Topic 5
Classes

Definition of Classes
Data member and Member function
Static data member and static member function
const member function
Self reference: this
Nested classes

Characteristics of Objects

- An Object is an instance in the real world.
  - has an identifier that uniquely identify itself
  - has property that it holds
  - has behavior that it provides

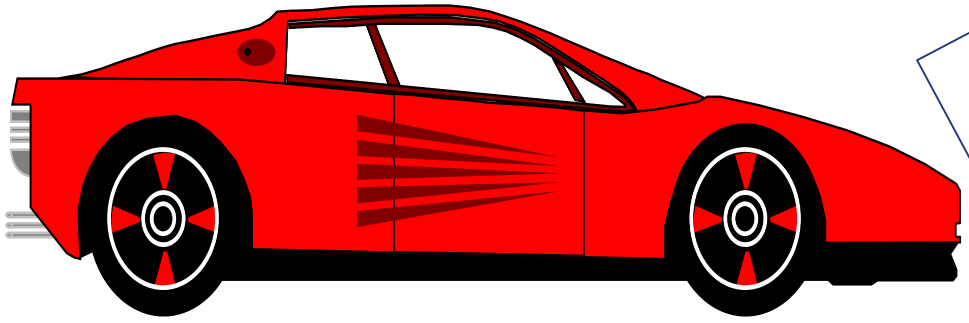
Objects: Example

ID: James

Property
James
35
CA
Intel Corp.

