|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2019\_1\_C++ \_10 | 학번 : | 20145128 | 이름 : | 박인근 |

* **강의 내용**

|  |
| --- |
| 1. 접근지정자에 따른 상속 : private |
| #include <iostream>  using namespace std;  class Base {  int a;  protected:  void setA(int a) { this->a = a; }  public:  void showA() { cout << a; }  };  class Derived : private Base {  int b;  protected:  void setB(int b) { this->b = b; }  public:  void showB() { cout << b; }  };  int main() {  Derived x;  x.a = 5; // ①  x.setA(10); // ②  x.showA(); // ③  x.b = 10; // ④  x.setB(10); // ⑤  x.showB(); // ⑥  } |
| **[실행결과]**  컴파일 오류가 납니다.  1번은 기본 클래스에서도 private 멤버이므로 파생 클래스에서 무엇을 상속받든 접근이 불가합니다.  4번 또한 파생 클래스에서도 private 멤버이므로 main에서 사용이 불가합니다.  2,3 번은 파생 클래스에서 private로 상속받을 시 기본 클래스의 protected, public 멤버를 private 으로 계승하기 때문에 main에서 사용이 불가능합니다.  5번은 protected 멤버이므로 main에서 사용이 불가합니다.  6번은 public 멤버이므로 main에서 사용이 가능합니다. |

|  |
| --- |
| 1. 접근지정자에 따른 상속 : protected |
| #include <iostream>  using namespace std;  class Base {  int a;  protected:  void setA(int a) { this->a = a; }  public:  void showA() { cout << a; }  };  class Derived : protected Base {  int b;  protected:  void setB(int b) { this->b = b; }  public:  void showB() { cout << b; }  };  int main() {  Derived x;  x.a = 5; // ①  x.setA(10); // ②  x.showA(); // ③  x.b = 10; // ④  x.setB(10); // ⑤  x.showB(); // ⑥  } |
| **[실행결과]**  컴파일 오류가 납니다.  1번은 기본 클래스에서도 private 멤버이므로 파생 클래스에서 무엇을 상속받든 접근이 불가합니다.  4번 또한 파생 클래스에서도 private 멤버이므로 main에서 사용이 불가합니다.  2,3 번은 파생 클래스에서 protected로 상속받을 시 기본 클래스의 protected, public 멤버를 protected 으로 계승하기 때문에 main에서 사용이 불가능합니다.  5번은 protected 멤버이므로 main에서 사용이 불가합니다.  6번은 public 멤버이므로 main에서 사용이 가능합니다. |

|  |
| --- |
| 1. 함수 재정의 : |
| **class Base** {  public:  **void f()** { cout << "Base::f() called" << endl; }  };  **class Derived : public Base** {  public: **//함수 재정의**  **void f()** { cout << "Derived::f() called" << endl; }  };  void main() {  **Derived d**, \*pDer;  pDer = &d;  **pDer->f(); // Derived::f() 호출**  **//pDer->Base::f() …. Base::f() 호출**    Base\* pBase;  pBase = pDer; // 업캐스팅  **pBase->f(); // Base::f() 호출**  } |
| **[실행결과]** |

