**Lab 3**

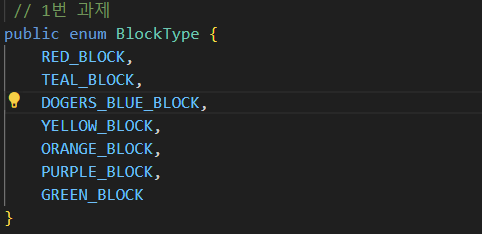
**1분반**

**2024년 05월 01일**

**32211792**

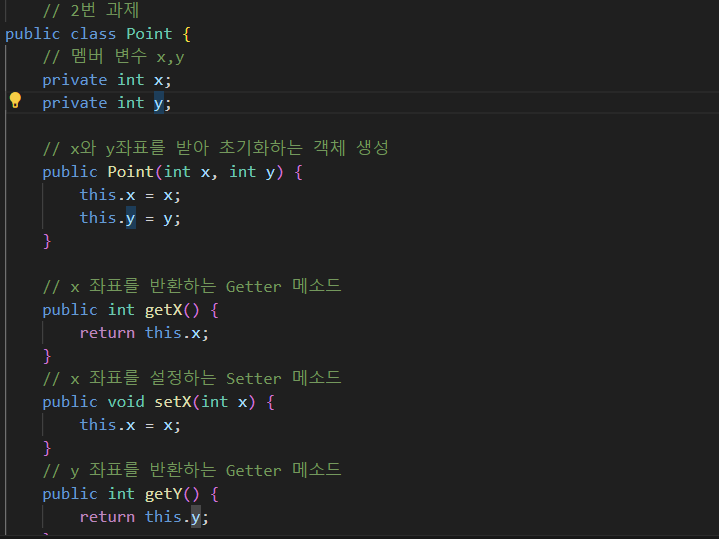
**박재홍**

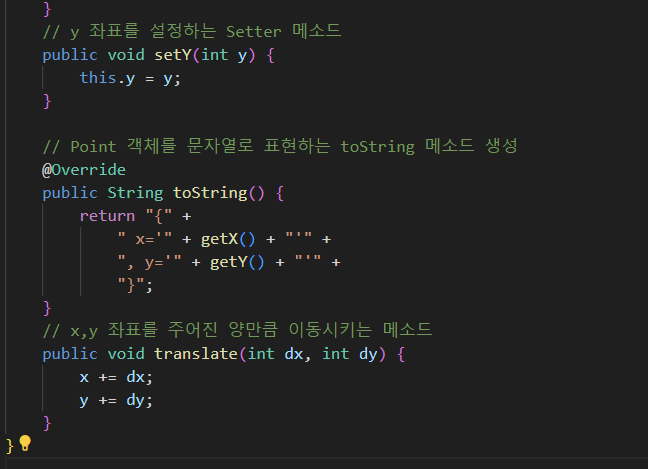
**1번 과제)**



enum 열거형을 이용해 다양한 종류의 블록을 나타내는 상수들을 정의해주었다. 각 상수는 특정 색상의 블록을 나타내며, 해당하는 블록의 종류를 식별하는데에 사용된다. 이러한 열거상수는 블록을 생성하고 식별하는데에 사용된다

