

Java study

김관석 최정윤 오다건



CONTENTS

01

Java란?

02

자바 언어 설명

03

자바 클래스

04

약수의 합 구하기



01

Java!!

01. 자바 알아보기

<자바의 특징>

쉬운 언어이다.

C와 C++의 언어의 문법을 기본으로 차용하여 개발된 언어

C와 C++에 비해 쉬운 언어

플랫폼에 독립적이다.

JVM()만 있으면 윈도우, 리눅스, 맥 등 어떤 플랫폼에서도 실행 가능

객체지향 언어이다.

메모리 관리를 자동으로 해준다.

객체지향 – 데이터를 추상화시켜 상태 행위를 가진 객체를 만들고 그 객체들 간의 유기적 상호작용을 통해 로직 구성하는 프로그래밍 방법

<http://www.oracle.com/technetwork/java/index.html>

01. 자바 개발순서



코드 작성



코드 컴파일



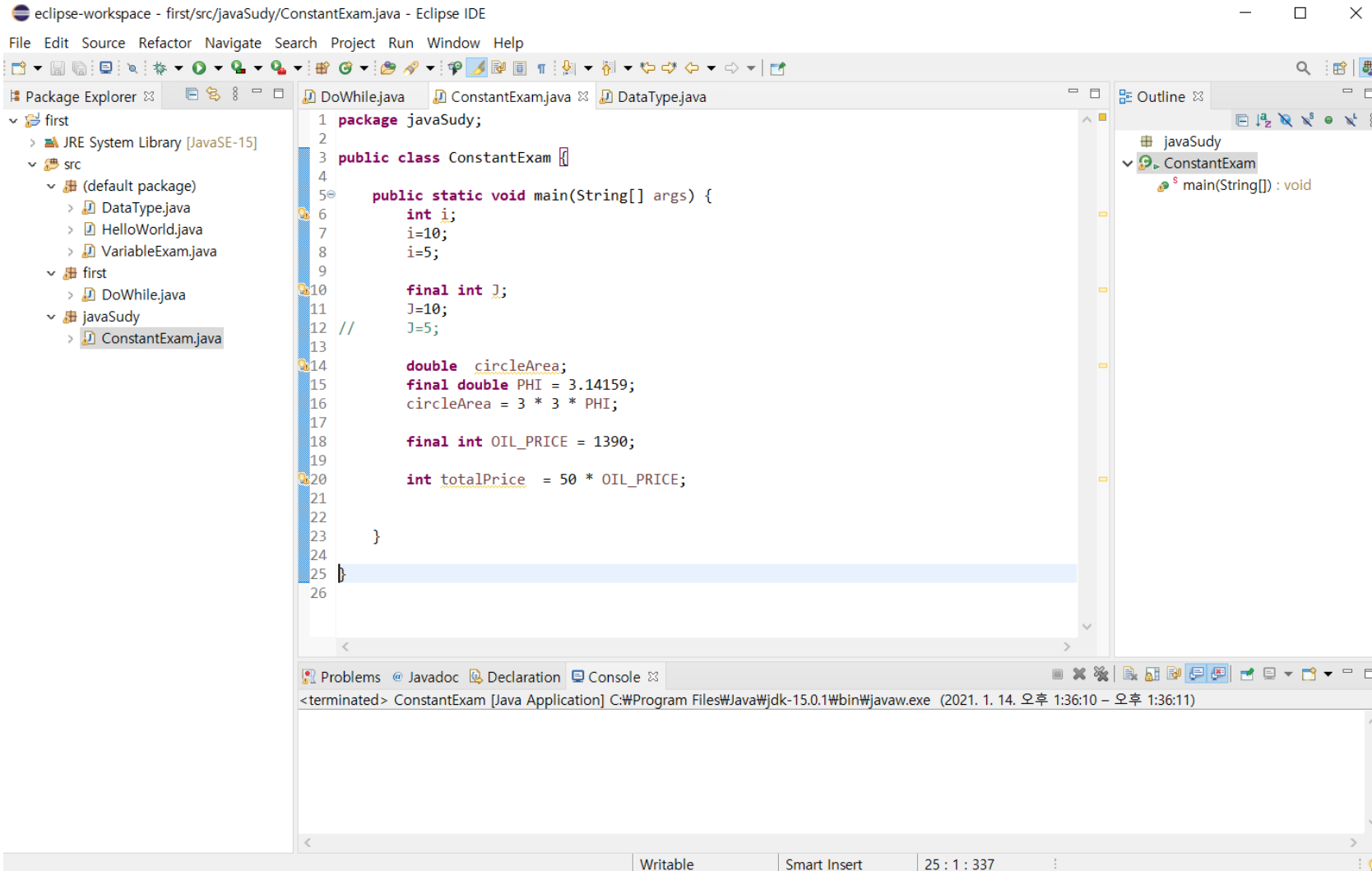
JVM으로 실행

컴파일: 특정 프로그래밍 언어로 쓰여 있는 문서를 다른 언어로 옮기는 프로그램

02

자바 code

02. 변수 생성



변수 설정 방법

변수의 타입 변수 이름 = 값;

상수 설정 방법

final 변수의 타입 변수 이름 = 값;

값이 변하면 위험한 경우 상수 사용

02. 기본형 타입

<기본형>

가장 기본이 되는 데이터 타입; 실수형, 문자형, 불린형

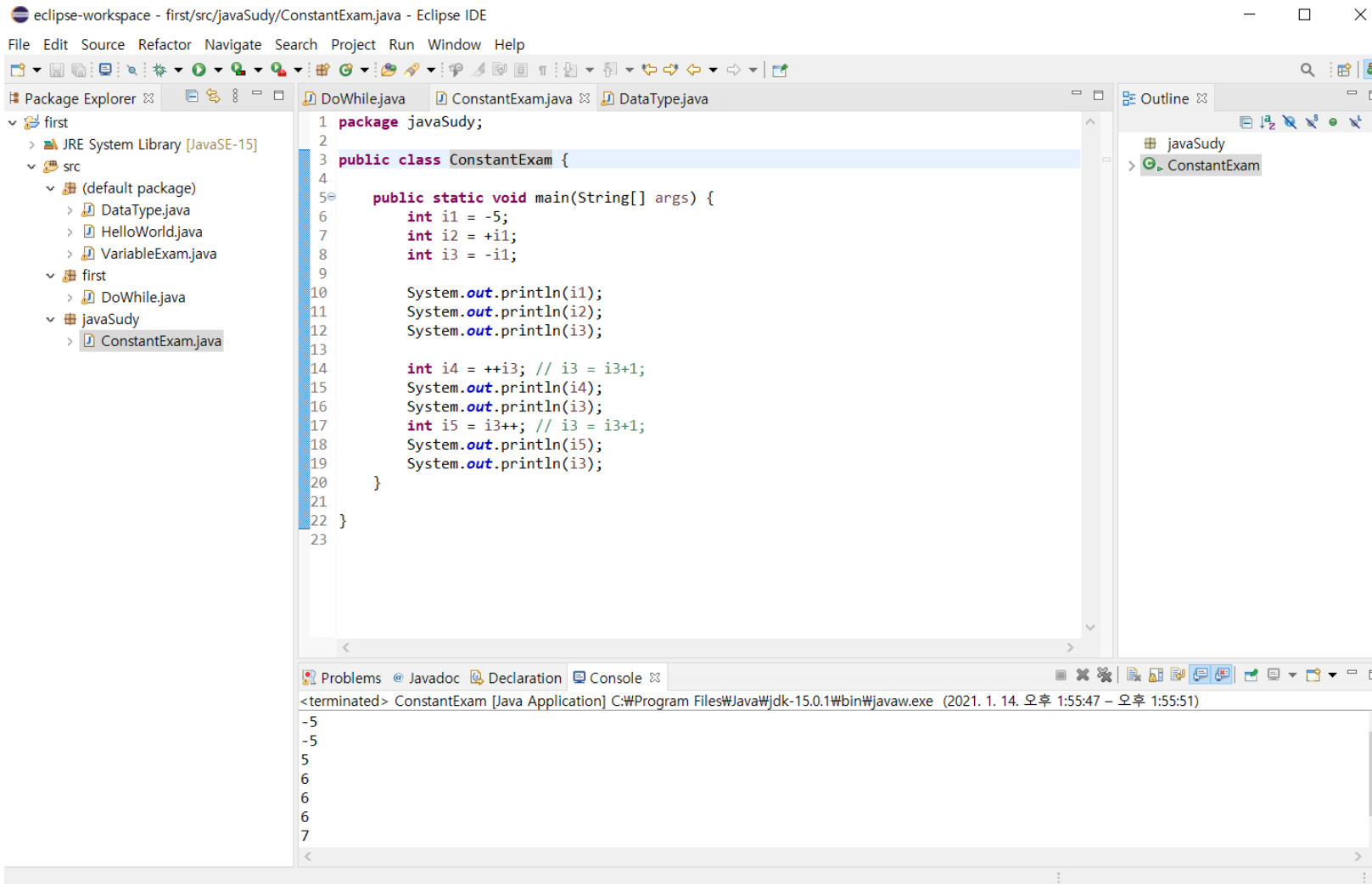
사용법

```
boolean isFun = true;  
char c = 'f'; // 문자 하나  
int x = 59;  
long big = 3456789L;  
float f = 32.5f  
double d = 23.34
```