Behavior
work
hike
eat
drink

Objects: Example



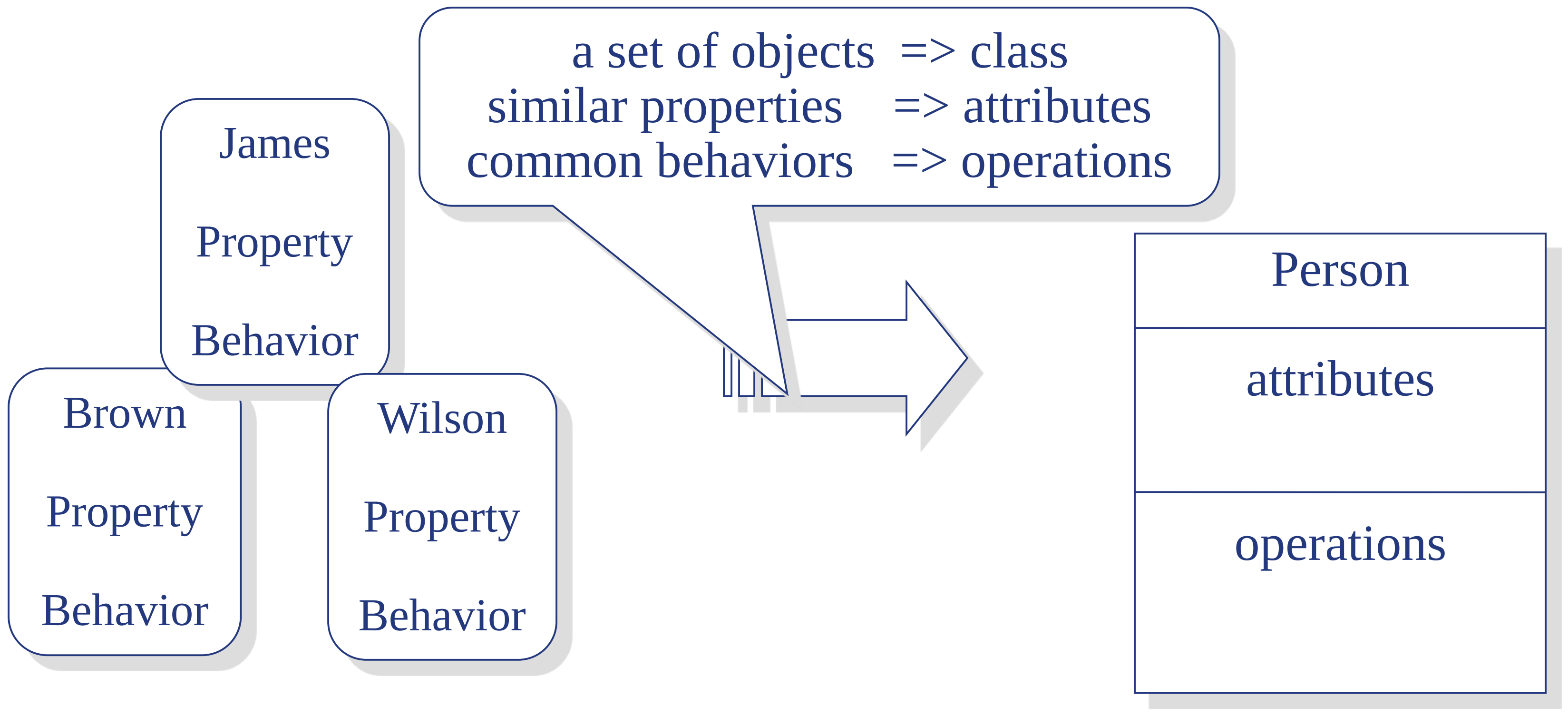
aCar

Property
red
5000 cc
200 km/h

Behavior
start
stop
speed up
speed down

Class

- A class defines a set of objects with similar properties and common behavior



Class Definition in C++

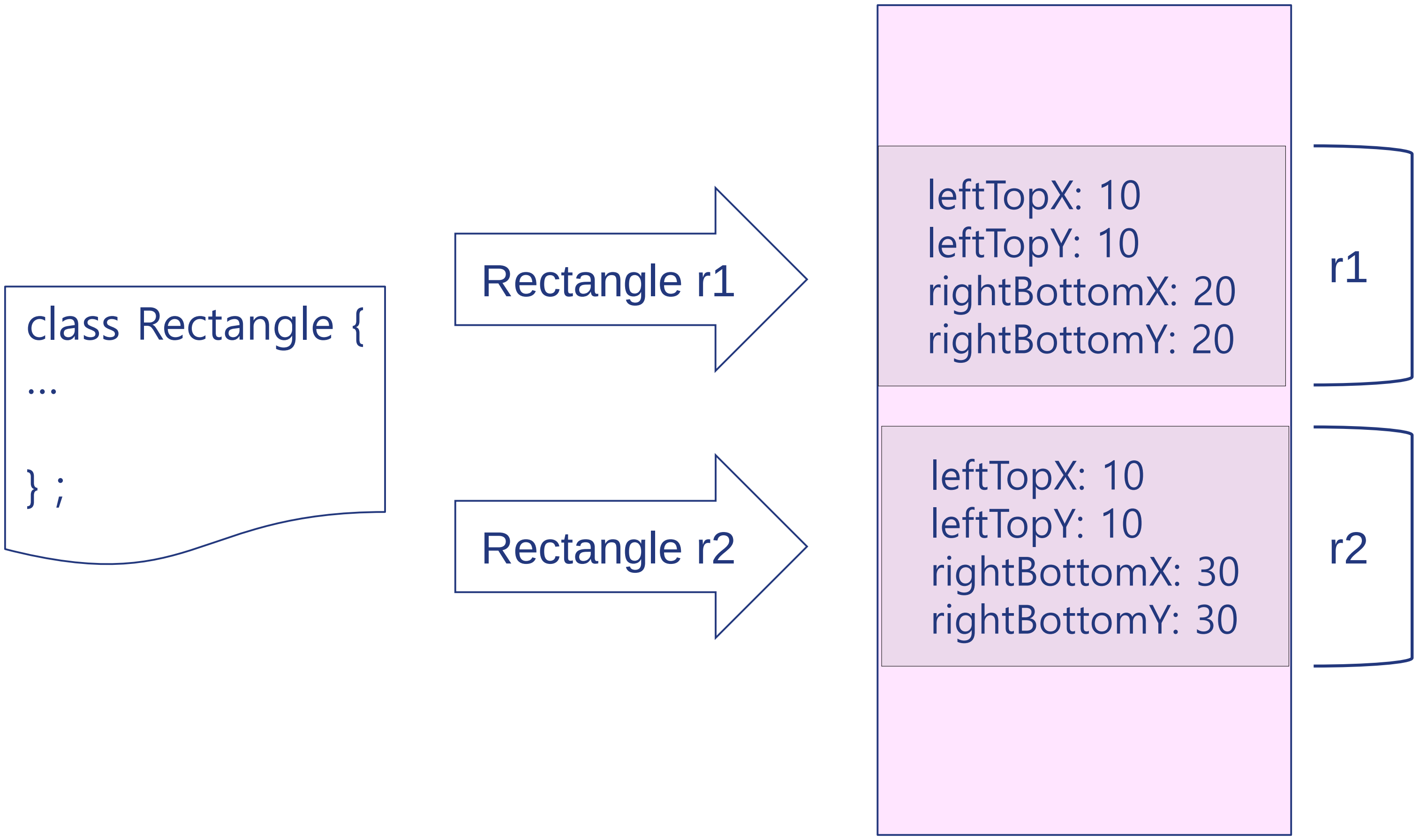
```
class  이름 {
   private:
      데이터 멤버
   public:
      멤버 함수
} ;
```