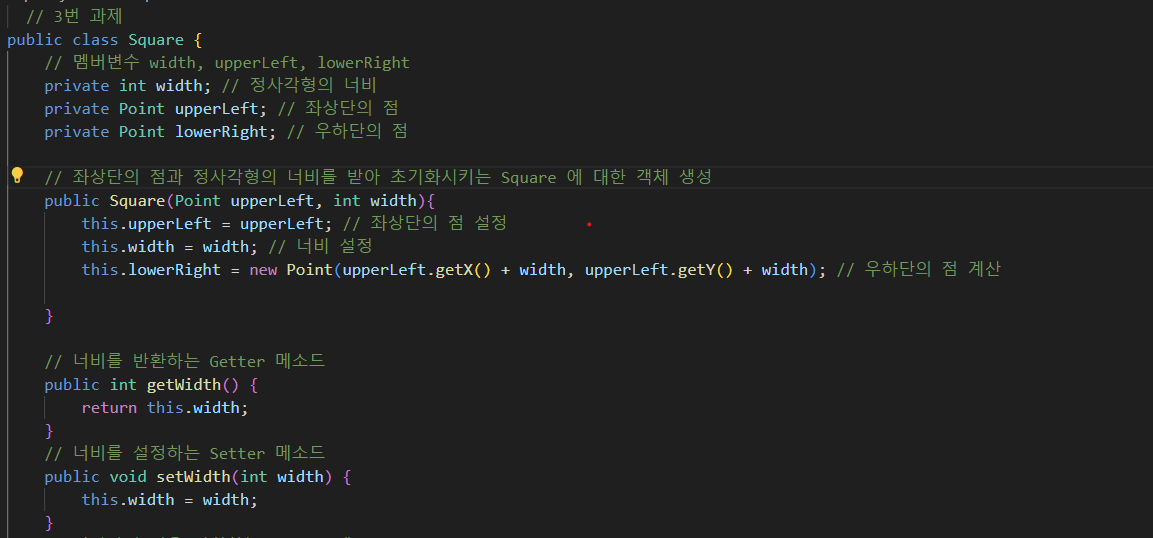
**2번 과제)**

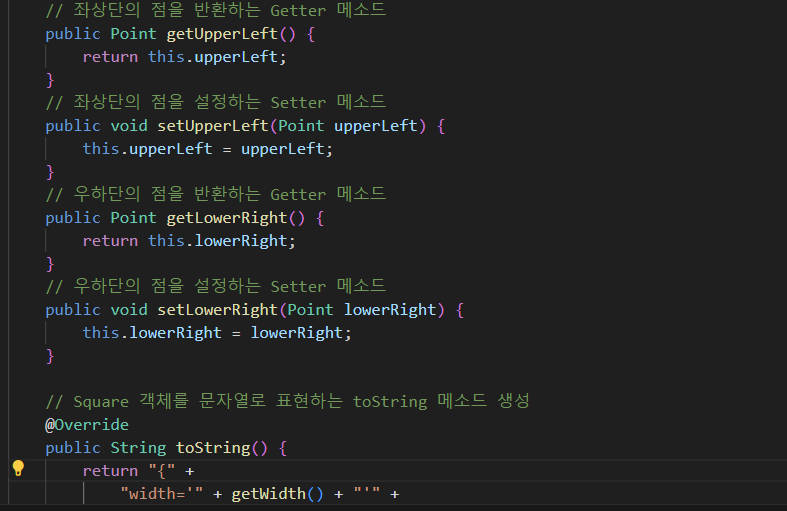


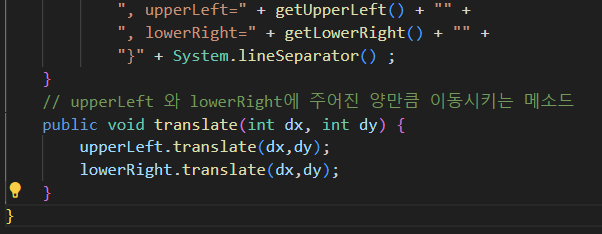


Point 클래스는 2차원 공간에서의 한 점을 나타내는 클래스입니다. 멤버 변수로 x와 y 좌표를 가지고 있으며, 이를 초기화하고 조작하는 메소드들을 포함하고 있습니다. getX()와 getY() 메소드로 좌표를 가져올 수 있고, setX()와 setY() 메소드로 좌표를 설정할 수 있습니다. 또한, translate(int dx, int dy) 메소드를 통해 주어진 양만큼 좌표를 이동시킬 수 있습니다. 마지막으로, toString() 메소드를 통해 Point 객체를 문자열로 표현할 수 있습니다.

