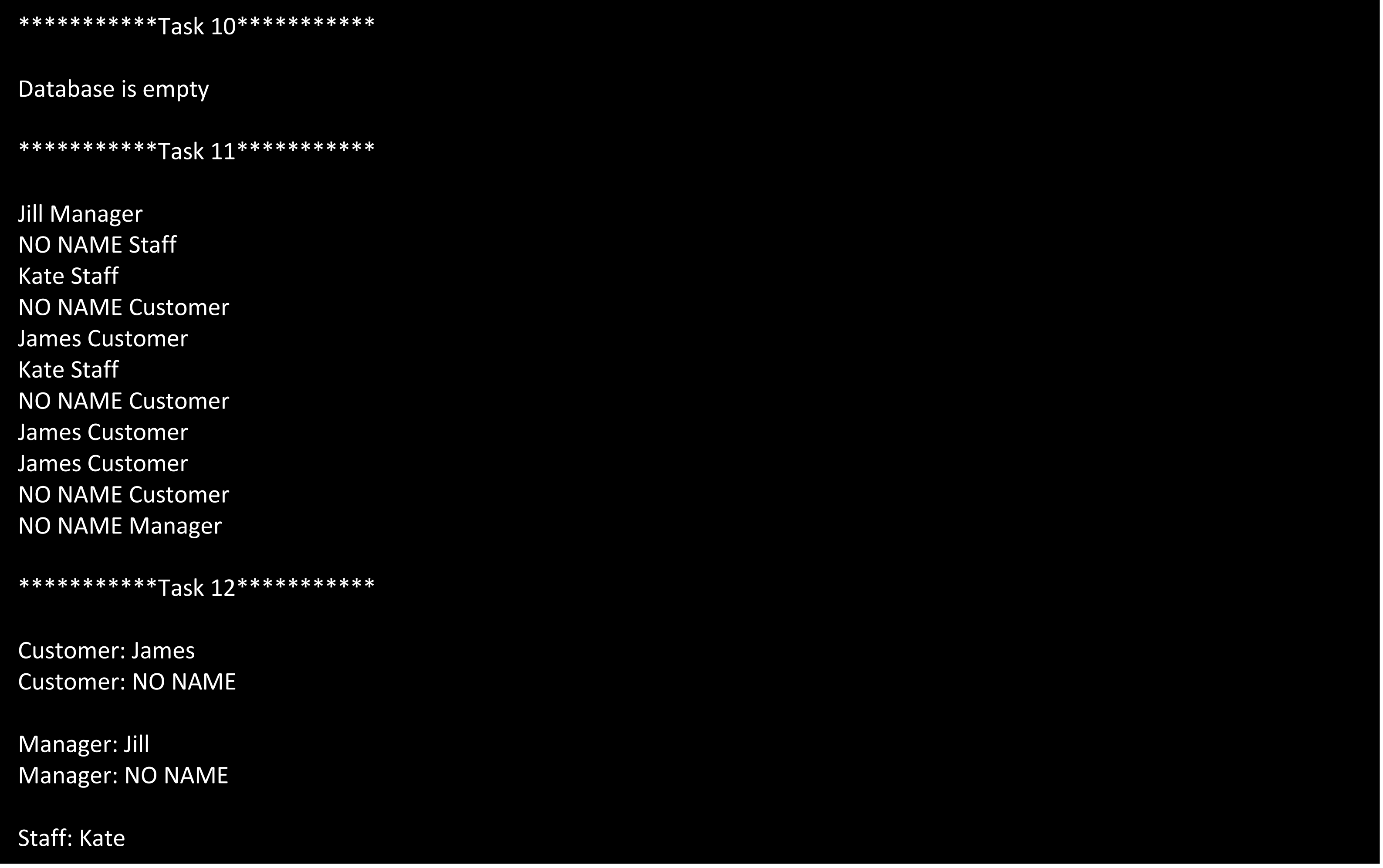
Say methods – use RTTI mechanism and use say method of a specific person.

Example output



```
***********Task 1***********

NO NAME Customer

James Customer


***********Task 2***********

NO NAME Staff

Kate Staff


***********Task 3***********

NO NAME Staff

Kate Staff
NO NAME Customer
James Customer

***********Task 4***********

NO NAME Manager

Jill Manager
```

```
***********Task 5***********

NO NAME Manager

Jill Manager
NO NAME Staff
Kate Staff
NO NAME Customer
James Customer


***********Task 6***********

Maximum limit of bosses is reached
Bill Boss

NO NAME Boss


***********Task 7***********

Bill Boss
NO NAME Manager
Jill Manager
NO NAME Staff
Kate Staff
NO NAME Customer
James Customer
NO NAME Manager
NO NAME Manager

***********Task 8***********

Staff: Kate gives 500$ for 50 days to: James Customer

***********Task 9***********

Boss
Bill
        Managers:
        1.NO NAME
                Staff:

        2.Jill
                Staff:
                1.NO NAME
                        Customers:

                2.Kate
                        Customers:
                        1.NO NAME
                        2.James


        3.NO NAME
                Staff:

        4.NO NAME
                Staff:
```



```
***********Task 10***********

Database is empty

***********Task 11***********

Jill Manager
NO NAME Staff
Kate Staff
NO NAME Customer
James Customer
Kate Staff
NO NAME Customer
James Customer
James Customer
NO NAME Customer
NO NAME Manager

***********Task 12***********

Customer: James
Customer: NO NAME

Manager: Jill
Manager: NO NAME

Staff: Kate
```

Good luck :)