Class

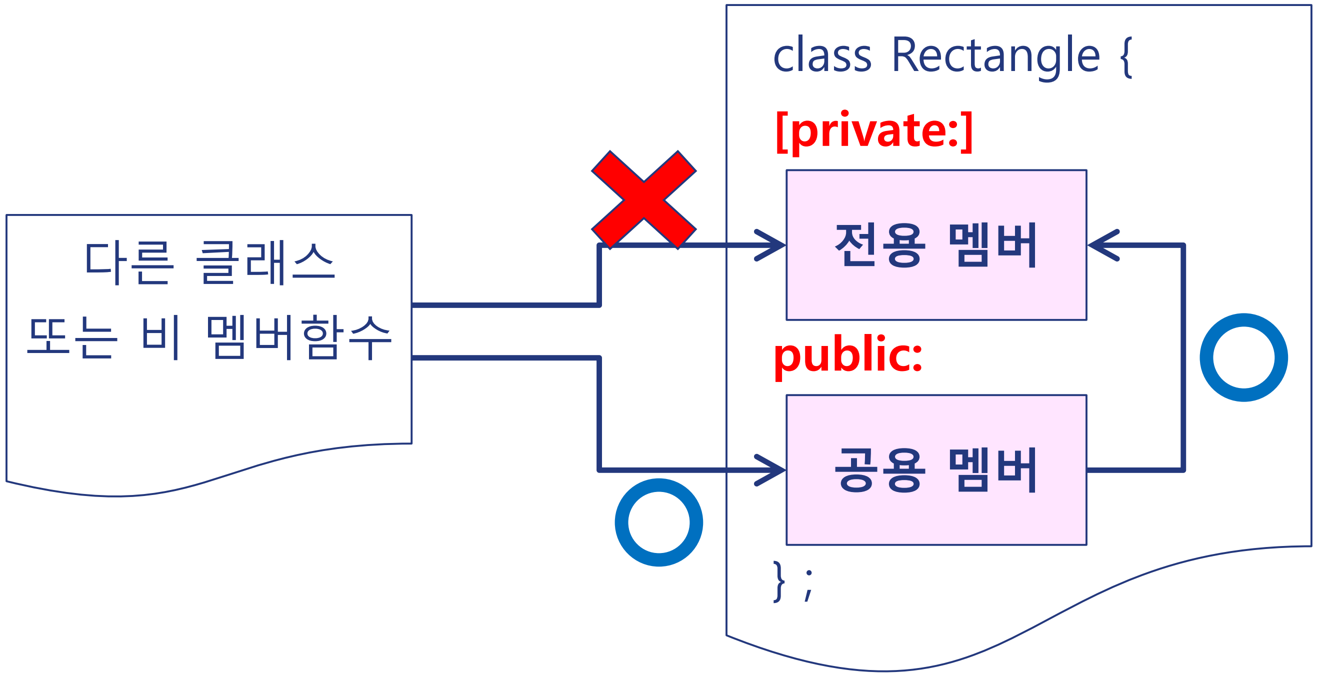
```
# include <iostream>
using namespace std ;

class Rectangle {
private:
  int leftTopX, leftTopY ;
  int rightBottomX, rightBottomY ;
public:
  Rectangle(int x1, int y1, int x2, int y2) {
    set(x1, y1, x2, y2) ;
  }
  void set(int x1, int y1, int x2, int y2) {
    leftTopX = x1 ; leftTopY = y1 ;
    rightBottomX = x2 ; rightBottomY = y2 ;
  }
  void getLeftTop(int& x, int& y) {
    x = leftTopX ; y = leftTopY ;
  }
  void getRightBottom(int& x, int& y) {
    x = rightBottomX ; y = rightBottomY ;
  }
  int getArea() {
    return (rightBottomX - leftTopX) *
      (rightBottomY - leftTopX) ;
  }
} ;
```

```
int main() {
    Rectangle r1(10, 10, 20, 20) ;

    int x1, y1, x2, y2 ;
    r1.getLeftTop(x1, y1) ;
    r1.getRightBottom(x2, y2) ;

    Rectangle r2(x1+10, y1+10,
        x2+10, y2+10) ;

    cout << r1.getArea() << '\t' <<
        r2.getArea() << endl ;
}
```

객체의 생성


class Rectangle {
…
} ;
Rectangle r1
Rectangle r2
leftTopX: 10
leftTopY: 10
rightBottomX: 20
rightBottomY: 20
leftTopX: 10
leftTopY: 10
rightBottomX: 30
rightBottomY: 30
r1
r2

정보은닉: private과 public



클래스의 정의: Rectangle.h

```
# ifndef __RECTANGLE_H
# define __RECTANGLE_H

class Rectangle {
public:
    int leftTopX, leftTopY ;
    int rightBottomX, rightBottomY ;
    // 클래스 내부에서 정의된 멤버 함수는 기본적으로 inline 함수임
    void setLeftTop(int x, int y) { leftTopX = x ; leftTopY = y ; }
    void setRightBottom(int x, int y) { rightBottomX = x ; rightBottomY = y ; }

    // 자신의 멤버 함수를 호출함
    void set(int x1, int y1, int x2, int y2) { setLeftTop(x1, y1) ; setRightBottom(x2, y2) ; }
    void getLeftTop(int& x, int& y) { x = leftTopX ; y = leftTopY ; }
    void getRightBottom(int& x, int& y) { x = rightBottomX ; y = rightBottomY ; }

    int getWidth() { return rightBottomX - leftTopX ; }
    int getHeight() { return rightBottomY - leftTopY ; }

    // 별도의 구현 파일을 이용함
    int getArea() ;
    void moveBy(int deltaX, int deltaY) ;
} ;
# endif
```