**3번 과제)**

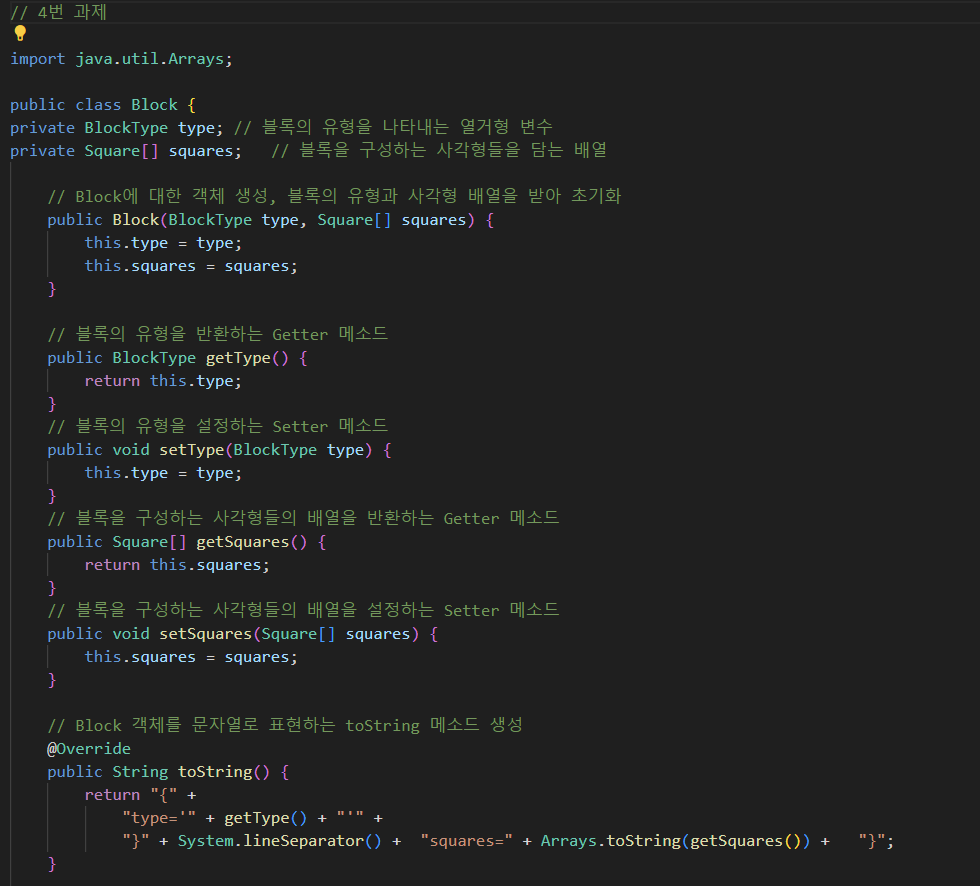


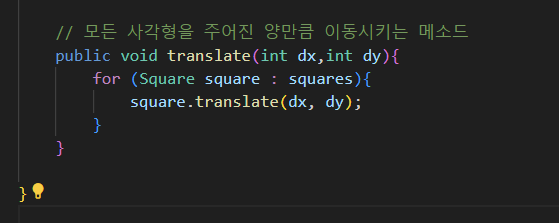




Square 클래스는 정사각형을 나타내는 클래스입니다. 멤버 변수로는 정사각형의 너비(width), 좌상단의 점(upperLeft), 우하단의 점(lowerRight)이 있습니다. Square 객체는 주어진 좌상단의 점과 너비를 이용하여 초기화됩니다. translate(int dx, int dy) 메소드를 통해 좌상단의 점과 우하단의 점을 주어진 양만큼 이동시킬 수 있습니다. 마지막으로, toString() 메소드를 통해 Square 객체를 문자열로 표현할 수 있습니다.

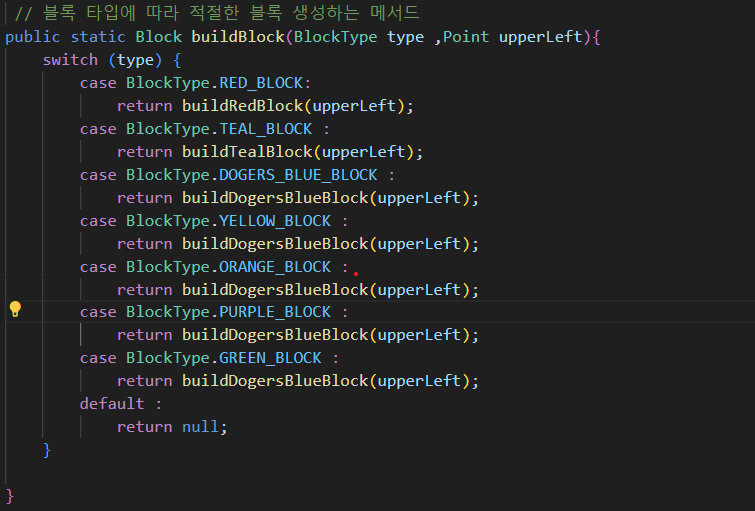
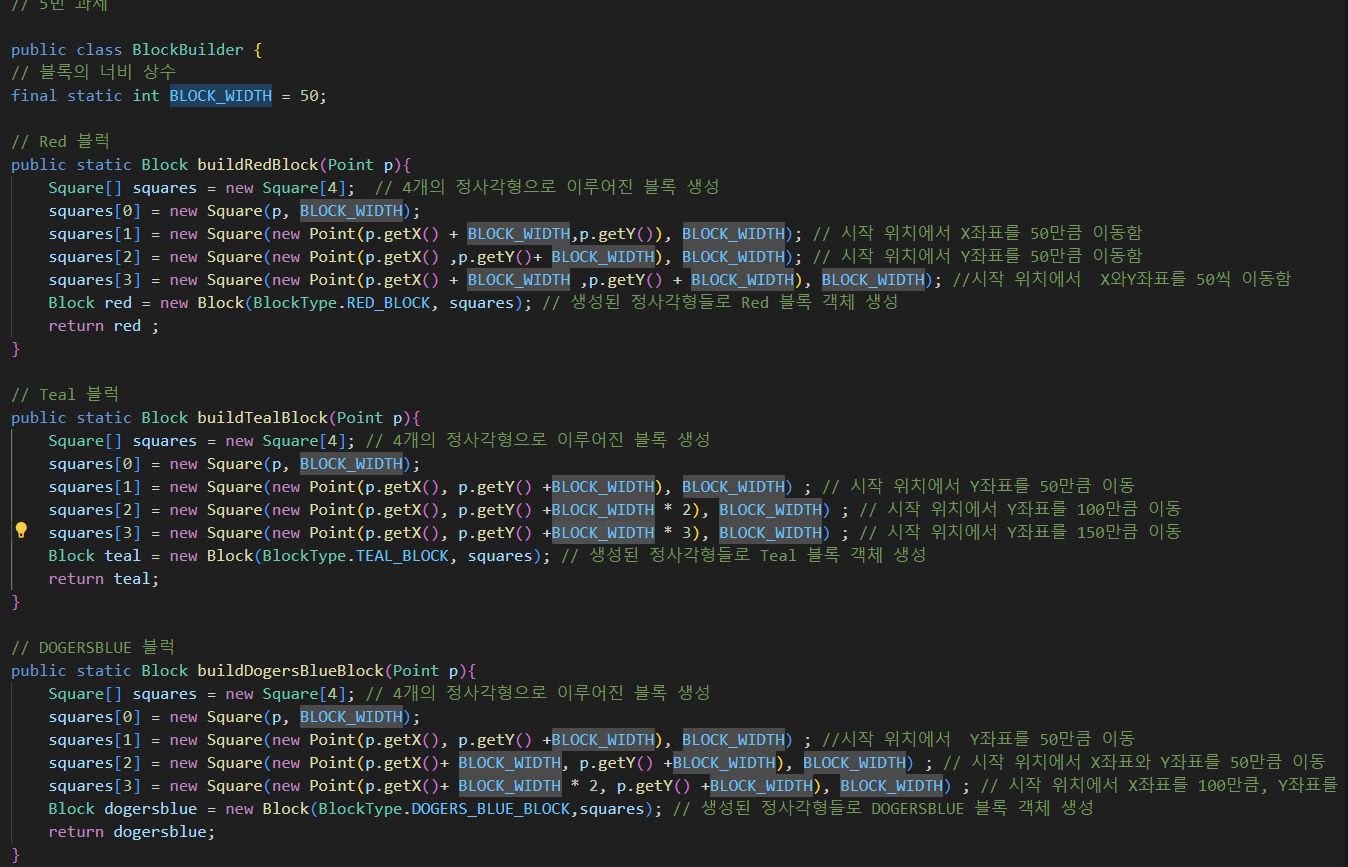
**4번 과제)**