자료형	키워드	크기	표현 범위	사용 예
논리형	boolean	1byte	true or false (0과 1이 아니다)	boolean isFun = true;
문자형	char	2byte	0~65, 535	char c = 'f';
정수형	byte	1byte	-128 ~ 127	byte b = 89;
	short	2byte	-32,768 ~ 32,767	short s = 32760;
	char	2byte	0 ~ 65, 535	char c = 64;
	int	4byte	-2147483648 : 2147483647	int x = 59; int z = x;
	long	8byte	...	long big = 3456789L;
실수형	float	4byte	-3.4E038 ~ 3.4E038	float f = 32.5f
	double	8byte	-1.7E308 ~ 1.7E308	double d = 23.34

02. 산술 연산자

부호(+, -), 증감(++, --), +, -, *, /, %



```
1 package javaSudy;
2
3 public class ConstantExam {
4
5     public static void main(String[] args) {
6         int i1 = -5;
7         int i2 = +i1;
8         int i3 = -i1;
9
10        System.out.println(i1);
11        System.out.println(i2);
12        System.out.println(i3);
13
14        int i4 = ++i3; // i3 = i3+1;
15        System.out.println(i4);
16        System.out.println(i3);
17        int i5 = i3++; // i3 = i3+1;
18        System.out.println(i5);
19        System.out.println(i3);
20    }
21
22 }
23
```

Console Output:

```
<terminated> ConstantExam [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (2021. 1. 14. 오후 1:55:47 - 오후 1:55:51)
-5
-5
5
6
6
6
7
```