Rectangle.cpp

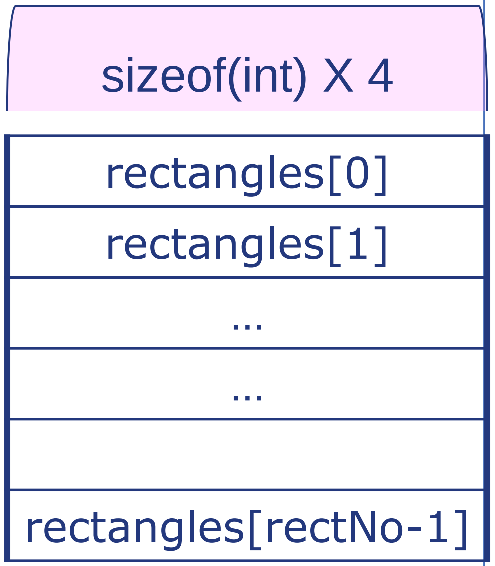
```
// Rectangle.cpp

# include "Rectangle.h"

// 클래스 멤버 함수를 클래스 외부에 정의하고 있음
int Rectangle::getArea() {
    return getWidth() * getHeight() ;
}

void Rectangle::moveBy(int deltaX, int deltaY) {
    setLeftTop(leftTopX+deltaX, leftTopY+deltaY) ;
    setRightBottom(rightBottomX+deltaX, rightBottomY+deltaY) ;
}
```

RectangleMain.cpp

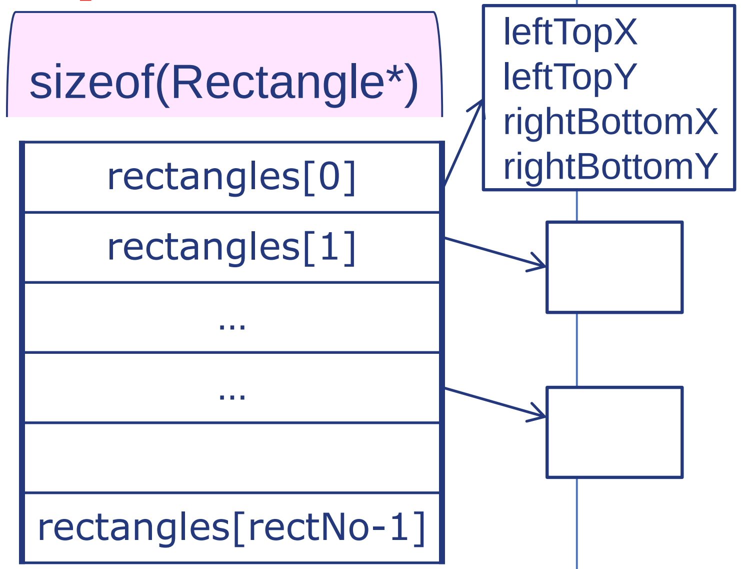
```
# include <iostream>
# include "Rectangle.h"
using namespace std ;

int main() {
    int x1, y1, x2, y2 ;
    cin >> x1 >> y1 >> x2 >> y2 ;

    Rectangle r1 ;
    r1.set(x1, y1, x2, y2) ;

    int x3, y3, x4, y4 ;
    r1.getLeftTop(x3, y3) ;
    r1.getRightBottom(x4, y4) ;

    Rectangle r2 ;
    r2.set(x3, y3, x4, y4) ;
    r2.moveBy(10, 20) ;

    cout << endl << r1.getArea() << '\t' << r2.getArea() << endl ;
}
```

객체의 동적 생성

```
# include <iostream>
# include "Rectangle.h"
using namespace std ;

int main() {
   int rectNo ;
   cin >> rectNo ;
   Rectangle* const rectangles = new Rectangle[rectNo] ;

   for ( unsigned int i = 0 ; i < rectNo ; i ++ ) {
      cout << "Enter Rectangle information" << endl ;
      int x1, y1, x2, y2 ;
      cin >> x1 >> y1 >> x2 >> y2 ;
      rectangles[i].set(x1, y1, x2, y2) ;
   }
   int totalArea = 0 ;
   for ( unsigned int i = 0 ; i < rectNo ; i ++ )  totalArea += rectangles[i].getArea() ;

   delete [] rectangles ;
   cout << "The total area: " << totalArea << endl ;
}
```