|  |
| --- |
| 1. 가상 함수와 오버라이딩 |
| **======= Circle.h =====**  #ifndef CIRCLE  #define CIRCLE  class Circle : public Shape {  protected:  virtual void draw();  };  #endif  **======= Line.h ====**  #ifndef LINE  #define LINE  class Line : public Shape {  protected:  virtual void draw();  };  #endif  **======== Rect.h =====**  #ifndef RECT  #define RECT  class Rect : public Shape {  protected:  virtual void draw();  };  #endif  **=======Shape.h ======**  #ifndef SHAPE  #define SHAPE  class Shape {  Shape\* next;  protected:  virtual void draw();  public:  Shape() { next = NULL;}  virtual ~Shape() { }  void paint();  Shape\* add(Shape\* p);  Shape\* getNext() { return next;}  };  #endif  **======== Circle.cpp =====**  #include <iostream>  #include "Shape.h"  #include "Circle.h"  using namespace std;  void Circle::draw() {  cout << "Circle" << endl;  }  **===== Line.cpp ====**  #include <iostream>  #include "Shape.h"  #include "Line.h"  using namespace std;  void Line::draw() {  cout << "Line" << endl;  }  =====Rect.cpp ====  #include <iostream>  #include "Shape.h"  #include "Rect.h"  using namespace std;  void Rect::draw() {  cout << "Rectangle" << endl;  }  **=== Shape.cpp ===**  #include <iostream>  #include "Shape.h"  using namespace std;  void Shape::paint() {  draw();  }  void Shape::draw() {  cout << "Shape" << endl;  }  Shape\* Shape::add(Shape \*p) {  this->next = p;  return p;  }  **==== main.cpp ====**  #include <iostream>  #include "Shape.h"  #include "Circle.h"  #include "Rect.h"  #include "Line.h"  using namespace std;  int main() {  Shape \*pStart=NULL;  Shape \*pLast;  pStart = new Circle(); // 처음에 원 도형을 생성한다.  pLast = pStart;  pLast = pLast->add(new Rect()); // 사각형 객체 생성  pLast = pLast->add(new Circle()); // 원 객체 생성  pLast = pLast->add(new Line()); // 선 객체 생성  pLast = pLast->add(new Rect()); // 사각형 객체 생성  // 현재 연결된 모든 도형을 화면에 그린다.  Shape\* p = pStart;  while(p != NULL) {  p->paint();  p = p->getNext();  }  // 현재 연결된 모든 도형을 삭제한다.  p = pStart;  while(p != NULL) {  Shape\* q = p->getNext(); // 다음 도형 주소 기억  delete p; // 기본 클래스의 가상 소멸자 호출  p = q; // 다음 도형 주소를 p에 저장  }  } |
| **[실행결과]** |

|  |
| --- |
| 1. 가상 함수와 오버라이딩 |
| #include <iostream>  using namespace std;  class Base {  public:  **virtual ~Base()** { cout << "~Base()" << endl; }  };  class Derived: public Base {  public:  **virtual ~Derived()** { cout << "~Derived()" << endl; }  };  int main() {  Derived \*dp = new Derived();  Base \*bp = new Derived();  **delete dp;** // Derived의 포인터로 소멸  **delete bp;** // Base의 포인터로 소멸  } |
| **[실행결과]** |

* 내용 점검

1. 교재 8장 연습문제

|  |  |
| --- | --- |
| 번호 | 문제풀이 |
| 1 | 2 |
| 2 | 1 |
| 3 | 3 |
| 4 | 4, protected 상속을 받았기 때문에 public 접근 불가 |
| 5 | 2, 기본클래스의 private 멤버는 무엇을 상속받든 파생클래스에서 접근 불가 |
| 6 | 1, private 멤버 이기 때문에 main에서 접근 불가  2, 1번과 마찬가지입니다.  4, 파생클래스(B)는 protected 로 A를 상속받았기 때문에 A의 public 멤버 함수인 setX 접근 불가입니다. |
| 7 | (1) 업 캐스팅 : 2, 다운 캐스팅 : 3  (2) 실제 포인터 q가 가리키는 것은 A타입의 객체이기 때문에 멤버 y는 존재하지 않습니다. |
| 8 | (1)3  (2)z,x,w  (3)3  (4) dp = ap; -> dp = (D\*)ap; |
| 9 | (1) 생성자 A  생성자 B //기본 클래스의 생성자 먼저 호출  (2) 생성자 A  생성자 B 10  (3) 생성자 A 32  생성자 B 400 |
| 10 | (1) 기본 클래스 A의 기본 생성자가 존재하기 않기 때문에 오류 발생합니다.  (2) B() : A(20) { cout << “생성자 B” << endl; }  (3) B(int x) : A(x+20) { cout << “생성자 B” << endl; } |
| 11 | 3 반대입니다. |
| 12 | 4 |
| 13 | class Satellite : public Rocket, public Computer { } ; |
| 14 | (1) class Hipen : public Pen, public Eraser { } ;  (2) class OmniPen : public Pen, public Eraser, public Lock { } ; |
| 15 | 4 , fCar가 Car와 Ariplane 두가지로 중복되어 생성되므로 power 멤버 접근 시 , 어떤 것을 선택할지 모호성을 띄기 때문에 오류 발생합니다. |
| 16 | 다중 상속받는 클래스를 생성할 시 screenSize 멤버에 접근 할 때 모호성이 발생하여 컴파일 오류가 발생할 수 있습니다. 따라서 수정하면  class ColorTV : virtual public TV  class InternetTV : virtual public TV  로 수정해야 합니다. |