Block 클래스는 블록을 나타내는 클래스입니다. 멤버 변수로는 블록의 유형을 나타내는 BlockType 변수와 블록을 구성하는 사각형들을 담는 Square 배열이 있습니다. Block 생성자를 통해 블록의 유형과 사각형 배열을 초기화할 수 있습니다. 각각의 Getter와 Setter 메소드를 통해 블록의 유형과 사각형 배열에 접근하고 수정할 수 있습니다. toString() 메소드를 통해 Block 객체를 문자열로 표현할 수 있습니다. translate(int dx, int dy) 메소드를 통해 블록을 이동시킬 수 있습니다.

**5번 과제)**



BlockBuilder 클래스는 블록을 생성하는 빌더 클래스입니다.

BLOCK\_WIDTH상수는 블록의 너비를 나타내며, 모든 블록의 너비가 동일하다고 가정합니다.

buildRedBlock(Point p)메서드는 주어진 시작 점 p를 기준으로 Red 블록을 생성합니다. Red 블록은 2x2 크기의 정사각형 4개로 이루어져 있으며, 각 정사각형은 시작 위치에서 하나 생성하고 그 다음은 시작 위치에서 X좌표를 50만큼 이동한 위치에서 하나 생성되고 그 다음은 시작 위치에서 Y좌표를 50만큼 이동한 위치에서 생성되고 그 다음은 시작 위치에서 X좌표와Y좌표에서 50씩 이동한 위치에서 생성된다.

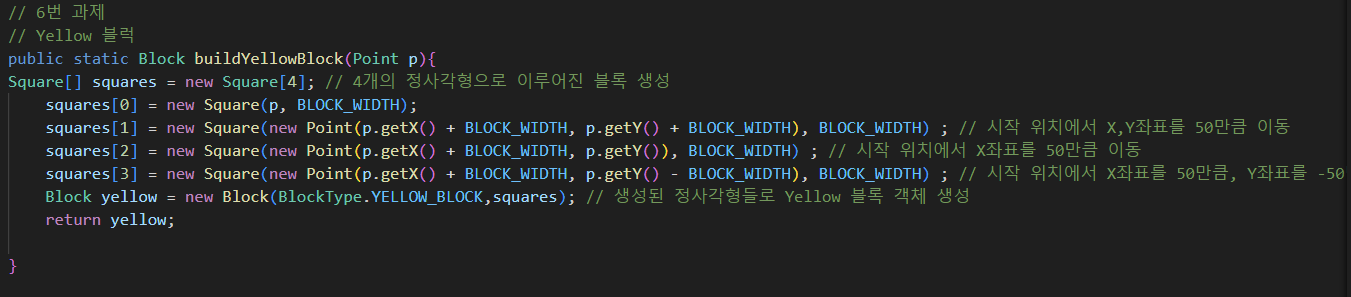
buildTealBlock(Point p)메서드는 주어진 시작 점 p를 기준으로 Teal 블록을 생성합니다. Teal 블록은 1x4 크기의 정사각형 4개로 이루어져 있으며, 각 정사각형은 시작 위치에서 하나 생성되고 그 다음은 시작 위치에서 Y좌표를 50만큼 이동한 위치에서 생성되고 그 다음은 시작 위치에서 Y좌표를 100만큼 이동한 위치에서 생성되고 그 다음은 시작 위치에서 Y좌표를 100만큼 이동한 위치에서 생성된다.

