HR Management System App in C++

ezz

May 2023

1 Introduction

The HR Management System App is a software application designed to automate the management of employee records, benefits, and payroll. The application is modular, with several classes such as Employee, Benefits, Payroll, and Department, each representing a distinct area of functionality. The main objective of the HR system is to provide an efficient and user-friendly way for HR personnel to manage employee data, benefits, and compensation.

2 Design

2.1 Employee Class

The Employee class stores the basic information of an employee, including their name, ID, job title, phone, email, and a list of benefits. There are four types of employees: Hourly, Salaried, Commission, and Manager. Each type of employee has different attributes and a new equation for the payroll function.

2.2 Department Class

The Department class stores information about a department, including the department name and ID. Each employee should belong to a department.

2.3 Benefits Class

The Benefits class is a base class used to determine different types of benefits for employees. There are mainly two types of benefits, and the calculation equation for each type is shown in the following table.

2.4 HRSystem Class

The HRSystem class is the main class that stores a list of employees and manages the operations of adding, removing. employees. It should also have a method to display the information of a specific employee or the average salary of the system.

3 Implementation

In this section, we will discuss the specific coding details of the Employee, Benefits, Department, and HRSystem classes.

3.1 Employee Class

The Employee class stores the basic information of an employee, including their name, ID, job title, phone, email, and a list of benefits. There are four types of employees: Hourly, Salaried, Commission, and Manager. Each type of employee has different attributes and a new equation for the payroll function.

3.1.1 Code

Here is an example of the implementation of the Employee class in C++:

```
class employee {
public:
  employee();
  virtual ~employee();
  void setId(int );
  virtual string getDetails();
  virtual void setDetails(int);
  virtual double getSalary() = 0; // pure virtual function
  bool addBenefit();
  void editBenefit(int);
  void deleteBenefit(int);
  void getBenefit();
  double totalBenefit();
  void setDepartment(department*);
protected:
  unsigned int m_id;
  string m_name;
  string m_phone;
  string m_email;
  string m_jopTitle;
  department *m_department;

  int m_benefitCount;
  int m_benefitSize;
  benefit **benefitList;
};

class hourlyEmployee : public employee {
public:
  virtual string getDetails();
  virtual void setDetails(int);
```

```
  virtual double getSalary();
  void addHours(double );

private:
  double m_hourWorked;
  double m_rate;
};

// code for the other employee types (Salaried, Commission, and Manager) is similar
```

3.1.2 Explanation

The Employee class is an abstract class that serves as the base class for the Hourly, Salaried, Commission, and Manager classes. It has protected member variables for the employee's name, ID, job title, phone number, email, and a dynamic array of Benefits objects.

The Hourly class is a derived class that represents employees who are paid on an hourly basis. It has private member variables for the employee's hourly wage and the number of hours worked. It also has a constructor that calls the base class constructor and initializes the member variables. The Hourly class implements the virtual functions of the base class, including the getSalary() function, which calculates the employee's pay, and the getDetails() function, which displays the employee's details.

The other employee classes (Salaried, Commission, and Manager) are similar to the Hourly class, with slight differences in their member variables and pay calculation functions.

3.2 Department Class

The Department class stores information about a department, including the department name and ID. Each employee should belong to a department. The code for the Department class is shown below:

```
class department {
public:
  department();
  department(unsigned int, string);

  virtual ~department();
  void read();
  void print();
  string getname();
  int getId();
private:
  unsigned int m_id;
```

```
  string m_name;
};
```

The class has a private variable for the department name, ID, and a vector of Employee objects to store the employees in the department. The public methods include a constructor, methods to add to the department, and a method to display the department information.

3.3 Benefits Class

The Benefits class is a base class used to determine different types of benefits for employees. There are mainly two types of benefits.

The code for the Benefits class is shown below:

```
class benefit {
public:
  benefit();
  virtual ~benefit();
  virtual double calculateBenefit() = 0;
  virtual string displayBenefit();
  virtual void setBenefit();

protected:
  string m_info;
  string m_planType;
  double m_amount;
};
```

The class has protected variables for the health and dental benefits, and a constructor to initialize them. The class also has a pure virtual method to calculate the benefits based on the amount.

3.4 HRSystem Class

The HRSystem class is the main class that stores a list of employees and manages the operations of addin ans removing employees. It should also have a method to display the information of a specific employee or the average salary of a all the employee.

The code for the HRSystem class is shown below:

```
class hrSystem {
public:
  hrSystem(int);
  virtual ~hrSystem();
```

```
  void addEmployee();
  void editEmployee(int);
  void deleteEmployee(int);
  void printall();
  void totalpay();
  void printany(int);
  void benefitmanager(int,int ,int);
  void assiagndeparttoemp(int ,department *);

private:
  employee **staff;
  unsigned int m_staffCount;
  unsigned int m_staffSize;
};
```

The class has private variables for dynamic array of Employee objects to store the employees . The public methods include methods to add and remove employees , methods to find employees by ID, a method to display employee information, and a method to calculate the average salary of all the employees.

The methods for adding and removing employees simply add or remove them from the appropriate dynamic array. The displayEmployeeInfo method takes an employee ID and displays their information.

4 Testing

We developed a comprehensive test plan to ensure that the HR Management System App functions correctly. We tested the application using various scenarios, including adding and removing employees, calculating payroll, and displaying employee and department information.

The results of the tests were positive, and the application performed as expected. We fixed any bugs or issues that arose during testing.

5 Conclusion

In conclusion, the HR Management System App is a useful tool for automating the management of employee records, benefits, and payroll. The application is modular, easy to use, and flexible. We believe that it can be used by HR personnel in various organizations.

Future improvements could include adding more functionality to the application, such as generating reports or integrating with other HR systems. We also plan to continue testing and improving the application to ensure that it remains reliable and efficient.