|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2018\_2\_C++ \_11 | 학번 : |  | 이름 : |  |

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  }  } |
| **[실행 결과]** |

* 추상 클래스와 순수 가상 함수

|  |
| --- |
| #include <iostream>  using namespace std;  class Calculator { //추상 클래스  public:  virtual int add(int a, int b) = 0; // 두 정수의 합 리턴, 순수 가상 함수  virtual int subtract(int a, int b) = 0; // 두 정수의 차 리턴  virtual double average(int a [], int size) = 0; // 배열 a의 평균 리턴. size는 배열의 크기  };  class GoodCalc : public Calculator { //추상 클래스 구현  public:  int add(int a, int b) { return a + b; }  int subtract(int a, int b) { return a - b; }  double average(int a [], int size) {  double sum = 0;  for(int i=0; i<size; i++)  sum += a[i];  return sum/size;  }  };  int main() {  int a[] = {1,2,3,4,5};  Calculator \*p = new GoodCalc();  cout << p->add(2, 3) << endl;  cout << p->subtract(2, 3) << endl;  cout << p->average(a, 5) << endl;  delete p;  } |
| **[실행 결과]** |

* 내용 점검

1. 교재 9장 연습문제

|  |  |
| --- | --- |
| 번호 | 문제풀이 |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |
| 8 |  |
| 9 |  |
| 10 |  |
| 11 |  |
| 12 |  |
| 13 |  |
| 14 |  |
| 15 |  |

* 프로그램 과제

1. 교재 466p : 문제 4번

|  |
| --- |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class LoopAdder { // 추상 클래스  string name; // 루프의 이름  int x, y, sum;  void read(); // x, y 값을 읽어 들이는 함수  void write(); // sum을 출력하는 함수  protected:  LoopAdder(string name="") { // 루프의 이름을 받는다. 초깃값은 ""  this->name = name;  }  int getX() { return x; }  int getY() { return y; }  virtual int calculate() = 0; // 순수 가상 함수. 루프를 돌면서 합을 구하는 함수  public:  void run(); // 연산을 진행하는 함수  };  void LoopAdder::read() { // x, y 입력  cout << name << ":" << endl;  cout << "처음 수에서 두번째 수까지 더합니다. 두 수를 입력하세요 >> ";  cin >> x >> y;  }  void LoopAdder::write() { // 결과 sum 출력  cout << x << "에서 " << y << "까지의 합 = " << sum << " 입니다" << endl;  }  void LoopAdder::run() {  read(); // x, y를 읽는다.  sum = calculate(); // 루프돌면서 계산한다.  write(); // 결과 sum을 출력한다.  }  class WhileLoopAdder : public LoopAdder {  protected:  int calculate() {  int sum=0;  int i=getX();  while(i<=getY()) {  sum += i;  i++;  }  return sum;  }  public:  WhileLoopAdder(string name) : LoopAdder(name) {}  };  class DoWhileLoopAdder : public LoopAdder {  protected:  int calculate() {  int sum=0;  int i=getX();  if(i > getY()) return 0;  do {  sum += i;  i++;  } while(i<=getY());  return sum;  }  public:  DoWhileLoopAdder(string name) : LoopAdder(name) {}  };  int main() {  WhileLoopAdder whileLoop("While Loop");  DoWhileLoopAdder doWhileLoop("Do while Loop");  whileLoop.run();  doWhileLoop.run();  } |
| **[실행 결과]** |

1. 교재 467p : 문제 5번

|  |
| --- |
| **[프로그램 소스]**  include <iostream>  using namespace std;  class AbstractGate { // 추상 클래스  protected:  bool x, y;  public:  void set(bool x, bool y) { this->x = x; this->y = y; }  virtual bool operation()=0; // 순수 가상 함수  };  class ANDGate : public AbstractGate {  public:  bool operation() {  return x&&y;  }  };  class ORGate : public AbstractGate {  public:  bool operation() {  return x||y;  }  };  class XORGate : public AbstractGate {  public:  bool operation() {  return x^y;  }  };  int main() {  ANDGate andGate;  ORGate orGate;  XORGate xorGate;  andGate.set(true, false);  orGate.set(true, false);  xorGate.set(true, false);  cout.setf(ios::boolalpha); // 불린 값을 "true', "false" 문자열로 출력할 것을 지시  cout << andGate.operation() << endl; // AND 결과는 false  cout << orGate.operation() << endl; // OR 결과는 true  cout << xorGate.operation() << endl; // XOR 결과는 true  } |
| **[실행 결과]** |

