Professional Coding Specialist

COS Pro JAVA 1 급

1 강. 기초 문법 정리 1

1. 설치
2. 변수
3. 배열

과정 소개

COS Pro(Coding Specialist Professional) 시험은 요구사항을 분석하여 프로그램을 설계, 구현하는 능력과 주어진 프로그램을 디버깅하여 수정하는 능력을 평가하는 자격증 시험이며, COS Pro 1 급 자격증은 높은 수준의 프로그래밍 능력을 증명할 수 있다. 이번 시간에는 COS Pro JAVA 1 급 시험 대비를 위한 모의고사를 풀어보기 전에 알아야 하는 JAVA 문법을 정리하는 첫 번째 시간으로 학습을 위한 환경 준비에 대한 간략한 소개와 JAVA 에서 변수를 사용하여 데이터를 다룰 때의 유의 사항, 배열에 대해 정리해 본다.

학습 목차

학습 목표

1. COS Pro JAVA 1 급 시험을 대비하는 학습을 진행하기 전에 JDK 를 설치하고, JAVA 통합 개발환경인 Eclipse 를 설치하여 JAVA 프로그램을 작성할 수 있다.
2. JAVA 에서 기본적으로 사용하는 자료형의 특징과 그 범위를 알고, 해당 자료형을 사용하여 변수를 생성하고 필요에 따라 형 변환(type conversion)을 할 수 있다.
3. JAVA 에서 사용하는 연산자의 기능을 이해하고 프로그래밍에 활용할 수 있다.
4. JAVA 에서 배열을 선언하고 사용하는 기본 문법을 정리할 수 있다.

1 설치

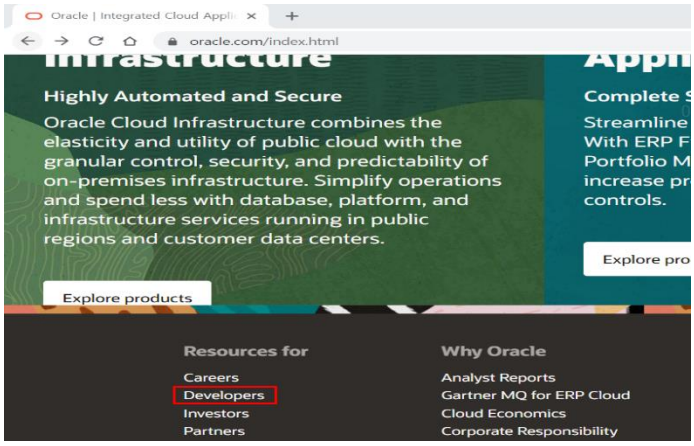
1. JDK 설치

1) JDK 설치 파일 다운로드

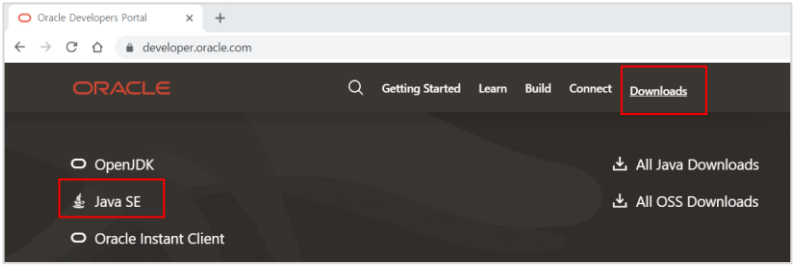
- JDK(Java Development Kit) : JRE + 개발에 필요한 실행 파일
- JRE(Java Runtime Environment) : JVM(Java Virtual Machine) + 클래스 라이브러리(JAVA API)
- https://www.oracle.com/java/technologies/downloads/ JDK17 을 다운로드
  (버전이 계속 업데이트 되므로 학습 시 최신 버전 설치 필요)

