

# 컴퓨터 프로그래밍1

## 실습 9주차

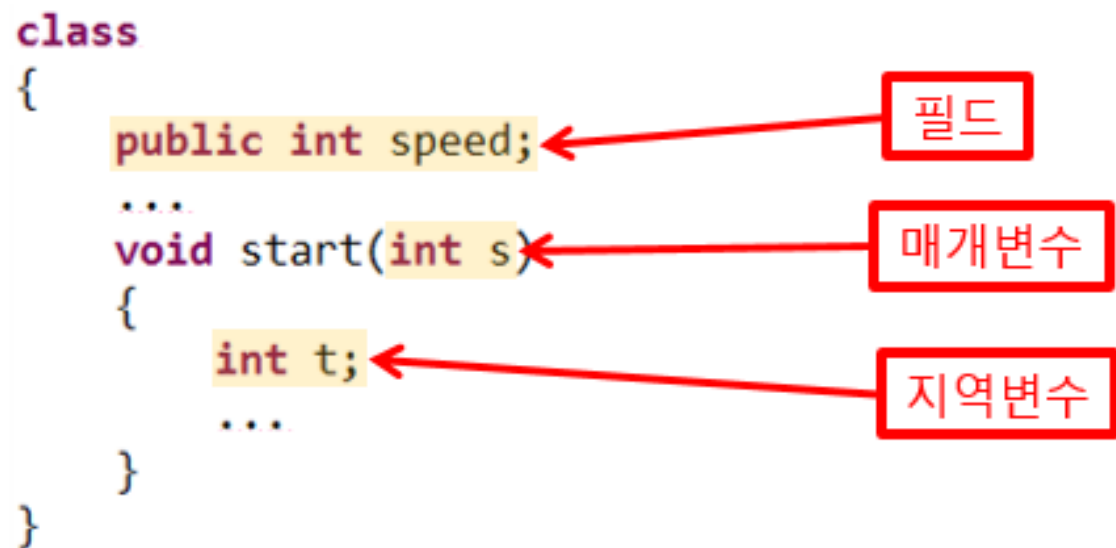
- 클래스 -

# 필드와 메소드 (1/10)

## 필드

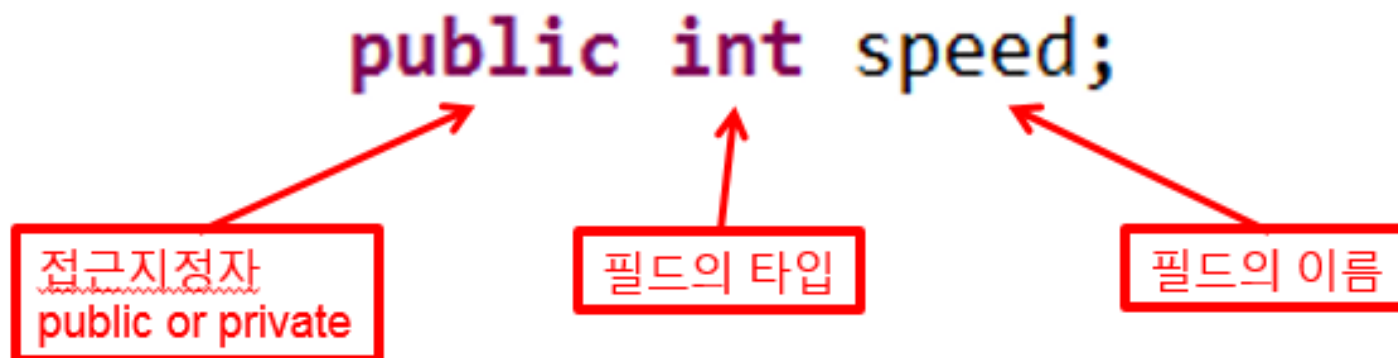
### ■ 변수의 종류

```
class
{
    public int speed;
    ...
    void start(int s)
    {
        int t;
        ...
    }
}
```



### ■ 필드 선언 형식

public int speed;