* 프로그램 과제

|  |
| --- |
| 1. 교재 420p : 문제5번 |
| **[프로그램 소스]**  #include <iostream>  using namespace std;  class BaseArray {  private:  int capacity; // 배열의 크기  int \*mem; //정수 배열을 만들기 위한 메모리 포인터  protected:  BaseArray(int capacity = 100) {  this->capacity = capacity;  mem = new int[capacity];  }  ~BaseArray() { delete[] mem; }  void put(int index, int val) { mem[index] = val; }  int get(int index) { return mem[index]; }  int getCapacity() { return capacity; }  };  class MyQueue : public BaseArray {  int a; //큐의 크기  int b; //원소를 dequeue 할 때 순서대로 출력하기 위한 변수  public:  MyQueue(int size) : BaseArray(size) {  a = 0; b = -1; // b 를 -1 하는 이유는 큐 출력 시 0 번째 부터 순차적으로 출력하기 위해서입니다.  }  void enqueue(int n) {  put(a, n);  a++;  }  int capacity() {  return getCapacity();  }  int length() {  return a;  }  int dequeue() {  a--;  b++;  return get(b);  }  };  int main() {  MyQueue mQ(100);  int n;  cout << "큐에 삽입할 5개의 정수를 입력하라>> ";  for (int i = 0;i < 5;i++) {  cin >> n;  mQ.enqueue(n); //큐에 삽입  }  cout << "큐의 용량:" << mQ.capacity() << ", 큐의 크기:" << mQ.length() << endl;  cout << "큐의 원소를 순서대로 제거하여 출력한다>> ";  while (mQ.length() != 0) {  cout << mQ.dequeue() << ' '; // 큐에서 제거하여 출력  }  cout << endl << "큐의 현재 크기 : " << mQ.length() << endl;  system("pause");  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 421p : 문제6번 |
| **[프로그램 소스]**  #include <iostream>  using namespace std;  class BaseArray {  private:  int capacity; // 배열의 크기  int \*mem; //정수 배열을 만들기 위한 메모리 포인터  protected:  BaseArray(int capacity = 100) {  this->capacity = capacity;  mem = new int[capacity];  }  ~BaseArray() { delete[] mem; }  void put(int index, int val) { mem[index] = val; }  int get(int index) { return mem[index]; }  int getCapacity() { return capacity; }  };  class MyStack : public BaseArray {  int a;  public:  MyStack(int size) : BaseArray(size) {  a = 0;  }  void push(int n) {  put(a, n);  a++;  }  int capacity() {  return getCapacity();  }  int length() {  return a;  }  int pop() {  a--;  return get(a);  }    };  int main() {  MyStack mStack(100);  int n;  cout << "스택에 삽입할 5개의 정수를 입력하라>> ";  for (int i = 0;i < 5;i++) {  cin >> n;  mStack.push(n); //스택에 푸쉬  }  cout << "스택용량:" << mStack.capacity() << ", 스택크기:" << mStack.length() << endl;  cout << "스택의 모든 원소를 팝하여 출력한다>> ";  while (mStack.length() != 0) {  cout << mStack.pop() << ' '; // 스택에서 팝  }  cout << endl << "스택의 현재 크기 : " << mStack.length() << endl;  system("pause");  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 422p : 문제8번 |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class Printer {  string Mname; // 모델명  string manufac; // 제조사  int pCount; // 인쇄 매수  int avaCount; // 인쇄 종이 잔량  public:  Printer(string Mname, string manufac, int ac) {  this->Mname = Mname;  this->manufac = manufac;  this->avaCount = ac;  }  bool print(int pages) {  if (avaCount < pages) {  cout << "용지가 부족하여 프린트할 수 없습니다."<<endl;  return false;  }  else {  avaCount -= pages;  return true;  }  }  string getMname() { return Mname; }  string getManufac() { return manufac; }  int getavaCount() { return avaCount; }  };  class InkjetPrinter : public Printer {  int avaInk; // 잉크 잔량  public:  InkjetPrinter(int avaInk) : Printer("Officejet V40","HP",5) {  this->avaInk = avaInk;  }  void printInkjet(int pages) {  if (print(pages)) {  if (avaInk < pages) {  cout << "잉크가 부족하여 프린트할 수 없습니다."<<endl;  }  else {  avaInk -= pages;  cout << "프린트하였습니다." << endl;  }  }  }  void show() {  cout << getMname() << " ," << getManufac() << " ,남은 종이 " << getavaCount() << "장 ,남은 잉크 " << avaInk << endl;  }  };  class LaserPrinter : public Printer {  int avaToner;  public:  LaserPrinter(int avaToner) : Printer("SCX-6x45","삼성전자",3) {  this->avaToner = avaToner;  }  void printLaser(int pages) {  if (print(pages)) {  if (avaToner < pages) {  cout << "토너가 부족하여 프린트할 수 없습니다."<<endl;  }  else {  avaToner -= pages;  cout << "프린트하였습니다." << endl;  }  }  }  void show() {  cout << getMname() << " ," << getManufac() << " ,남은 종이 " << getavaCount() << "장 ,남은 토너 " << avaToner << endl;  }  };  int main() {  InkjetPrinter \*p = new InkjetPrinter(10);  LaserPrinter \*q = new LaserPrinter(20);  int print, cnt;  char answer;  cout << "현재 작동중인 2 대의 프린터는 아래와 같다" << endl;  cout << "잉크젯 : "; p->show();  cout << "레이저 : "; q->show();    while (1) {  cout << "프린터(1:잉크젯, 2:레이저)와 매수 입력>>";  cin >> print >> cnt;  if (print == 1) {  p->printInkjet(cnt);  p->show();  q->show();  }  if (print == 2) {  q->printLaser(cnt);  p->show();  q->show();  }  cout << "계속 프린트 하시겠습니까(y/n)>>";  cin >> answer;  if (answer == 'y') continue;  else break;  }      system("pause");  } |
| **[실행결과]** |