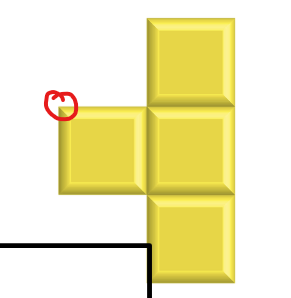
buildDogersBlueBlock(Point p)메서드는 주어진 시작 점 p를 기준으로 Dogers Blue 블록을 생성합니다. Dogers Blue 블록은 3x1 크기의 정사각형 4개로 이루어져 있으며, 각 정사각형은 시작 위치에서 하나 생성하고 그 다음은 시작 위치에서 Y좌표를 50만큼 이동한 위치에서 생성되고 그 다음은 시작 위치에서 X와 Y좌표를 50만큼 이동한 회의에서 생성되고 그 다음은 X좌표 100, Y좌표를 50만큼 이동한 위치에서 생성된다.

buildBlock(BlockType type, Point upperLeft)메서드는 주어진 블록 유형과 시작 점을 기준으로 해당 유형의 블록을 생성합니다. 이 메서드는 switch-case문을 사용하여 주어진 블록 유형에 따라 적절한 블록을 생성하는 방식으로 동작합니다. 블록 유형에 따라 적절한 블록 생성 메서드를 호출하여 블록을 생성하고 반환합니다.

이렇게 구현된 BlockBuilder 클래스를 사용하면 주어진 시작 점을 기준으로 다양한 유형의 블록을 생성할 수 있습니다.

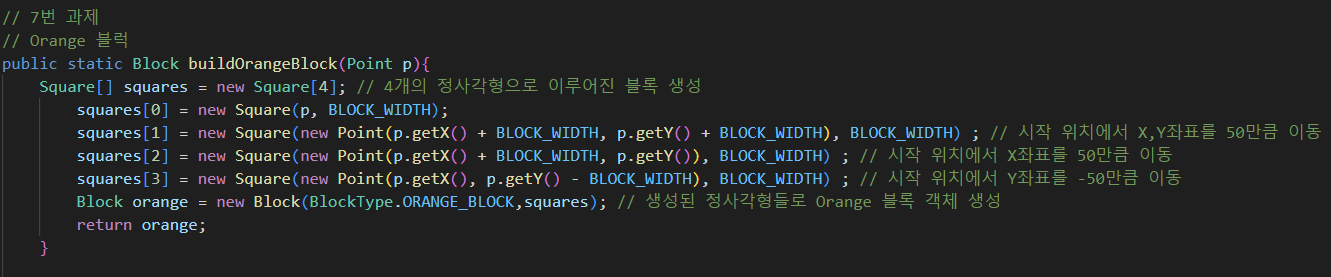
**6번 과제)**

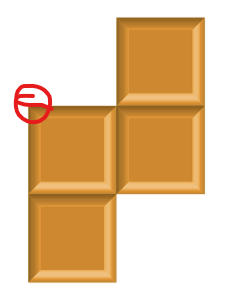
****



buildYellowBlock(Point p)메서드는 주어진 시작 점 p를 기준으로 Yellow Blue 블록을 생성합니다. Yellow 블록은 3x1 크기의 정사각형 4개로 이루어져 있으며, 시작 지점은 빨간점에서 시작한다. 각 정사각형은 시작 위치에서 하나 생성하고 그 다음은 시작 위치에서 X와 Y좌표를 50만큼 이동한 위치에서 생성되고 그 다음은 시작 위치에서 X 좌표를 50만큼 이동한 위치에서 생성되고 그 다음은 X좌표 50, Y좌표를 -50만큼 이동한 위치에서 생성된다.

**7번 과제)**

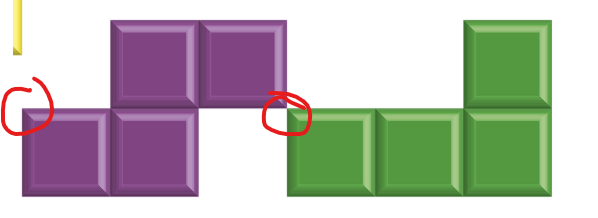




buildOrangeBlock(Point p)메서드는 주어진 시작 점 p를 기준으로 Orange Blue 블록을 생성합니다. Orange 블록은 2x2 크기의 정사각형 4개로 이루어져 있으며, 시작 지점은 빨간점에서 시작한다. 각 정사각형은 시작 위치에서 하나 생성하고 그 다음은 시작 위치에서 X와 Y좌표를 50만큼 이동한 위치에서 생성되고 그 다음은 시작 위치에서 X 좌표를 50만큼 이동한 위치에서 생성되고 그 다음은 Y좌표를 -50만큼 이동한 위치에서 생성된다.