# 필드와 메소드 (2/10)

## 필드

- 필드는 선언과 동시에 초기화 될 수 있다

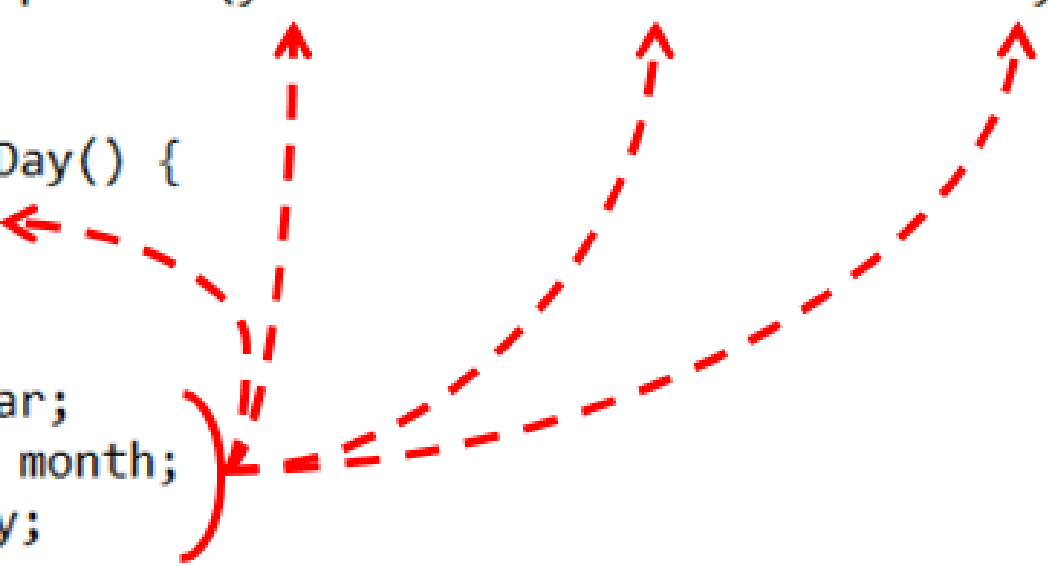
```
public class Classroom
{
    public static int capacity = 60;
    private boolean use = false;
}
```

# 필드와 메소드 (3/10)

## 필드

- 필드는 정의된 위치와 상관없이 클래스 안 어디서든 사용이 가능하다

```
public class Date {  
  
    public void printDate() {  
        System.out.println(year + "." + month + "." + day);  
    }  
  
    public int getDay() {  
        return day;  
    }  
    // 필드 선언  
    private int year;  
    private String month;  
    private int day;  
}
```



# 필드와 메소드 (4/10)

## 접근자와 설정자

- 객체 지향 방법의 핵심은 구현의 세부 사항을 감추는 것
- 필드에 직접 접근하기 위한 접근자와 설정자를 사용

```
class Car {  
    private String color; // 색상  
    private int speed; // 속도  
    private int gear; // 기어  
  
    public String getColor() {  
        return color;  
    }  
    public void setColor(String c) {  
        color = c;  
    }  
    public int getSpeed() {  
        return speed;  
    }  
    public void setSpeed(int s) {  
        speed = s;  
    }  
    public int getGear() {  
        return gear;  
    }  
    public void setGear(int g) {  
        gear = g;  
    }  
}
```

필드가 모두 private로 선언되었기 때문에 클래스 내부에서만 사용이 가능하다.