|  |
| --- |
| 1. 교재 423p 문제9번 |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class Console { //메뉴 출력  public:  int msel() { // 메뉴 선택  int a;  cout << "\n예약:1, 취소:2, 보기:3, 끝내기:4 >>";  cin >> a;  return a;  }  int tsel() { // 비행 시간 선택  int a;  cout << "07시:1, 12시:2, 17시:3 >>";  cin >> a;  return a;  }  };  class Seat {  string name;  int seatNumber;  public:  Seat() {  this->name = "---"; // 예약이 없을 시  this->seatNumber = seatNumber;  }  string getName() { return name; }  void setName(string name) { this->name = name; }  };  class Schedule {  Seat \*p;  public:  Schedule() {  p = new Seat[8]; //8개의 Seat 객체 생성  }  void show(int a) {  if (a == 1) cout << "07시:";  else if (a == 2) cout << "12시:";  else if (a == 3) cout << "17시:";  for (int i = 0; i < 8; i++) {  cout << "\t" << p[i].getName();  }  cout << endl;  }  void setName(int a, string name) {  p[a - 1].setName(name);  }  string getName(int a) {  return p[a - 1].getName();  }  };  class AirlineBook {  Schedule \*p;  public:  AirlineBook() {  p = new Schedule[3]; //Schedule 객체 3개 생성  }  void system() {  int a, snum;  string name;  Console q;  while (1) {  a = q.msel(); // 메뉴 선택 무엇인지  if (a == 1) { // 예약  a = q.tsel();  if (a == 1 || a == 2 || a == 3) {  p[a - 1].show(a); // a-1 해야 0번 정보도 보여줌  cout << "좌석 번호 >>";  cin >> snum;  cin.ignore(1); // 문자를 입력받기 위해 입력 버퍼를 비워줍니다.  cout << "이름 입력 >>";  getline(cin, name);  p[a - 1].setName(snum, name); //좌석번호와 이름을 Schedule 객체에 저장  }  else  cout << "1,2,3 중 하나만 입력해 주십시오." << endl;  }  else if (a == 2) { //취소  a = q.tsel();  if (a == 1 || a == 2 || a == 3) {  p[a - 1].show(a);  cout << "좌석 번호 >>";  cin >> snum;  cin.ignore(1);  cout << "이름 입력 >>";  getline(cin, name);  if (name != p[a - 1].getName(snum)) {  cout << "예약된 이름과 다릅니다.";  }  else  p[a - 1].setName(snum, "---"); // 이름을 비워줍니다.  }  else  cout << "1,2,3 중 하나만 입력해 주십시오." << endl;  }  else if (a == 3) {  p[0].show(1); //07시  p[1].show(2); //12시  p[2].show(3); //17시  }  else if (a == 4) {  cout << "예약 시스템을 종료합니다." << endl;  break;  }  else  cout << "1,2,3,4 중 하나만 입력해 주십시오." << endl;  }  }  };  int main() {    AirlineBook \*p = new AirlineBook();  cout << "\*\*\*\*\*\* 한성한공에 오신것을 환영합니다 \*\*\*\*\*\*" << endl;  p->system();    system("pause");  } |
| **[실행결과]** |