int i4 = ++i3;
-> i3 = i3 + 1 / i4 = i3

int i5 = i3++;
-> i5 = i3 / i3 = i3 + 1

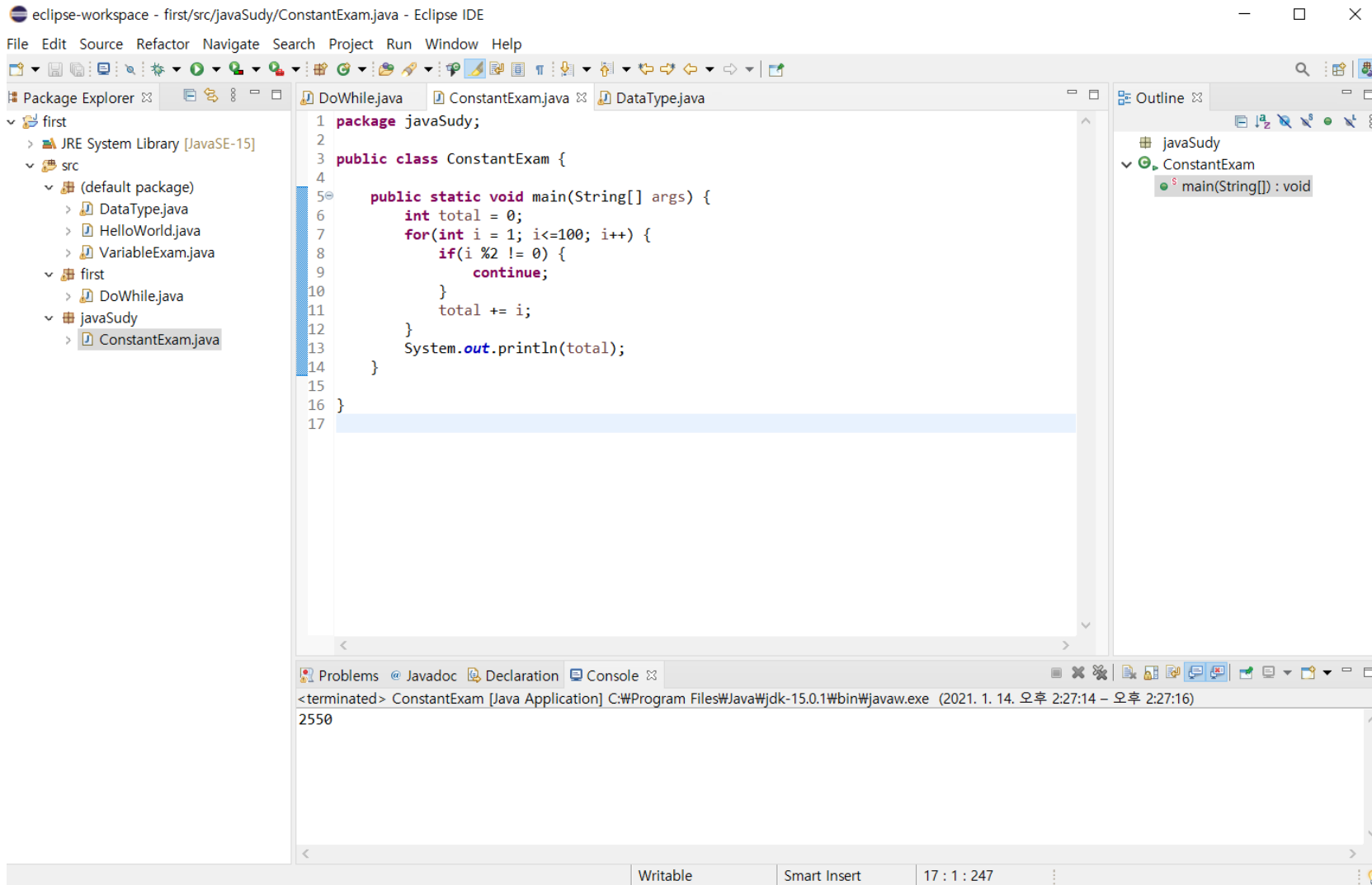
02. if 문

```
1 package javaSudy;
2
3 public class ConstantExam {
4
5     public static void main(String[] args) {
6         int x = 50;
7         int y = 60;
8
9         if(x == y) {
10             System.out.println("x는 y와 같습니다.");
11         } else {
12             System.out.println("x와 y는 다릅니다.");
13         }
14     }
15 }
16 }
17 }
```

Console Output:
<terminated> ConstantExam [Java Application] C:\Program Files\Java\jdk-15.0.1\bin\javaw.exe (2021. 1. 14. 오후 2:17:50 - 오후 2:17:52)
x와 y는 다릅니다.

```
if(조건식){
    실행문;
} else if(조건식){
    실행문;
} else{
    실행문;
}
```

02. for 문



for(초기화 식; 조건식; 증감식){
 실행문;
 실행문;
}

continue;
- Continue 아래의 실행문들은 실행하지 않고 다시 for loop을 돌도록 설정

break;
- 그 즉시 for문을 종료하고 빠져나옴

03

자바의 클래스

03. 클래스

```
package
public class Car{ }
↳ class : 객체를 만들기 위한 일종의 틀. (자동차 틀)
```

참조 타입

```
Car c1 = new Car();
String str = new String("Hello");
```

new 연산자 뒤에 나오는 생성자를 이용해 메모리에 객체 만들라
만들어진 객체를 참조하는 변수. 메모리에 만들어진 객체 = **인스턴스**.
주요하다.

int, float, long... String 클래스 기본형 타입 아닌
↳ new 두(세)이길 이용해 객체를 메모리에 올려줌. ↳ 인스턴스
String 클래스를 참조하는 변수 생성자.
String str = "hello"; String 클래스는 new 없이도 가능
이 문자열이 상수가 저장되는 영역에 저장: 변화 없음.

class 안

↳ **field**; 자동자가 갖고 있는 속성

```
public class Car{
    String name;
    int number;
}
```

Car 클래스 인스턴스화

```
Car c1 = new Car();
c1.name = "소방차";
System.out.println(c1.name);
```

↳ **메소드**: 자동자의 행동; 전진, 후진...

```
public void method1(){
    System.out.println("method 1 실행");
}
```

return 값이 없을 때 / 있으면 int, String 이런 거 사용 가능

Scope

```
public class ScopeExam{
```

```
    int globalScope = 10;
    static int staticVal = 7;
    public void scopeTest(int value){
        int localScope = 10;
    }
```

```
    public static void main(String[] args){
```

```
        System.out.println(staticVal);
    }
```

