REPORT #5



|  |
| --- |
| **제 출 일** : 2017년 11월 21일 |
| **과 목 명** : 컴퓨터프로그래밍및실습 |
| **담당교수 : 김정국 교수님** |
| **학 과** : 컴퓨터전자시스템 |
| **학 번** : 201703091 |
| **성 명 : 전기범** |

**HANKUK UNIVERSITY OF FOREIGN STUDIES**



**1 . 직원 클래스 / 프로그래머 클래스 / 매니저 클래스**

* 1. **문제 정의**

직원 클래스를 생성 후, 프로그래머 클래스와 매니저 클래스에 상속, 각각의 변수들 추가. 모든 클래스의 생성자를 한 번씩 호출해야 하며 파생 클래스의 출력 함수를 한 번 씩 호출해야한다. 단, 파생 클래스가 2개면 두 개의 출력 결과가 있어야 함.

**1.2. 문제 해결 방법**

직원 클래스을 생성, 멤버 변수와 설정자, 접근자, 기본 생성자를 만든 후 다른 클래스에 상속, 각 클래스에 해당되는 멤버 변수들을 protected로 생성. main문의 생성자와 출력 함수 호출을 사용하여 문제를 해결하였습니다.

**1.3. 소스 코드**

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  /\* set~ : ~설정자 get~ : ~접근자\*/  class Employee // 직원 클래스  {  protected:  int id; // 사원번호  string name; // 이름  int pay; // 급여  public:  Employee(int i = 0, string n = "미설정", int p = 0) :name(n), id(i), pay(p){} // 기본 생성자  void setID(int i);  int getID();  void setname(string n);  string getname();  void setpay(int p);  int getpay();  void print(); // 출력함수  };  void Employee::setID(int i)  {  id = i;  }  int Employee::getID()  {  cout << "사원번호 : " << id << endl;  return id;  }  void Employee::setname(string n)  {  name = n;  }  string Employee::getname()  {  cout << "이름 : " << name << endl;  return name;  }  void Employee::setpay(int p)  {  pay = p;  }  int Employee::getpay()  {  cout << "급여 : " << pay << endl;  return pay;  }  void Employee::print()  {  getID();  getname();  getpay();  }  class Programmer : public Employee // 프로그래머 클래스, 직원 클래스 상속  {  private:  string lang; // 언어  public:  Programmer(int i = 0, string n = "미설정", int p = 0, string l = 0) :Employee(i, n, p), lang(l){} // 기본 생성자  void setlang(string l);  string getlang();  void print(); // 출력함수  };  void Programmer::setlang(string l)  {  lang = l;  }  string Programmer::getlang()  {  cout << "주 사용 언어 : " << lang << endl;  return lang;  }  void Programmer::print()  {  Employee::print();  getlang();  }  class Manager : public Employee // 매니저 클래스, 직원 클래스 상속  {  private:  string part; // 부서  public:  Manager(int i = 0, string n = "미설정", int p = 0, string pt = 0) :Employee(i, n, p), part(pt){} // 기본 생성자  void setpart(string pt);  string getpart();  void print(); // 출력함수  };  void Manager::setpart(string pt)  {  part = pt;  }  string Manager::getpart()  {  cout << "부서 : " << part << endl;  return part;  }  void Manager::print()  {  Employee::print();  getpart();  }  int main()  {  Employee A;  Employee B(12345, "김병호", 5000000); // 사원번호, 이름, 급여  Programmer C(56789, "오순철", 4000000, "C언어"); // 사원번호, 이름, 급여, 언어  Manager D(65432, "박금주", 4200000, "디자인팀"); // 사원번호, 이름, 급여, 부서  /\* 생성자 \*/  A.print();  cout << "------------------------" << endl;  B.print();  cout << "------------------------" << endl;  C.print();  cout << "------------------------" << endl;  D.print();  cout << "------------------------" << endl;  /\* 출력함수 호출 \*/  } |

**1.4 결과 화면**

|  |
| --- |
|  |

**2 . 사람 클래스 / 강의자 클래스 / 학생 클래스**

**2.1 문제 정의**

사람 클래스를 생성 후, 강의자 클래스와 학생 클래스에 상속, 각각의 변수들 추가. 모든 클래스의 생성자를 한 번씩 호출해야 하며 파생 클래스의 출력 함수를 한 번 씩 호출해야한다. 단, 파생 클래스가 2개면 두 개의 출력 결과가 있어야 함.