① https://www.oracle.com/ 접속

② **사이트 하단의 [Developers] 클릭**

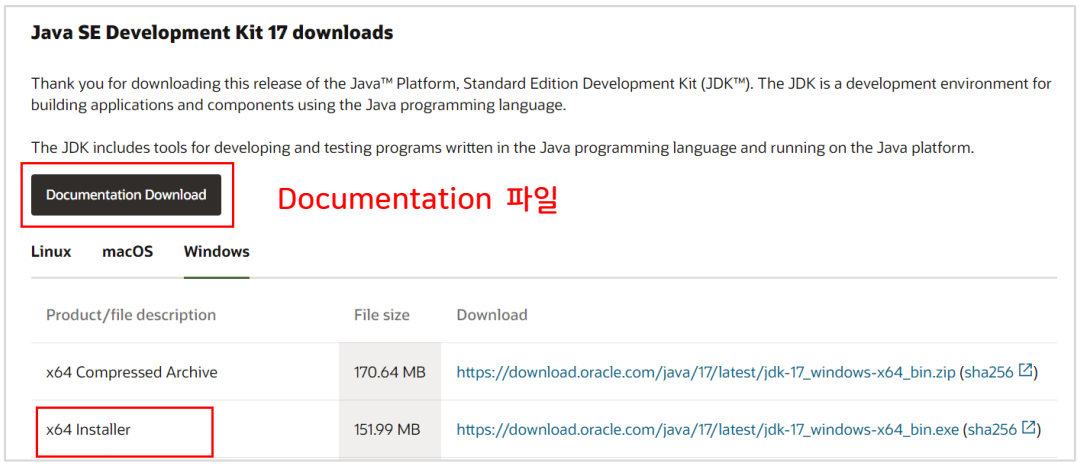


③ [Downloads] 클릭 - [Java SE] 클릭



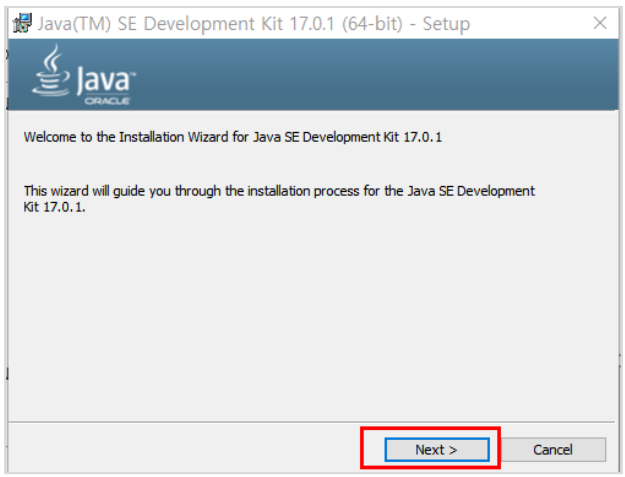
④ 최신 버전 설치 파일 및 Documentation 파일 다운로드
(강의에서는 17 버전을 다운로드 받았으며, 설치하는 시점을 기준으로 최신 버전 다운로드 추천)
본인의 OS 버전에 맞는 파일을 다운받아 설치

- **[설치 파일]**은 jdk-17_windows-x64_bin.exe 처럼 실행 파일이므로 더블 클릭하면 설치 시작
- **[Documentaion 파일]**은 jdk-17.0.1_doc-all.zip 파일처럼 압축을 풀고 해당 폴더의 index 파일을 열면 API 에 대한 설명을 볼 수 있음