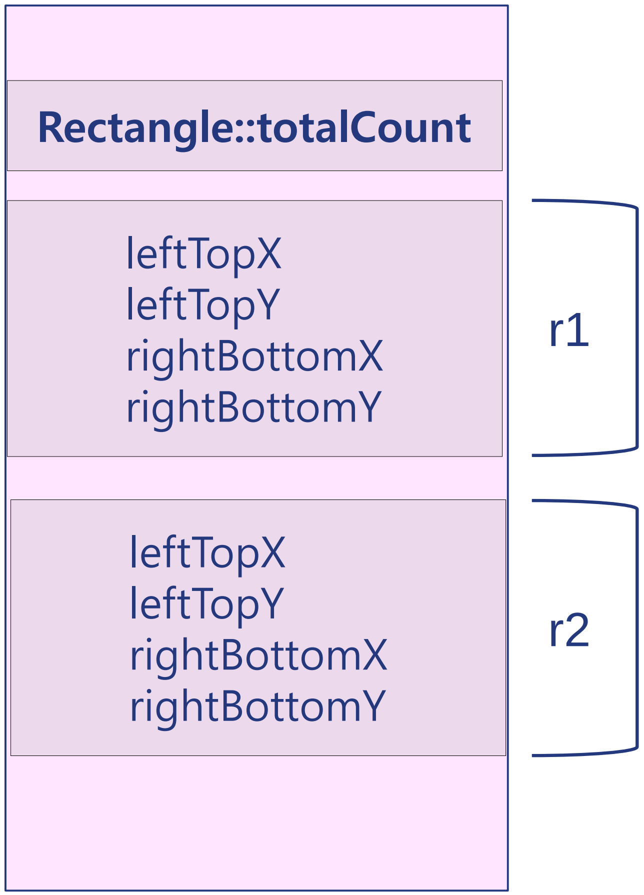


```
int main() {
    int rectNo ;
    cin >> rectNo ;
    Rectangle** const rectangles = new Rectangle*[rectNo] ;

    int count = 0 ;
    while ( true && count < rectNo) {
       string command ;
       cin >> command ;
       if ( command == "ADD" ) {
          const Rectangle* const r = readRectangle() ;
          rectangles[count] = r ;
          count ++ ;
       }
       else if ( command == "AREA" ) {
          cout << getTotalArea(rectangles, count) << endl ;
       }
       else if ( command == "CLEAR" ) {
          deleteRectangles(rectangles, count) ;
          count = 0 ;
       }
       else {
          cerr << "Invalid command!" << endl ;
       }
    }
    delete [] rectangles ;
}
```



```
# include <iostream>
# include <string>
# include "Rectangle.h"
using namespace std ;

Rectangle* readRectangle() {
  int x1, y1, x2, y2 ;
  cin >> x1 >> y1 >> x2 >> y2 ;

  Rectangle* const r = new Rectangle ;
  r->set(x1, y1, x2, y2) ;
  return r ;
}
void deleteRectangles(Rectangle ** const ppR, int n) {
  for ( int i = 0 ; i < n ; i ++ ) {
    delete ppR[i] ;
    ppR[i] = ‘\0’ ;
  }
}
int getTotalArea(Rectangle **  const ppR, int n) {
  int totalArea = 0 ;
  for ( int i = 0 ; i < n ; i ++ )
    totalArea += ppR[i]->getArea() ;
  return totalArea ;
}
```

Static data member



Static data member

```
// Rectangle.h
# ifndef __RECTANGLE_H
# define __RECTANGLE_H

class Rectangle {
public:
   static int allCount ;

   int leftTopX, leftTopY ;
   int rightBottomX, rightBottomY ;

   void setLeftTop(int x, int y) {
      leftTopX = x ; leftTopY = y ;
   }
   void setRightBottom(int x, int y) {
      rightBottomX = x ; rightBottomY = y ;
   }
   …
} ;
# endif
```

```
// Rectangle.cpp
# include "Rectangle.h"

int Rectangle::allCount = 0 ;
…
```

```
# include <iostream>
# include "Rectangle.h"
using namespace std ;

int main() {
   Rectangle r1 ;
   r1.set(1, 1, 2, 2) ;
   cout << Rectangle::allCount << endl ;

   Rectangle r2 ;
   r2.set(10, 10, 20, 20) ;
   cout << Rectangle::allCount << endl ;
}
```

```
# ifndef __RECTANGLE_H
# define __RECTANGLE_H

class Rectangle {
public:
   static int allCount ;

   int leftTopX, leftTopY ;
   int rightBottomX, rightBottomY ;

   // 생성자: 객체 생성시 자동 호출됨
   Rectangle() { allCount ++ ; }
   // 소멸자: 객체 소멸시 자동 호출됨
   ~Rectangle() { allCount -- ; }
   …
} ;
# endif
```

```
# include "Rectangle.h"


int Rectangle::allCount = 0 ;


…
```

```
# include <iostream>
# include "Rectangle.h"
using namespace std ;

Rectangle gRectangle1, gRectangle2 ;

int main() {
     cout << Rectangle::allCount << endl ;
     Rectangle r1 ;
     cout << Rectangle::allCount << endl ;

     for ( int i = 0 ; i < 3 ; i ++ ) {
          Rectangle r ;
          cout << Rectangle::allCount << endl ;
     }

     Rectangle* pR = new Rectangle ;
     cout << Rectangle::allCount << endl ;
     delete pR ;
     cout << Rectangle::allCount << endl ;
}
```