**8번 과제)**

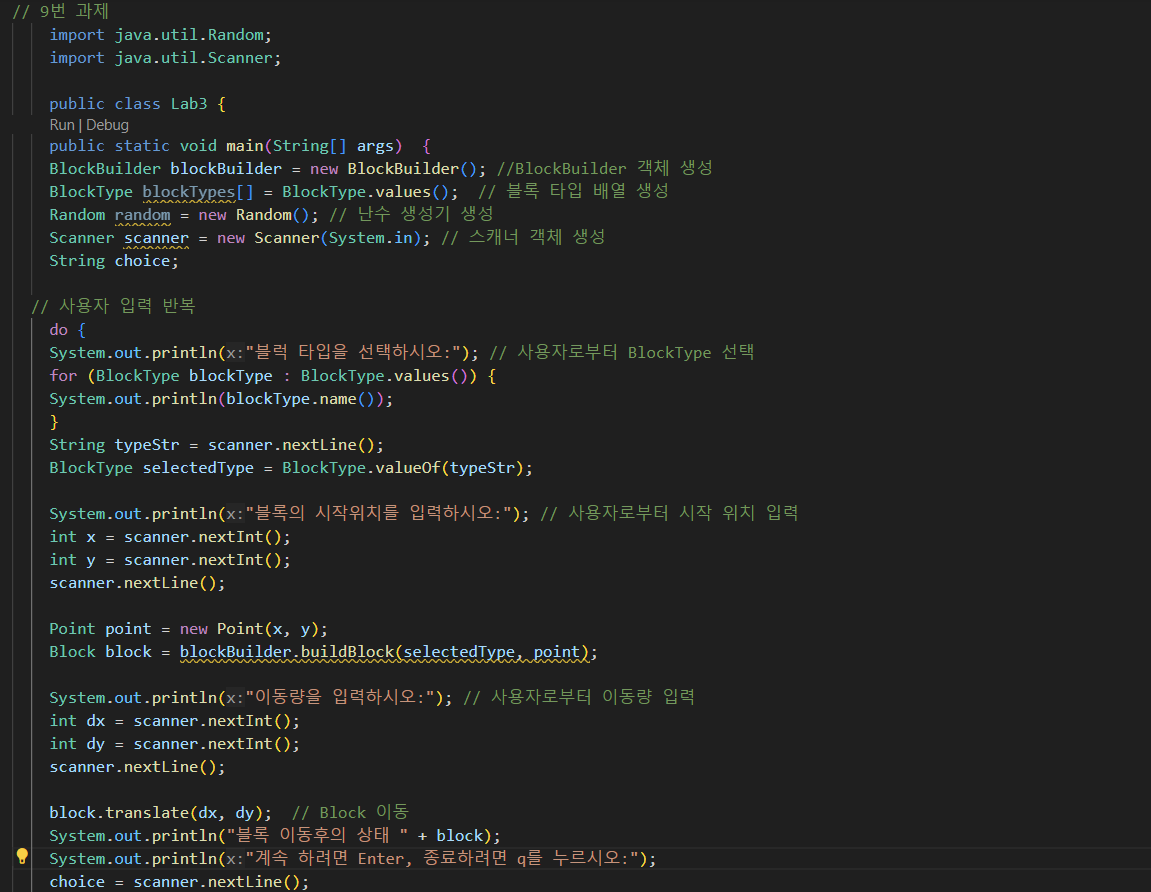




buildPurpleBlock(Point p)메서드는 주어진 시작 점 p를 기준으로 Purple Blue 블록을 생성합니다. Purple 블록은 2x2 크기의 정사각형 4개로 이루어져 있으며, 시작 지점은 빨간점에서 시작한다. 각 정사각형은 시작 위치에서 하나 생성하고 그 다음은 시작 위치에서 X 좌표를 50만큼 이동한 위치에서 생성되고 그 다음은 시작 위치에서 X,Y 좌표를 50만큼 이동한 위치에서 생성되고 그 다음은 X좌표를 100만큼, Y좌표를 50만큼 이동한 위치에서 생성된다.