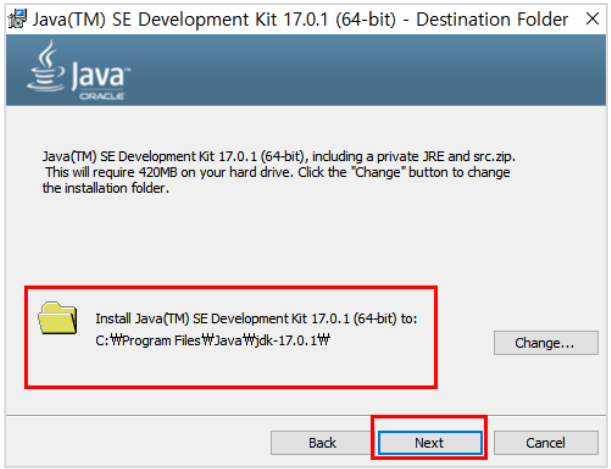


2) JDK 설치

① JDK 설치 파일 (jdk-17_windows-x64_bin.exe)을 실행한 후 나타나는 안내 창에서 [Next] 클릭



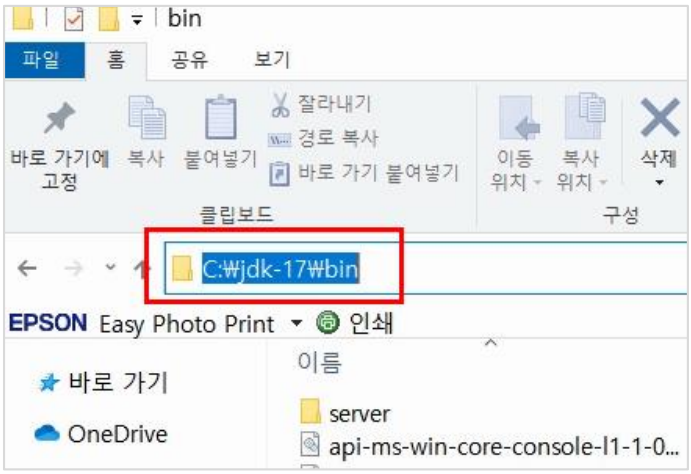
② [Next] 클릭하면 설치 시작 (Next 클릭하기 전에 JDK가 설치되는 폴더 변경 가능)



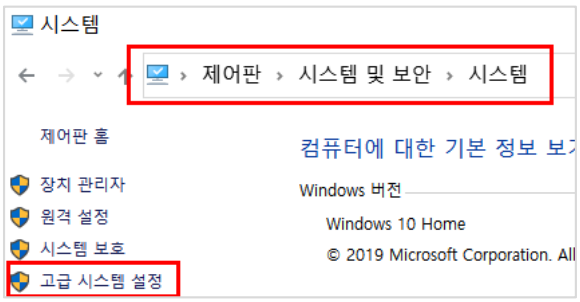
3) PATH 추가

PATH 에 JDK 설치 경로를 추가하여 Eclipse 이외의 다른 개발 도구가 JAVA 프로그램 개발에 사용될 경우를 대비 → 어떤 위치에서도 JDK 를 사용할 수 있도록 함

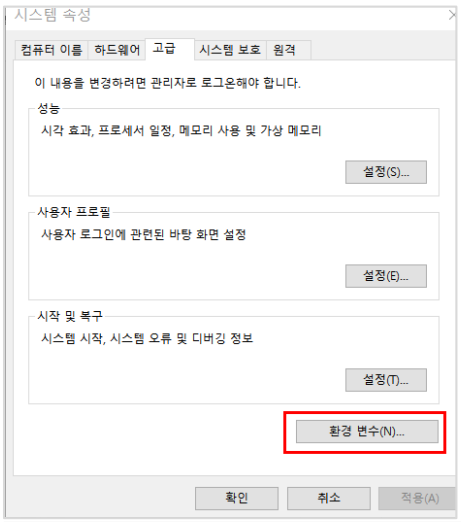
① 탐색기를 이용하여 JDK 가 설치된 폴더 안의 bin 폴더까지 들어간 후 폴더 경로를 마우스로 클릭해서 복사