**2.2. 문제 해결 방법**

사람 클래스을 생성, 멤버 변수와 설정자, 접근자, 기본 생성자를 만든 후 다른 클래스에 상속, 각 클래스에 해당되는 멤버 변수들을 protected로 생성. main문의 생성자와 출력 함수 호출을 사용하여 문제를 해결하였습니다.

**2.3. 소스 코드**

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  /\* set~ : ~설정자 get~ : ~접근자\*/  class Person // 사람 클래스  {  protected:  string name; // 이름  int birth; // 생년월일  public:  Person(string n = "미설정", int i = 20170000) :name(n), birth(i){} // 기본 생성자  void setname(string n);  string getname();  void setBirth(int b);  int getBirth();  void print(); // 출력함수  };  void Person::setname(string n)  {  name = n;  }  string Person::getname()  {  cout << "이름 : " << name << endl;  return name;  }  void Person::setBirth(int b)  {  birth = b;  }  int Person::getBirth()  {  cout << "생년월일 : " << birth << endl;  return birth;  }  void Person::print()  {  getname();  getBirth();  }  class Instructor : public Person // 강의자 클래스, 사람 클래스 상속  {  private:  string subject; // 강의 과목  public:  Instructor(string n = "미설정", int i = 0, string s = 0) :Person(n, i), subject(s){} // 기본 생성자  void setsubject(string s);  string getsubject();  void print(); // 출력함수  };  void Instructor::setsubject(string s)  {  subject = s;  }  string Instructor::getsubject()  {  cout << "강의 과목 : " << subject << endl;  return subject;  }  void Instructor::print()  {  Person::print();  getsubject();  }  class Student : public Person // 학생 클래스, 사람 클래스 상속  {  private:  string major; // 전공  public:  Student(string n = "미설정", int i = 0, string m = "") :Person(n, i), major(m){} // 기본 생성자  void setmajor(string pt);  string getmajor();  void print(); // 출력함수  };  void Student::setmajor(string m)  {  major = m;  }  string Student::getmajor()  {  cout << "전공 : " << major << endl;  return major;  }  void Student::print()  {  Person::print();  getmajor();  }  int main()  {  Person A;  Person B("김병호", 19981114); // 이름, 생년월일  Instructor C("오순철", 19851028, "C언어"); // 이름, 생년월일, 강의과목  Student D("박금주", 19981110, "컴퓨터공학"); // 이름, 생년월일, 전공  /\* 생성자 \*/  A.print();  cout << "------------------------" << endl;  B.print();  cout << "------------------------" << endl;  C.print();  cout << "------------------------" << endl;  D.print();  cout << "------------------------" << endl;  /\* 출력함수 호출 \*/  } |

**2.4 결과 화면**

|  |
| --- |
|  |

**3 . 계좌 클래스 / 이자 클래스 / 확인 클래스**

**3.1 문제 정의**

계좌 클래스를 생성 후, 이자 클래스와 확인 클래스에 상속, 각각의 변수들 추가. 이자 계산 함수를 통하여 현재 잔액의 이자를 알 수 있어야 하며, 게좌의 잔액을 적금 계좌로 입금하면 적금 잔액은 해당 금액만큼 올라가고, 계좌의 잔액은 줄어듦. 한달은 30일로 가정하며, 모든 클래스의 생성자를 한 번씩 호출해야 하며 파생 클래스의 출력 함수를 한 번 씩 호출해야한다. 단, 파생 클래스가 2개면 두 개의 출력 결과가 있어야 함.