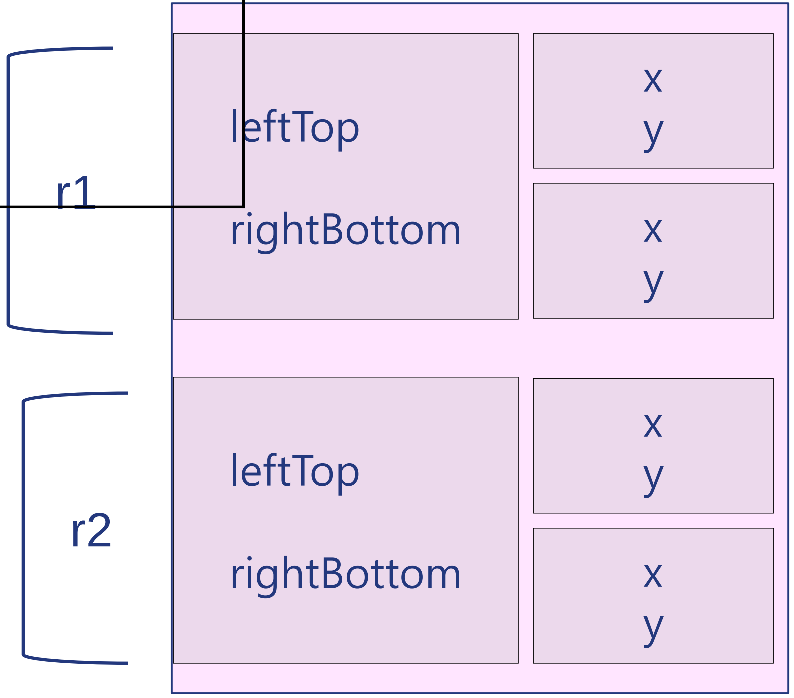
Static member function

```
# ifndef __RECTANGLE_H
# define __RECTANGLE_H

class Rectangle {
   static int allCount ;
   int leftTopX, leftTopY ;
   int rightBottomX, rightBottomY ;
public:
   // 정적 데이터멤버만 호출가능함
   static int getAllCount() { return allCount ; }
   static bool noRectangle() { return allCount == 0 ; }
   Rectangle() { allCount ++ ; }
   ~Rectangle() { allCount -- ; }
   …
} ;
# endif
# include <iostream>
# include <vector>
# include <string>
# include "Rectangle.h"
using namespace std ;
int main() {
  vector<Rectangle*> rectangles ;
  do {
     string command ;
     cin >> command ;
     if ( command == "ADD" )
        rectangles.push_back(new Rectangle) ;
     else if ( command == "DELETE" ) {
        vector<Rectangle*>::iterator head = rectangles.begin() ;
        Rectangle* r = *head ;
        delete r ;
        rectangles.erase(head) ;
     }
     else break ;
     cout << Rectangle::getAllCount() << endl ;
  } while ( Rectangle::noRectangle() == false ) ;
  for ( vector<Rectangle*>::iterator Iter = rectangles.begin( ) ; Iter != rectangles.end( ) ; Iter++ ) {
     Rectangle* r = *Iter ;
     delete r ;
  }
}
```

객체 데이터 멤버

```
class Point {
private:
  int x ;
  int y ;
public:
  Point( int _x=0, int _y=0) {
    x = _x  ;  y = _y ;
  }
} ;
```

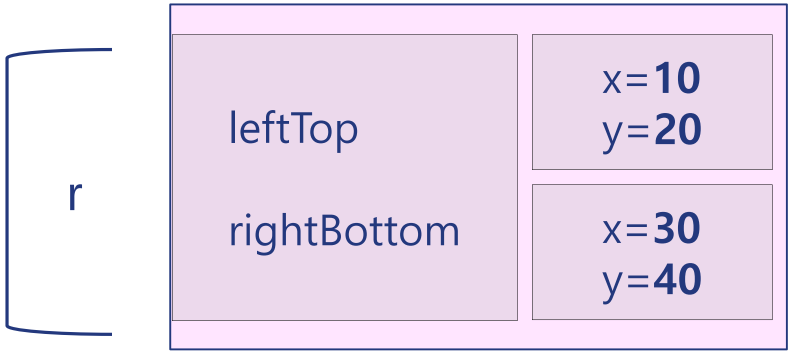
```
class Rectangle {
private:
  Point leftTop ;
  Point rightBottom ;
public:
  …
} ;
```



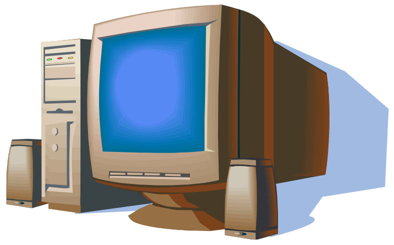
멤버 초기화 목록 – 객체 데이터 멤버의 초기화

```
Rectangle(int x1, int y1, int x2, int y2)
    : leftTop(x1, y1), rightBottom(x2, y2)
    { }
```

```
Rectangle r(10, 20, 30, 40) ;
```



Computer and Monitor



samsungPC
(samsungMonitor)

hpPC
(hpMonitor)

Computer and Monitor

예제 프로그램	실행 결과

`int main() {`
`  Monitor samsungMonitor("SamsungMonitor", 100) ;`
`  Computer samsungPC("Samsung",`
`    samsungMonitor,`
`    samsungMonitor.getPrice() + 200) ;`
 	