Static

- 같은 클래스 내에 있어도 해당 변수 사용 X
- static 한 field 나 static 한 메소드는 Class가 인스턴스화 되기 전에도 사용 가능
- static 한 변수는 공유된다. (인스턴스 여러개 생성해도 static한 변수는 하나!)

```
ValableScopeExam v1 = new ValableScopeExam();
ValableScopeExam v2 = new ValableScopeExam();
v1.globalScope = 20;
v2.globalScope = 30;
```

```
System.out.println(v1.globalScope); //20 이 출력된다.
System.out.println(v2.globalScope); //30 이 출력된다.
```

```
v1.staticVal = 10;
v2.staticVal = 20;
```

```
System.out.println(v1.staticVal); //20 이 출력된다.
System.out.println(v2.staticVal); //20 이 출력된다.
```

04

약수의 합 구하기

04. 약수의 합 구하기

<Question>

자연수 n 을 입력받아
 n 의 약수를 모두 더한 값을 리턴하는 함수, `solution`을 완성해주세요.

<제한 사항>

n 은 0이상 30000이하인 자연수입니다.

<example>

n : 12, return 28

04. 약수의 합 구하기

```
1  class Solution {  
2      public int solution(int n) {  
3          int answer = 0;  
4          for(int i=1; i<=n; i++){  
5              if (n%i == 0){  
6                  answer = answer + i ;  
7              }  
8          }  
9          return answer;  
10     }  
11 }
```

1. Solution 이라는 클래스 생성

04. 약수의 합 구하기

```
1  class Solution {  
2      public int solution(int n) {  
3          int answer = 0;  
4          for(int i=1; i<=n; i++){  
5              if (n%i == 0){  
6                  answer = answer + i ;  
7              }  
8          }  
9          return answer;  
10     }  
11 }
```

1. Solution 이라는 클래스 생성

2. 정수 타입의 n 받고, 리턴값도 정수 타입.

Public (return type)class name(매개변수 type)

04. 약수의 합 구하기

```
1  class Solution {
2      public int solution(int n) {
3          int answer = 0;
4          for(int i=1; i<=n; i++){
5              if (n%i == 0){
6                  answer = answer + i ;
7              }
8          }
9          return answer;
10     }
11 }
```

1. Solution 이라는 클래스 생성
2. 정수 타입의 n 받고, 리턴값도 정수 타입.
3. 약수의 '합'을 구하는 것이므로
리턴할 answer= 0 으로 초기화.

04. 약수의 합 구하기

```
1  class Solution {
2      public int solution(int n) {
3          int answer = 0;
4          for(int i=1; i<=n; i++){
5              if (n%i == 0){
6                  answer = answer + i ;
7              }
8          }
9          return answer;
10     }
11 }
```

1. Solution 이라는 클래스 생성
2. 정수 타입의 n 받고, 리턴값도 정수 타입.
3. 약수의 '합'을 구하는 것이므로 리턴할 answer= 0 으로 초기화.
4. 어느 수의 약수를 구하기 위해선 n이라는 수 만큼 반복해야함.
for 문 생성!

Ex) 4의 약수는 1,2,4 이므로 4까지 하나하나 반복해보아야함.

04. 약수의 합 구하기



```
For (int i =1; i가 n 이 될 때까지 반복 ; i가 1씩 증가){  
    만약 (n이라는 숫자가 i로 나누어 떨어진다면){  
        answer 에 i 만큼 더한다;  
    }  
}
```



```
4      for(int i=1; i<=n; i++){  
5          if (n%i == 0){  
6              answer = answer + i ;  
7              }  
8      }
```

04. 약수의 합 구하기

```
1  class Solution {
2      public int solution(int n) {
3          int answer = 0;
4          for(int i=1; i<=n; i++){
5              if (n%i == 0){
6                  answer = answer + i ;
7              }
8          }
9          return answer;
10     }
11 }
```

1. Solution 이라는 클래스 생성
2. 정수 타입의 n 받고, 리턴값도 정수 타입.
3. 약수의 '합'을 구하는 것이므로 리턴할 answer= 0 으로 초기화.
4. 어느 수의 약수를 구하기 위해선 n이라는 수 만큼 반복해야함. for 문 생성!

Ex) 4의 약수는 1,2,4 이므로 4까지 하나하나 반복해보아야함.

5. 마지막으로 **answer** 를 리턴해준다.



Thank you