1. 교재 469p : 문제 7번

|  |
| --- |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class Shape {  protected:  string name; // 도형의 이름  int width, height; // 도형이 내접하는 사각형의 너비와 높이  public:  Shape(string n="", int w=0, int h=0) { name = n; width = w; height = h; }  virtual double getArea() { return 0; } // dummy 값 리턴  string getName() { return name; } // 이름 리턴  };  class Oval : public Shape {  public:  Oval(string name, int width, int height) : Shape(name, width, height) {}  virtual double getArea() { return 3.14\*width\*height; }  };  class Rect : public Shape {  public:  Rect(string name, int width, int height) : Shape(name, width, height) {}  virtual double getArea() { return width\*height; }  };  class Triangular : public Shape {  public:  Triangular(string name, int width, int height) : Shape(name, width, height) {}  virtual double getArea() { return width\*height/2; }  };  int main() {  Shape \*p[3];  p[0] = new Oval("빈대떡", 10, 20);  p[1] = new Rect("찰떡", 30, 40);  p[2] = new Triangular("토스트", 30, 40);  for(int i=0; i<3; i++)  cout << p[i]->getName() << " 넓이는 " << p[i]->getArea() << endl;  for(int i=0; i<3; i++) delete p[i];  } |
| **[실행 결과]** |

1. 교재 469p : 문제 8번

|  |
| --- |
| **[프로그램 소스]**  #include <iostream>  #include <string>  using namespace std;  class Shape {  protected:  string name; // 도형의 이름  int width, height; // 도형이 내접하는 사각형  public:  Shape(string n="", int w=0, int h=0) { name = n; width = w; height = h; }  virtual double getArea()=0;  string getName() { return name; } // 이름 리턴  };  class Oval : public Shape {  public:  Oval(string name, int width, int height) : Shape(name, width, height) {}  virtual double getArea() { return 3.14\*width\*height; }  };  class Rect : public Shape {  public:  Rect(string name, int width, int height) : Shape(name, width, height) {}  virtual double getArea() { return width\*height; }  };  class Triangular : public Shape {  public:  Triangular(string name, int width, int height) : Shape(name, width, height) {}  virtual double getArea() { return width\*height/2; }  };  int main() {  Shape \*p[3];  p[0] = new Oval("빈대떡", 10, 20);  p[1] = new Rect("찰떡", 30, 40);  p[2] = new Triangular("토스트", 30, 40);  for(int i=0; i<3; i++)  cout << p[i]->getName() << " 넓이는 " << p[i]->getArea() << endl;  for(int i=0; i<3; i++) delete p[i];  } |
| **[실행 결과]** |