`  cout << samsungPC.getPrice() << endl ;`
`  samsungPC.run("Hello C++") ;`
`}`
	
	`300`
	`Runs on Samsung`
	`Samsung Monitor: Hello C++`

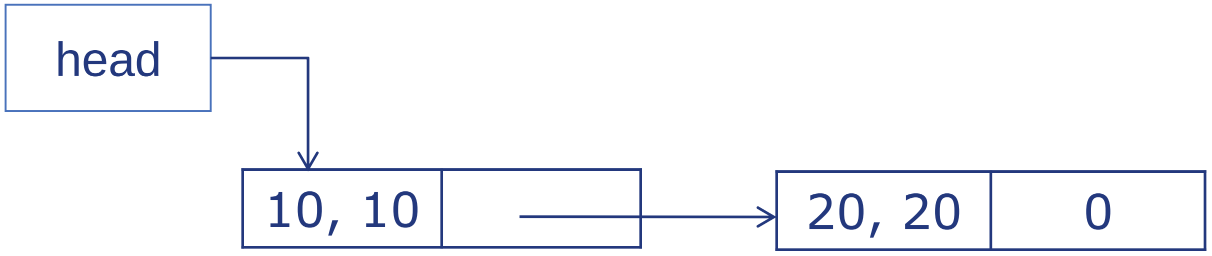
Monitor

```
# include <iostream>
# include <string>
using namespace std ;

class Monitor {
  string maker ;
  int price ;
public:
  Monitor(const string& _maker, int _price) : maker(_maker) { price = _price ; }
  void display(const string& msg) {
     cout << maker << ": " << msg << endl ;
  }
  int getPrice() { return price ; }
} ;
```

Computer

```
class Computer {
  string maker ;
  Monitor monitor ; // Monitor 객체를 데이터 멤버로 가짐
  int price ;
public:
  Computer(const string& _maker, const Monitor& _monitor, const int _price) :
    maker(_maker), monitor(_monitor) {
    price = _price ;
   }
  void run(const string& msg) {
    cout << "Runs on " << maker << endl ;
    monitor.display(msg) ;
  }
  int getPrice() { return price ; }
} ;
```

단방향 연결 리스트



```
class Point {
   int x, y ;
   Point* pNext ;
public:
   Point(int _x=0, int _y=0, Point* const _pNext=0) {
      x = _x ; y = _y ;
      pNext = _pNext ;
   }
   Point* getNext() { return pNext ; }
   void setNext(Point* const _pNext) { pNext = _pNext ; }
   void print() {
      cout << x << ", " << y << endl ;
   }
}
```

단방향 연결 리스트

```
class SinglyLinkedList {
  Point* head ;
public:
  SinglyLinkedList() { head = 0 ; }
  void print() {
    Point* pPoint = head ;
    while ( pPoint ) {
      pPoint->print() ;
      pPoint = pPoint->getNext() ;
    }
  }
  void append(Point* const newPoint) {
    if ( head == 0 ) head = newPoint ;
    else head->setNext(newPoint) ;
  }
  void insertAfter(Point* const prev, Point* const newPoint) {
    newPoint->setNext(prev->getNext()) ;
    prev->setNext(newPoint) ;
  }
```

단방향 연결 리스트

```
  void remove(Point* const toBeRemoved) {
    if ( toBeRemoved == head ) {
      head = head->getNext() ; return ;
    }
    Point* pPoint = head ;
    while ( pPoint ) {
      if ( pPoint->getNext() == toBeRemoved ) {
        Point* pNext = toBeRemoved->getNext() ;
        pPoint->setNext(pNext) ;
        break ;
      }
      pPoint = pPoint->getNext() ;
    }
  }
} ;
```

const member function

```
# ifndef __RECTANGLE_H
# define __RECTANGLE_H

class Rectangle {
public:
   static int allCount ;
   int leftTopX, leftTopY ;
   int rightBottomX, rightBottomY ;

   Rectangle() { allCount ++ ; }
   ~Rectangle() { allCount -- ; }
   static int getAllCount() { return allCount ; } // not const
   static bool noRectangle() { return allCount == 0 ; }
   void setLeftTop(int x, int y) { leftTopX = x ; leftTopY = y ; }
   void setRightBottom(int x, int y) { rightBottomX = x ; rightBottomY = y ; }
   void set(int x1, int y1, int x2, int y2) { setLeftTop(x1, y1) ; setRightBottom(x2, y2) ; }
   void getLeftTop(int& x, int& y) const { x = leftTopX ; y = leftTopY ; }
   void getRightBottom(int& x, int& y) const { x = rightBottomX ; y = rightBottomY ; }
   int getWidth() const { return rightBottomX - leftTopX ; }
   int getHeight() const { return rightBottomY - leftTopY ; }
   int getArea() const ;
   void moveBy(int deltaX, int deltaY) ;
} ;
# endif
# include "Rectangle.h"
int Rectangle::allCount = 0 ;
int Rectangle::getArea() const { return getWidth() * getHeight() ; }
void Rectangle::moveBy(int deltaX, int deltaY) {
  setLeftTop(leftTopX+deltaX, leftTopY+deltaY) ;
  setRightBottom(rightBottomX+deltaX, rightBottomY+deltaY) ;
}
```

```
# include <iostream>

# include "Rectangle.h"
using namespace std ;

void readRectangle(Rectangle& r) {
   int x1, y1, x2, y2 ;
   cin >> x1 >> y1 >> x2 >> y2 ;
   r.setLeftTop(x1, y1) ; r.setRightBottom(x2, y2) ;
}
void printRectangle(const Rectangle& r) {
   int x1, y1, x2, y2 ;
   r.getLeftTop(x1, y1) ; r.getRightBottom(x2, y2) ;
   cout << x1 << '\t' << y1 << '\t' << x2 << '\t' << y2 << endl ;
   // r.setLeftTop(0, 0) ; // ERROR
}
int main() {
   Rectangle r ;
   readRectangle(r) ;
   printRectangle(r) ;
}
```

