Default Constructor

- A default constructor is a constructor which can be called with no arguments - either defined with an 1) empty parameter list, or with 2) default arguments provided for every parameter.

```
class Rectangle {
    int leftTopX, leftTopY ;
    int rightBottomX, rightBottomY ;
public:
    Rectangle() { set(0, 0, 0, 0) ; }
    …
} ;
```

```
class Rectangle {
    int leftTopX, leftTopY ;
    int rightBottomX, rightBottomY ;
public:
    Rectangle(int x1=0, int y1=0,
        int x2=0, int y2=0) {
        set(x1, y1, x2, y2);
    }
    …
} ;
```

```
int main() {
    Rectangle r1 ;
    // Rectangle r1()로 하면 Rectangle을
    // return하는 함수r1()에 대한 선언과 혼동됨
    r1.print();
    Rectangle r2 = Rectangle();
    r2.print();

    Rectangle* const pR = new Rectangle() ;
    pR->print();

    delete pR ;
}
```

Default Constructor

- A compiler will generate default constructor, if there are no constructors.

```
class Rectangle {
    int leftTopX, leftTopY ;
    int rightBottomX, rightBottomY ;
public:
    // 생성자가 없음
    // 기본 생성자가 자동으로 생성됨
    // Rectangle() { }
    void set(int x1, int y1, int x2, int y2) {
        ...
    }
    ...
    ...
} ;
```

```
int main() {
    //기본 생성자로 객체 생성
    Rectangle r1 ;
    r1.print();             // garbage

    Rectangle* const pR = new Rectangle() ;
    pR->print();            // 0, 0, 0, 0

    delete pR ;

    Rectangle r2 = Rectangle();
    r2.print();             // 0, 0, 0, 0

    Rectangle r3{} ;  // since C++ 11
    r3.print();             // 0, 0, 0, 0
}
```

Defaulted Constructor

- if T is a class type with explicitly-declared default constructor, then the object is zero-initialized.
- Use =default instead of {}

```
class Rectangle {
   int leftTopX, leftTopY ;
   int rightBottomX, rightBottomY ;
public:
   Rectangle() = default;
   …
} ;
```

```
int main() {
   Rectangle r1;
   r1.print();          // garbage

   Rectangle r2{};
   r2.print();          // 0, 0, 0, 0
}
```

```
class Rectangle {
   int leftTopX, leftTopY ;
   int rightBottomX, rightBottomY ;
public:
   Rectangle() {}
   …
} ;
```

```
int main() {
   Rectangle r1;
   r1.print();          // garbage

   Rectangle r2{};
   r2.print();          // garbage
}
```

Delegating Constructor

- A constructor that calls another constructor is known as a delegating constructor
- A constructor can call another constructor to avoid code duplication among constructors

```
class Point {
    int x, y;
public:
    Point(int _x=0, int _y=0) : x{_x}, y{_y} {}
};
class Rectangle {
    Point leftTop;
    Point rightBottom;
public:
    Rectangle() = default;
    Rectangle(int x, int y)
        : Rectangle(x, y, x+10, y+10) {}
    Rectangle(int x1, int y1, int x2, int y2)
        : leftTop(x1, y1), rightBottom(x2, y2) {}
};
```

```
int main() {
    Rectangle r1{};
    Rectangle r2(0, 0, 10, 10);
    Rectangle r3(0, 0);
}
```

Deleted Constructor

- The use of constructor is suppressed by specifying **= delete**

```
class Rectangle {
    Point leftTop, rightBottom;
public:
   Rectangle() = delete;                                    //constructor()
    Rectangle(int x, int y) : Rectangle(x, y, x+10, y+10) {}
    Rectangle(int x1, int y1, int x2, int y2) : leftTop(x1, y1), rightBottom(x2, y2) {}
    //Rectangle(const Rectangle& r) = delete;      //copy constructor
};
Rectangle readRectangle() {
    int x1, y1, x2, y2;
    cin >> x1 >> y1 >> x2 >> y2;
    //return Rectangle(x1, y1, x2, y2); // ERROR use of deleted copy constructor
    return Rectangle(x1, y1, x2, y2);
}
int main() {
   // Rectangle r1{}; // ERROR use of deleted constructor
   Rectangle r2(0, 0, 100, 100);
   // Rectangle r3{r1}; // ERROR use of deleted copy constructor
}
```

Good Design: 기본 생성자는 단순하고 예외를 던지지 않는 것이 좋다! (C.44)

- 실패할 수도 있는 연산 없이 기본적인 값을 설정할 수 있는 것은 에러 핸들링과 이동 연산에 관한 추론을 단순화 함

```
// elem은 nullptr이거나, new를 사용해 할당된 공간을 가리킨다.
class Vector1 {
    public:
        // {nullptr, nullptr, nullptr}과 동일하다. 예외를 던지지 않는다.
        Vector1() noexcept {}
        Vector1(int n) :elem{new int [n]}, space{elem + n}, last{elem} {}
    private:
        unique_ptr<int*> elem = nullptr;
        int* space = nullptr;
        int* last = nullptr;
};
```