**3.2. 문제 해결 방법**

계좌 클래스을 생성, 멤버 변수와 설정자, 접근자, 기본 생성자를 만든 후 다른 클래스에 상속, 각 클래스에 해당되는 멤버 변수들을 protected로 생성. main문의 생성자와 출력 함수 호출을 사용하여 문제를 해결하였습니다. 적금 클래스의 기본 생성자에서 main문의 생성자 순서 문제로 기본 생성자의 순서가 적금 계좌번호, 적금 금액, 계좌번호, 잔액 순으로 설정. 해당 순서가 아닐경우 출력결과와 일치하도록 만들었습니다.

**3.3. 소스 코드**

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  /\* set~ : ~설정자 get~ : ~접근자\*/  class Account // 계좌 클래스  {  protected:  string account\_number; // 계좌번호  int balance; // 잔액  public:  Account(string A = "620-111222-333", int b = 1000000) :account\_number(A), balance(b){} // 기본 생성자  void setAccount\_Number(string A);  string getAccount\_Number();  void setBalance(int b);  int getBalance();  void deposit(int d); // 입금 함수  void withdraw(int w); // 출금 함수  void print();// 출력 함수  };  void Account::setAccount\_Number(string A)  {  account\_number = A;  }  string Account::getAccount\_Number()  {  cout << "계좌번호 : " << account\_number << endl;  return account\_number;  }  void Account::setBalance(int b)  {  balance = b;  }  int Account::getBalance()  {  cout << "잔액 : " << balance << endl;  return balance;  }  void Account::deposit(int d)  {  balance += d;  }  void Account::withdraw(int w)  {  balance -= w;  }  void Account::print()  {  getAccount\_Number();  getBalance();  }  class Interest :public Account // 이자 클래스, 계좌 클래스 상속  {  private:  double rate; // 이자율  int rate\_money; // 이자율\*잔액  public:  Interest(string A = "620-111222-333", int b = 1000000, double r = 0) :Account(A, b), rate(r){} // 기본 생성자  void setrate(double r);  double getrate();  void setcalc(); // 이자 계산 함수  int getcalc();  void print(); // 출력함수  };  void Interest::setrate(double r)  {  rate = r;  }  double Interest::getrate()  {  cout << "이자율 : " << rate << endl;  return rate;  }  void Interest::setcalc()  {  rate\_money = (rate\*balance) / 100;  }  int Interest::getcalc()  {  cout << "이자 금액 : " << rate\_money << endl;  return rate\_money;  }  void Interest::print()  {  Account::print();  setcalc(); // 이자 계산 함수 호출, 미호출시 rate\_money값 안들어감  getrate();  getcalc();  }  class Saving :public Account // 적금 클래스, 계좌 클래스 상속  {  private:  string saving\_account\_number; // 적금 계좌번호  int saving\_balance = 0; // 적금 잔액  int saving; // 적금 금액  public:  Saving(string san = "620-111222-333", int s = 0, string A = "620-111222-333", int b = 1000000) :saving\_account\_number(san), saving\_balance(s), Account(A, b){} // 기본 생성자, 메인문 초기화로인하여 적금 계좌번호, 적금 금액 선 초기화  void setSaving\_Account\_Number(string SAN);  string getSaving\_Account\_Number();  void setSaving\_Balance(int S); // 적금 함수  int getSaving\_Balance();  void print(); // 출력함수  };  void Saving::setSaving\_Account\_Number(string SAN)  {  saving\_account\_number = SAN;  }  string Saving::getSaving\_Account\_Number()  {  cout << "적금 계좌번호 : " << saving\_account\_number << endl;  return saving\_account\_number;  }  void Saving::setSaving\_Balance(int S)  {  saving = S; // 적금 금액  balance -= S; // 잔액 -= 적금 금액  saving\_balance += S; // 적금 잔액 += 적금 금액  }  int Saving::getSaving\_Balance()  {  cout << "적금 금액 : " << saving << endl;  cout << "적금 잔액 : " << saving\_balance << endl;  return saving\_balance;  }  void Saving::print()  {  Account::print();  cout << endl;  getSaving\_Account\_Number();  getSaving\_Balance();  }  int main()  {  Account A;  Account B("620-123456-789", 1500000); // 계좌번호, 잔액  Interest C("620-123456-789", 1000000, 3); // 계좌번호, 잔액, 이자율  Saving D("620-987654-123", 0); // 계좌번호, 잔액, 적금 잔액  /\* 생성자 \*/  B.deposit(5000); // 입금 함수 호출  B.withdraw(7000); // 출금 함수 호출  D.setSaving\_Balance(200000); // 적금 함수 호출, 20만원 적금  A.print();  cout << "------------------------" << endl;  B.print();  cout << "------------------------" << endl;  C.print();  cout << "------------------------" << endl;  D.print();  cout << "------------------------" << endl;  /\* 출력함수 호출 \*/  D.setSaving\_Balance(100000); // 적금 함수 호출, 10만원 적금  D.print();  cout << "------------------------" << endl;  } |