this

❖ pointer to the object itself

```
# ifndef __RECTANGLE_H
# define __RECTANGLE_H
class Rectangle {
  int leftTopX, leftTopY ;
  int rightBottomX, rightBottomY ;
public:
  ...
  // this의 활용 예1)
  void setLeftTopX(const int leftTopX) { this->leftTopX = leftTopX ; }
  void setLeftTopY(const int leftTopY) { this->leftTopY = leftTopY ; }
  void setRightBottomX(const int rightBottomX) { this->rightBottomX = rightBottomX ; }
  void setRightBottomY(const int rightBottomY) { this->rightBottomY = rightBottomY ; }
} ;
# endif
```

this의 활용 예

```
# ifndef __RECTANGLE_H
# define __RECTANGLE_H
class Rectangle {
public:
  Rectangle* copy() const {
     Rectangle* r = new Rectangle ;
     r->setLeftTopX(getLeftTopX()) ;
     r->setLeftTopY(getLeftTopY()) ;
     r->setRightBottomX(getRightBottomX()) ;
     r->setRightBottomY(getRightBottomY()) ;
     return r ;
  }
Rectangle* copy() const {
     Rectangle* r = new Rectangle(*this) ;
     return r ;
  }
  bool isEqual(const Rectangle& r) const {
     return leftTopX == r.leftTopX && leftTopY == r.leftTopY
        && rightBottomX == r.rightBottomX
        && rightBottomY == r.rightBottomY ;
  }
} ;
# endif
```

```
# include <iostream>
# include <cassert>
# include "Rectangle.h"
using namespace std ;

int main() {
   Rectangle r ;
   r.set(0, 0, 100, 200) ;

   Rectangle* pR = r.copy() ;
   assert ( pR->isEqual(r) ) ;

   delete pR ;
}
```

this의 활용 예

```
# include <iostream>

# include "Rectangle.h"
using namespace std ;

int main() {
  Rectangle r ;
  r.set(0, 0, 100, 200) ;

  cout << r.moveBy(10, 10).print() << endl ; // 10 10 110 210
  // expected: 30 30 130 230, but actually 20 20 120 220
  cout << r.moveBy(10, 10).moveBy(10, 10).print() << endl ;

  r.moveBy(10, 10).print() ;
  cout << r.moveBy(10, 10).print().getArea() << endl ;
}
```

this의 활용 예

```
# ifndef __RECTANGLE_H
# define __RECTANGLE_H
class Rectangle {
public:
  …
  Rectangle& moveBy(int deltaX, int deltaY) ;
  const Rectangle& print() const {
    cout << leftTopX << '\t' << leftTopY << '\t' << rightBottomX << '\t' << rightBottomY << endl ;
    return *this ;
  }
  int getArea() const { … }
} ;
# endif
```

```
# include "Rectangle.h"
…
Rectangle& Rectangle::moveBy(int deltaX, int deltaY) {
  setLeftTop(leftTopX+deltaX, leftTopY+deltaY) ;
  setRightBottom(rightBottomX+deltaX, rightBottomY+deltaY) ;
  return *this ;
}
```

Nested classes

```
…
#include <iostream>
using namespace std ;
class Rectangle {
public:
   class Point {
   public:
      int x, y ;
      void print() const { cout << x << '\t' << y ; }
      bool isEqual(const Point& p) const { return x == p.x && y == p.y ; }
   } ;

   Point leftTop, rightBottom ;
   void setLeftTop(int x, int y) { leftTop.x = x ; leftTop.y = y ; }
   void setRightBottom(int x, int y) { rightBottom.x = x ; rightBottom.y = y ; }
   bool isEqual(const Rectangle& r) const {
      return  leftTop.isEqual(r.leftTop) && rightBottom.isEqual(r.rightBottom) ;
   }
   const Rectangle& print() const {
      leftTop.print() ; cout << '\t' ; rightBottom.print() ;
      return *this ;
   }
   …
}
```

Nested classes

```
# include <iostream>
# include <string>
using namespace std ;
# include "Rectangle.h"

int main() {
    Rectangle r1 ;
    r1.set(0, 0, 100, 200) ;

    Rectangle r2 ;
    r2.set(10, 10, 110, 210) ;

    r1.print() ; cout << endl ;
    r2.print() ; cout << endl ;

    string msg = r1.isEqual(r2) ? "same" : "different" ;
    cout << msg << endl ;

    Rectangle::Point pt ;
    // 가능은 하지만, 이럴 필요가 있으면Point를 nested class로 하지 않는 것이 좋다
}
```