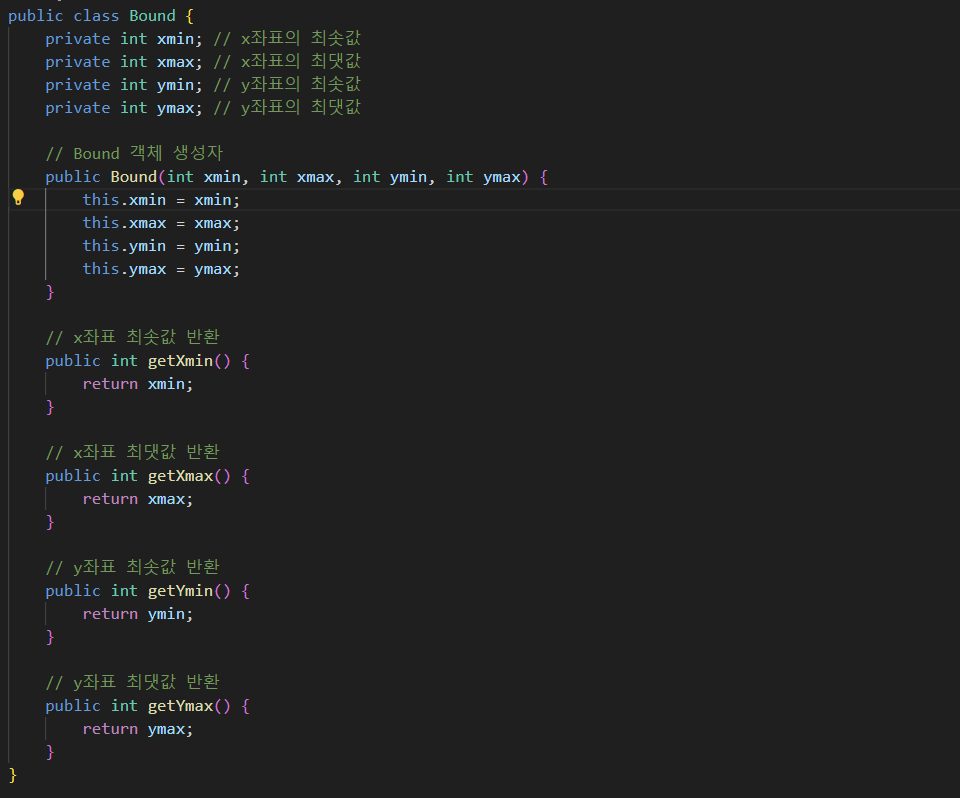
buildGreenBlock(Point p)메서드는 주어진 시작 점 p를 기준으로 Green Blue 블록을 생성합니다. Green 블록은 3x1 크기의 정사각형 4개로 이루어져 있으며, 시작 지점은 빨간점에서 시작한다. 각 정사각형은 시작 위치에서 하나 생성하고 그 다음은 시작 위치에서 X 좌표를 50만큼 이동한 위치에서 생성되고 그 다음은 시작 위치에서 X 좌표를 100만큼 이동한 위치에서 생성되고 그 다음은 X좌표를 100만큼, Y좌표를 50만큼 이동한 위치에서 생성된다.

**9번 과제)**



BlockBuilder blockBuilder = new BlockBuilder();: BlockBuilder 객체를 생성합니다. 이 객체는 블록을 생성하는 데 사용됩니다.BlockType blockTypes[] = BlockType.values();: BlockType 열거형의 모든 값들을 배열로 가져옵니다. 이 배열은 사용자에게 블록 타입을 선택하도록 표시하는 데 사용됩니다.Random random = new Random();: 난수 생성기를 생성합니다. 나중에 필요한 경우 사용됩니다.Scanner scanner = new Scanner(System.in);: 사용자의 입력을 받기 위한 Scanner 객체를 생성합니다.do { ... } while(!choice.equals("q"));: 사용자가 "q"를 입력할 때까지 반복하는 do-while 루프입니다. 사용자가 프로그램을 종료하고 싶을 때 "q"를 입력할 수 있습니다.System.out.println("블럭 타입을 선택하시오:");: 사용자로부터 블록 타입을 선택하라는 메시지를 출력합니다.for (BlockType blockType : BlockType.values()) { System.out.println(blockType.name()); }: 사용자가 선택할 수 있는 모든 블록 타입을 출력합니다.String typeStr = scanner.nextLine();: 사용자의 입력을 읽어와서 선택된 블록 타입을 문자열로 저장합니다.BlockType selectedType = BlockType.valueOf(typeStr);: 사용자가 입력한 문자열을 BlockType 열거형으로 변환합니다.System.out.println("블록의 시작위치를 입력하시오:");: 사용자로부터 블록의 시작 위치를 입력하라는 메시지를 출력합니다.int x = scanner.nextInt(); int y = scanner.nextInt(); scanner.nextLine();: 사용자로부터 x좌표와 y좌표를 읽어옵니다.Point point = new Point(x, y);: 입력된 x좌표와 y좌표를 사용하여 Point 객체를 생성합니다.Block block = blockBuilder.buildBlock(selectedType, point);: 선택된 블록 타입과 시작 위치를 사용하여 블록을 생성합니다. sstem.out.println("이동량을 입력하시오:");: 사용자로부터 블록을 이동시킬 양을 입력하라는 메시지를 출력합니다.int dx = scanner.nextInt(); int dy = scanner.nextInt(); scanner.nextLine();: 사용자로부터 x축과 y축으로 이동할 양을 읽어옵니다.block.translate(dx, dy);: 입력된 이동량에 따라 블록을 이동시킵니다.System.out.println("블록 이동후의 상태 " + block);: 블록을 이동시킨 후의 상태를 출력합니다.System.out.println("계속 하려면 Enter, 종료하려면 q를 누르시오:");: 사용자에게 계속할지 종료할지 물어봅니다.

**10번 과제)**



private int xmin; 는 바운딩 박스의 x좌표의 최솟값을 나타내는 변수이다.

private int xmax; 는 바운딩 박스의 x좌표의 최댓값을 나타내는 변수이다.

private int ymin; 는 바운딩 박스의 y좌표의 최솟값을 나타내는 변수이다.

private int ymax; 는 바운딩 박스의 y좌표의 최댓값을 나타내는 변수이다.