# 필드와 메소드 (5/10)

## 접근자와 설정자

- 클래스 사용 시 클래스 내부에 있는 필드에 접근하기 위해 접근자와 설정자를 사용

```
public class CarTest2 {  
    public static void main(String[] args) {  
        // 객체 생성  
        Car myCar = new Car();  
        myCar.setColor("red");  
        myCar.setSpeed(100);  
        myCar.setGear(1);  
  
        System.out.println("현재 자동차의 색상은 " + myCar.getColor());  
        System.out.println("현재 자동차의 속도는 " + myCar.getSpeed());  
        System.out.println("현재 자동차의 기어는 " + myCar.getGear());  
    }  
}
```

설정자 메소드 호출

접근자 메소드 호출

# 필드와 메소드 (6/10)

## 지역 변수

- 지역 변수는 필드와 달리, 메소드 안에서만 사용

```
class Box {  
    int width=0, length=0, height=0;  
  
    public int getVolume()  
    {  
        int volume;  
        volume = width*length*height;  
        return volume;  
    }  
}
```

필드: 클래스 전체에서 사용 가능

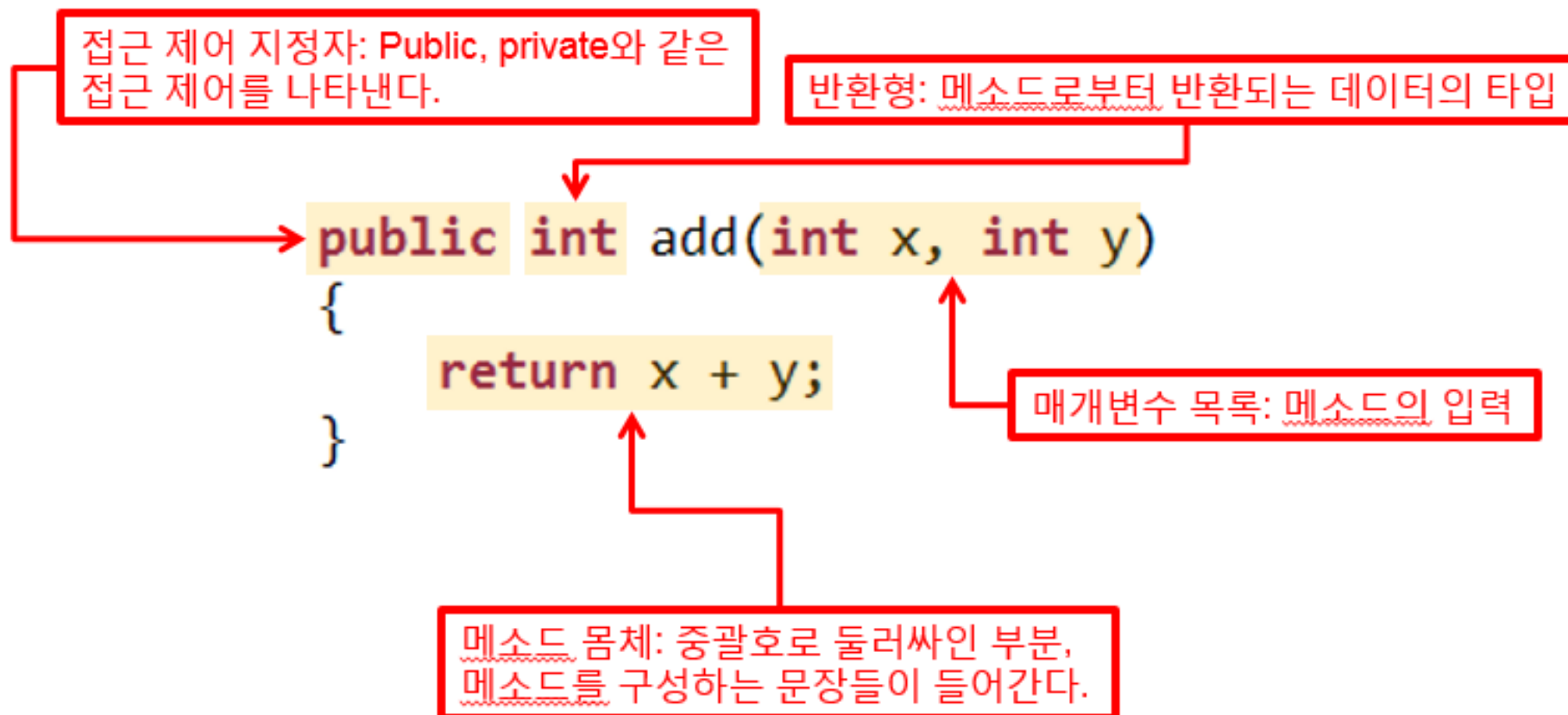
지역변수

지역변수 volume의  
사용범위

# 필드와 메소드 (7/10)

## 메소드 형식

- 메소드의 선언은 다음과 같은 형식을 가진다





# 필드와 메소드 (8/10)

## 메소드의 반환값

- 메소드는 하나의 반환값을 가질 수 있으며, 메소드 호출로 값을 받는 경우 반환값의 타입과 일치 해야 한다.

```
Car myCar = new Car();  
int value = myCar.getSpeed();
```

```
class Car {  
    private int speed = 0; // 속도  
  
    public int getSpeed(){ return speed; }  
    public void setSpeed(int s){ speed = s; }  
}
```

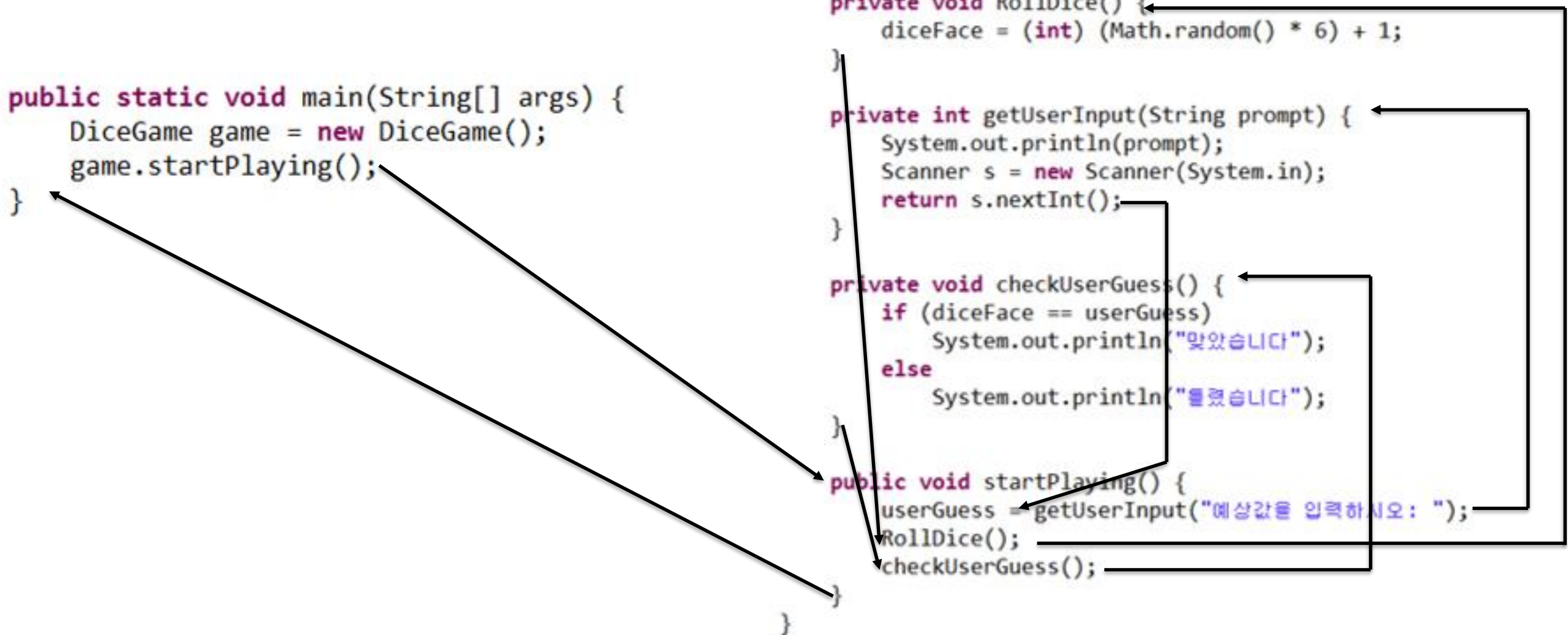
# 필드와 메소드 (9/10)