**3.4 결과 화면**

|  |
| --- |
|  |

**4 . 근로자 클래스 / 시간 근로자 클래스 / 월급 근로자 클래스**

**4.1 문제 정의**

근로자 클래스를 생성 후, 시간 근로자 클래스와 월급 근로자 클래스에 상속, 각각의 변수들 추가. 시간 근로자는 40시간까지는 시간당 임금을 받으나, 40시간을 초과하면 시간당 1.5배의 임금을 받음. 월급 근로자는 무조건 50시간에 해당하는 임금을 받으며 한달은 30일로 가정한다. 모든 클래스의 생성자를 한 번씩 호출해야 하며 파생 클래스의 출력 함수를 한 번 씩 호출해야한다. 단, 파생 클래스가 2개면 두 개의 출력 결과가 있어야 함.

**4.2. 문제 해결 방법**

근로자 클래스을 생성, 멤버 변수와 설정자, 접근자, 기본 생성자를 만든 후 다른 클래스에 상속, 각 클래스에 해당되는 멤버 변수들을 protected로 생성. main문의 생성자와 출력 함수 호출을 사용하여 문제를 해결하였습니다. 시간 근로자 클래스의 주급 계산 함수에서 if문과 else if문을 사용하여 근무 시간이 40시간 이하일 경우 주급 = 일한 시간\*시간당 임금, 40시간 초과시 주급 = ((일한 시간 – 40)\*(시간당 임금\*1.5)) + (40 \* 시간당 임금)을 계산하도록 하였습니다. 또한 월급 = (일한 시간 \* 시간당 임금)\*(일한 개월 수 \* 30일(1달))으로 계산 하였습니다.