이 클래스는 주어진 네 개의 값을 받아서 바운딩 박스의 속성을 설정하는 생성자를 가지고 있다.

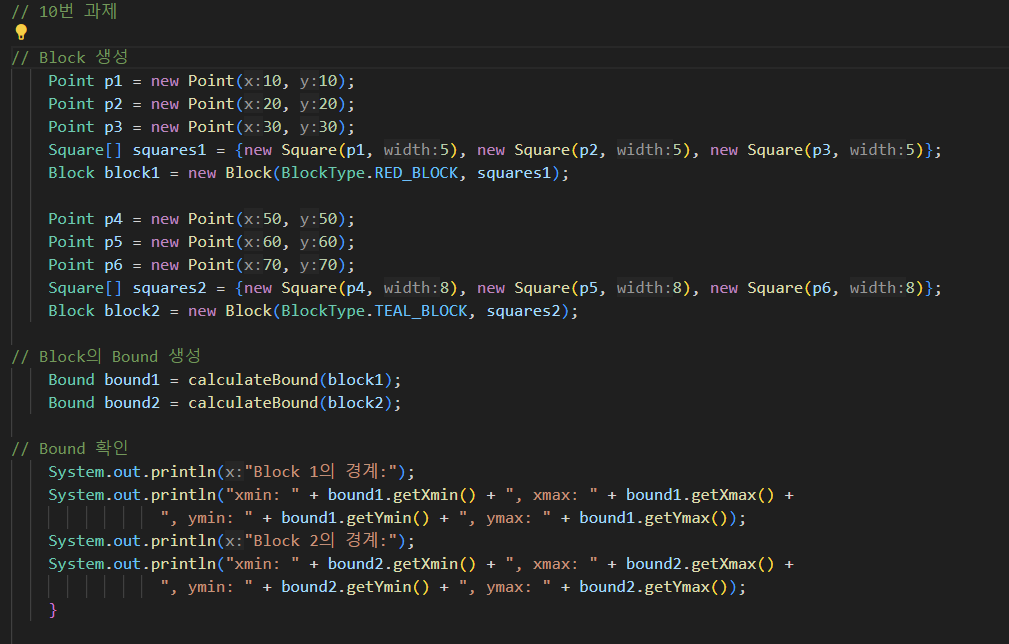
또한, x좌표와 y좌표의 최솟값과 최댓값을 반환하는 메소드를 가지고 있다.

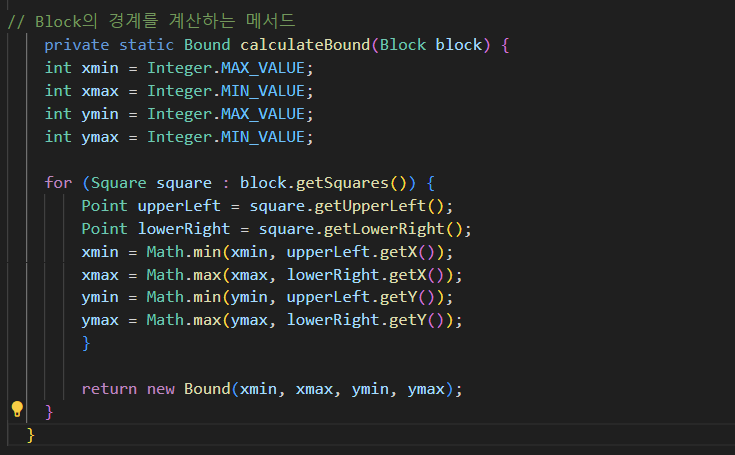
public int getXmin(); 는 x좌표의 최솟값을 반환합니다.

public int getXmax(); 는 x좌표의 최댓값을 반환합니다.

public int getYmin(): 는 y좌표의 최솟값을 반환합니다.

public int getYmax(); 는 y좌표의 최댓값을 반환합니다.





point 객체를 사용하여 각각의 정사각형 좌상단 점을 설정한다. 각 정사각형의 크기는 5이고 이러한 정사각형 배열들을 담은 Square 배열을 생성한다. 이 배열과 “RED\_BLOCK” 타입을 사용하여 첫번째 블록을 생성한다 동일한 방식으로 두번째 블록을 생성하는데 이 블록의 타입은 “TEAL\_BLOCK” 이다. 다음 calculateBound 메소드는 블록을 인자로 받아서 그 블록의 바운딩 박스를 계산한다. x의 최소,최대값 그리고 y의 최소,최대값을 초기화하고 각각의 정사각형에 대해 해당 정사각형의 좌상단 점과 우하단 점의 x,y 값을 비교하여 바운딩 박스의 최솟값과 최댓값을 갱신한다. 마지막으로 새로운 Bound 객체를 생성하여 이를 반환한다. 그 후 두 블록에 대해 각각의 바운딩 박스를 출력하고 이를 통해 각 블록의 바운딩 박스의 최솟값