1. 교재 471p : 문제 9번

|  |
| --- |
| **[프로그램 소스]**  ========== BasePrinter.h =======  #ifndef BASEPRINTER\_H  #define BASEPRINTER\_H  #include <string>  using namespace std;  class BasePrinter {  string model;  string manufacturer;  int printedCount;  int availableCount;  public:  BasePrinter(string model, string manufacturer, int totalPapers);  protected:  virtual bool print(int pages=1);  virtual void show();  string getModel() { return model; }  string getManufacturer() { return manufacturer; }  int getPrintedCount() { return printedCount; }  int getAvailableCount() { return availableCount; }  };  ========== InkJetPrinter.h =======  #ifndef INKJETPRINTER\_H  #define INKJETPRINTER\_H  #include <string>  #include "BasePrinter.h"  using namespace std;  class InkJetPrinter : public BasePrinter {  int availableInk;  public:  InkJetPrinter(string model, string manufacturer, int totalPapers, int totalInk);  bool print(int pages=1);  void show();  };  #endif  ========== LaserPrinter.h =======  #ifndef LASERPRINTER\_H  #define LASERPRINTER\_H  #include <string>  #include "BasePrinter.h"  using namespace std;  class LaserPrinter : public BasePrinter {  int availableToner;  public:  LaserPrinter(string model, string manufacturer, int totalPapers, int totalToner);  bool print(int pages=1);  void show();  };  #endif  ========== PrinterManager.h =======  #ifndef PRINTERMANAGER\_H  #define PRINTERMANAGER\_H  #include "InkJetPrinter.h"  #include "LaserPrinter.h"  class PrinterManager {  InkJetPrinter \*ip;  LaserPrinter \*lp;  public:  PrinterManager();  ~PrinterManager();  void operate();  };  #endif  ========== BasePrinter.cpp =======  #include <iostream>  #include "BasePrinter.h"  using namespace std;  BasePrinter::BasePrinter(string model, string manufacturer, int totalPapers) {  this->model = model;  this->manufacturer = manufacturer;  this->printedCount = 0;  this->availableCount = totalPapers;  }  bool BasePrinter::print(int pages) {  if(availableCount < pages) {  cout << "용지가 부족하여 프린트할 수 없습니다." << endl;  return false;  }  printedCount+=pages;  availableCount-=pages;  return true;  }  void BasePrinter::show() {  cout << model+" ," << manufacturer << " ,남은 종이 " << availableCount << "장" << " ,";  }  ========== PrinterManager.cpp =======  #include <iostream>  #include "PrinterManager.h"  using namespace std;  PrinterManager::PrinterManager() {  ip = new InkJetPrinter("Officejet V40", "HP", 5, 10);  lp = new LaserPrinter("SCX-6x45", "삼성전자", 3, 20);  }  PrinterManager::~PrinterManager() {  delete ip;  delete lp;  }  void PrinterManager::operate() {  cout << "현재 작동중인 2 대의 프린터는 아래와 같다" << endl;  cout << "잉크젯 : ";  ip->show();  cout << endl;  cout << "레이저 : ";  lp->show();  cout << endl;  int printer, pages;  char conti;  while(true) {  cout << endl << "프린터(1:잉크젯, 2:레이저)와 매수 입력>>";  cin >> printer >> pages;  switch(printer) {  case 1 : // 잉크젯  ip->print(pages);  break;  case 2 : // 레이저  lp->print(pages);  break;  }  ip->show();  cout << endl;  lp->show();  cout << endl;  cout << "계속 프린트 하시겠습니까(y/n)>>";  cin >> conti;  if(conti == 'y')  continue;  else  break;  }  }  ========== LaserPrinter.cpp =======  #include <iostream>  #include "BasePrinter.h"  #include "LaserPrinter.h"  using namespace std;  LaserPrinter::LaserPrinter(string model, string manufacturer, int totalPapers, int totalToner)  : BasePrinter(model, manufacturer, totalPapers)  {  this->availableToner = totalToner;  }  bool LaserPrinter::print(int pages) {  if(!BasePrinter::print(pages)) // BasePrinter::print() 호출  return false; // 용지 부족  if(availableToner < pages) {  cout << "토너가 부족하여 프린트할 수 없습니다." << endl;  return false;  }  availableToner--;  cout << "프린트하였습니다. " << endl;  return true;  }  void LaserPrinter::show() {  BasePrinter::show(); // BasePrinter::show() 호출  cout << "남은토너 " << availableToner;  }  ========== InkJetPrinter.cpp =======  #include <iostream>  #include "BasePrinter.h"  #include "InkJetPrinter.h"  using namespace std;  InkJetPrinter::InkJetPrinter(string model, string manufacturer, int totalPapers, int totalInk)  : BasePrinter(model, manufacturer, totalPapers)  {  this->availableInk = totalInk;  }  bool InkJetPrinter::print(int pages) {  if(!BasePrinter::print(pages)) // BasePrinter::print() 호출  return false; // 용지 부족  if(availableInk < pages) {  cout << "잉크가 부족하여 프린트할 수 없습니다." << endl;  return false;  }  availableInk-=pages;  cout << "프린트하였습니다. " << endl;  return true;  }  void InkJetPrinter::show() {  BasePrinter::show(); // BasePrinter::show() 호출  cout << "남은 잉크 " << availableInk;  }  ============== main.cpp ==========  #include "PrinterManager.h"  int main() {  PrinterManager man;  man.operate();  } |
| **[실행 결과]** |