**4.3. 소스 코드**

|  |
| --- |
| #include <iostream>  #include <string>  using namespace std;  /\* set~ : ~설정자 get~ : ~접근자\*/  class Worker // 근로자 클래스  {  protected:  string name; // 이름  int pay; // 시간당 임금  public:  Worker(string n = "미설정", int p = 10000) :name(n), pay(p){} // 기본 생성자  void setName(string n);  string getName();  void setPay(int p);  int getPay();  void print(); // 출력 함수  };  void Worker::setName(string n)  {  name = n;  }  string Worker::getName()  {  cout << "이름 : " << name << endl;  return name;  }  void Worker::setPay(int p)  {  pay = p;  }  int Worker::getPay()  {  cout << "시간당 임금 : " << pay << endl;  return pay;  }  void Worker::print()  {  getName();  getPay();  }  class PartTimer :public Worker // 시간 근로자 클래스, 근로자 클래스 상속  {  protected:  int work\_time; // 일한 시간  int weekly\_pay; // 주급  public:  PartTimer(string n = "미설정", int p = 10000, int wt = 0) :Worker(n, p), work\_time(wt), weekly\_pay(0){} // 기본 생성자  void setWork\_Time(int wt);  int getWork\_Time();  void setWeekly\_Pay(); // 주급 계산 함수  int getWeekly\_Pay();  void print(); // 출력 함수  };  void PartTimer::setWork\_Time(int wt)  {  work\_time = wt;  }  int PartTimer::getWork\_Time()  {  cout << "일한 시간 : " << work\_time << endl;  return work\_time;  }  void PartTimer::setWeekly\_Pay()  {  if (work\_time <= 40) // 일한 시간 40시간이하 : 주급 = 일한 시간\*시간당 임금  weekly\_pay = (work\_time\*pay);  else if (work\_time > 40) // 일한 시간 40시간 초과 : 주급 = ((일한시간 - 40)\*(시간당 임금\*1.5)) + (40 \* 시간당 임금)  weekly\_pay = ((work\_time - 40)\*(pay\*1.5)) + (40 \* pay);  }  int PartTimer::getWeekly\_Pay()  {  cout << "계산된 주급 : " << weekly\_pay << endl;  return weekly\_pay;  }  void PartTimer::print()  {  Worker::print();  getWork\_Time();  setWeekly\_Pay(); // 주급 계산 함수 호출, 미 호출시 weekly\_pay값 안들어감  getWeekly\_Pay();  }  class Monthly\_Paid :public Worker // 월급 근로자 클래스, 근로자 클래스 상속  {  protected:  int work\_time; // 일한 시간  int work\_month; // 일한 개월 수  int monthly\_pay; // 월급  public:  Monthly\_Paid(string n = "미설정", int p = 10000, int wt = 0, int wm = 0) :Worker(n, p), work\_time(wt), work\_month(wm), monthly\_pay(0){} // 기본 생성자  void setWork\_Time(int wt);  int getWork\_Time();  void setWork\_Month(int wm);  int getWork\_Month();  void setMonthly\_Pay(); // 월급 계산 함수  int getMonthly\_Pay();  void print();  };  void Monthly\_Paid::setWork\_Time(int wt)  {  work\_time = wt;  }  int Monthly\_Paid::getWork\_Time()  {  cout << "일한 시간 : " << work\_time << endl;  return work\_time;  }  void Monthly\_Paid::setWork\_Month(int wm)  {  work\_month = wm;  }  int Monthly\_Paid::getWork\_Month()  {  cout << "일한 개월 수 : " << work\_month << endl;  return work\_month;  }  void Monthly\_Paid::setMonthly\_Pay()  {  monthly\_pay = (work\_time \* pay)\*(work\_month \* 30); // 월급 = (일한 시간 \* 시간당 임금)\*(일한 개월 수 \* 30일(1달))  }  int Monthly\_Paid::getMonthly\_Pay()  {  cout << "계산된 월급 : " << monthly\_pay << endl;  return monthly\_pay;  }  void Monthly\_Paid::print()  {  Worker::print();  getWork\_Time();  getWork\_Month();  setMonthly\_Pay(); // 월급0 계산 함수 호출, 미 호출시 monthly\_pay값 안들어감  getMonthly\_Pay();  }  int main()  {  Worker A;  Worker B("김병호", 10000); // 이름, 시간당 임금  PartTimer C("오순철", 15000, 30); // 이름, 시간당 임금, 일한 시간  PartTimer D("김강호", 15000, 44); // 이름, 시간당 임금, 일한 시간  Monthly\_Paid E("박금주", 15000, 54, 3); // 이름, 시간당 임금, 일한 시간, 일한 개월 수  /\* 생성자 \*/  A.print();  cout << "------------------------" << endl;  B.print();  cout << "------------------------" << endl;  C.print();  cout << "------------------------" << endl;  D.print();  cout << "------------------------" << endl;  E.print();  cout << "------------------------" << endl;  /\* 출력함수 호출 \*/  } |

**4.4 결과 화면**

|  |
| --- |
|  |