## 메소드 호출 과정

- 메소드를 호출하면 실행하고 있던 메소드는 잠시 중단되고, 호출된 메소드가 실행된다. 호출된 메소드가 종료하면 잠시 중단되었던 원래의 메소드가 실행을 재개한다

```
public static void main(String[] args) {  
    DiceGame game = new DiceGame();  
    game.startPlaying();  
}
```

```
class DiceGame {  
  
    int diceFace;  
    int userGuess;  
  
    private void RollDice() {  
        diceFace = (int) (Math.random() * 6) + 1;  
    }  
  
    private int getUserInput(String prompt) {  
        System.out.println(prompt);  
        Scanner s = new Scanner(System.in);  
        return s.nextInt();  
    }  
  
    private void checkUserGuess() {  
        if (diceFace == userGuess)  
            System.out.println("맞았습니다");  
        else  
            System.out.println("틀렸습니다");  
    }  
  
    public void startPlaying() {  
        userGuess = getUserInput("예상값을 입력하십시오: ");  
        RollDice();  
        checkUserGuess();  
    }  
}
```



# 필드와 메소드 (10/10)

## 가변길이 인수

- 가변길이 인수는 메소드로 전달될 인수의 정확한 개수를 알 수 없을 때 사용된다.

```
class Test {  
    void sub(int... v) {  
        System.out.println("인수의 개수 : " + v.length);  
  
        for (int x : v)  
            System.out.print(x + " ");  
  
        System.out.println();  
    }  
}  
  
public class VarArgsTest {  
  
    public static void main(String args[]) {  
        Test c = new Test();  
        c.sub(1);  
        c.sub(2, 3, 4, 5, 6);  
        c.sub();  
    }  
}
```

# 실습

---

## 책에 대한 클래스 작성

- 책을 나타내는 클래스 작성 (class Book)
- 책은 제목, 저자, 가격의 필드를 가짐
- 각 필드에 대한 접근자, 설정자 메소드를 작성

## 책의 클래스를 테스트할 main 함수 작성

- BookTest라는 이름의 클래스를 생성하고 main 함수를 작성
- main 함수에 Book 클래스의 객체를 2개 생성하고 설정자를 이용하여 책의 제목, 저자, 가격을 설정 (임의로 작성)
- 생성한 Book 객체들의 정보를 Book 클래스의 메소드를 이용하여 출력

# 과제 (1/2)

---

## 상자에 대한 클래스 작성

- 상자를 나타내는 클래스 작성 (class Box)
- 상자는 가로, 세로, 높이를 가지며 이를 필드로 표현
- 각 필드에 대한 접근자, 설정자 메소드를 작성
- 상자의 부피를 계산하는 int getVolume() 메소드 작성 (메소드 내부에서 접근자 사용)
- 상자의 필드 값을 출력하는 void print() 메소드 작성 (메소드 내부에서 접근자 사용)

## 상자 클래스를 테스트할 main 함수 작성

- BoxTest라는 이름의 클래스를 생성하고 main 함수를 작성
- main 함수에 Box 클래스의 객체를 생성하고 설정자를 이용하여 가로,세로,높이가 10, 20, 50인 Box 객체 생성
- 생성한 Box 객체의 필드값, 부피를 Box 클래스의 메소드를 이용하여 화면에 출력

# 과제 (2/2)

---

## 교재 p.195 5번 문제

5. 직원을 나타내는 Employee 클래스를 작성하여 보자. 직원은 이름, 전화 번호, 연봉을 필드로 가지고 있다. 각 필드에 대하여 접근자와 설정자를 작성하라. EmployeeTest 클래스를 작성하여서 Employee 객체를 생성하고 테스트 하라.

# Q&A

---



수고하셨습니다