② [제어판] - [시스템 및 보안] - [시스템]에서 [고급 시스템 설정] 클릭

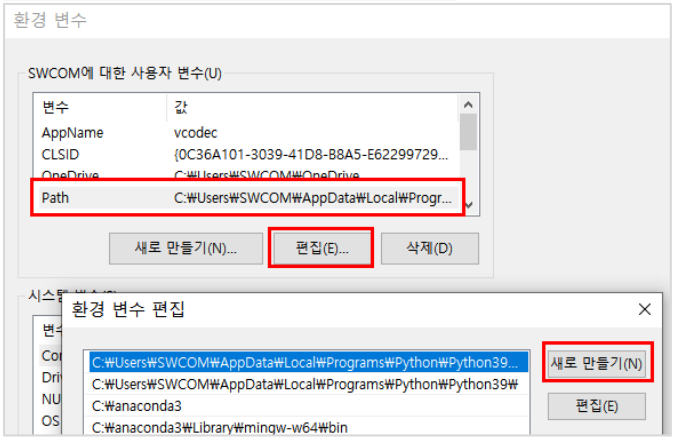


③ [시스템 속성] 창의 [고급] 탭에서 [환경 변수] 선택

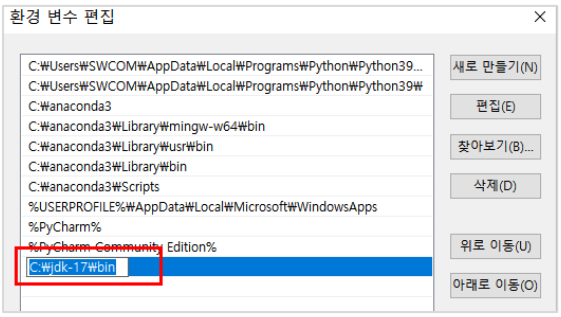


④ [환경 변수]창에서 [시스템 변수] 중 'Path' 를 선택하고 [편집] 클릭

⑤ [환경 변수 편집] 창에서 [새로 만들기] 클릭



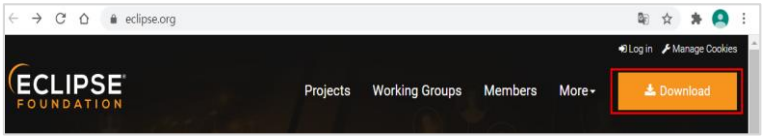
⑥ 새로 추가된 입력란에 탐색기에서 복사한 jdk 의 bin 경로를 입력하고 [확인] 클릭

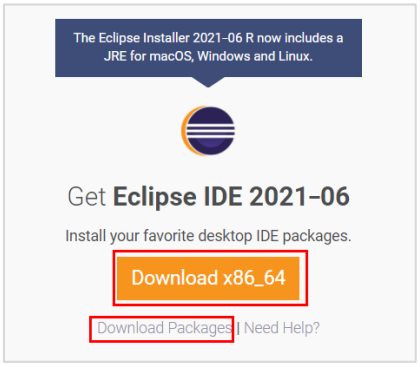


2. Eclipse 시작

1) Eclipse 설치 파일 다운로드

- https://www.eclipse.org/downloads/ 에 있는 최신 설치 파일 다운로드

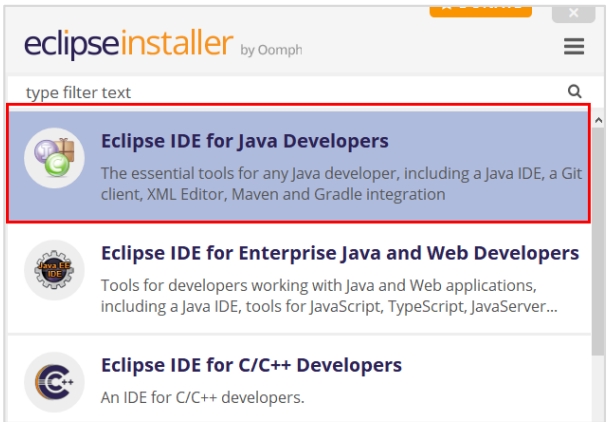




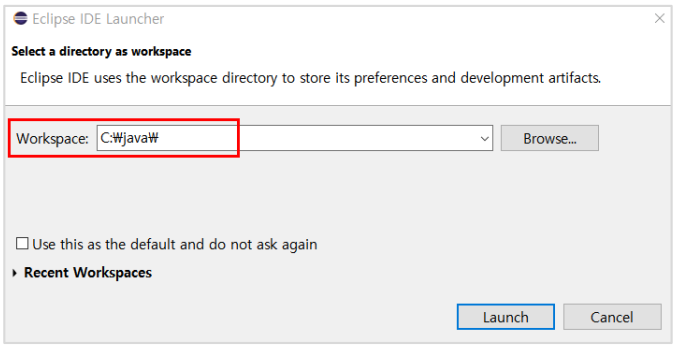
2) Eclipse 설치

① Eclipse 설치 파일을 실행 후 설치할 패키지로 'Eclipse IDE for Java Developers' 선택

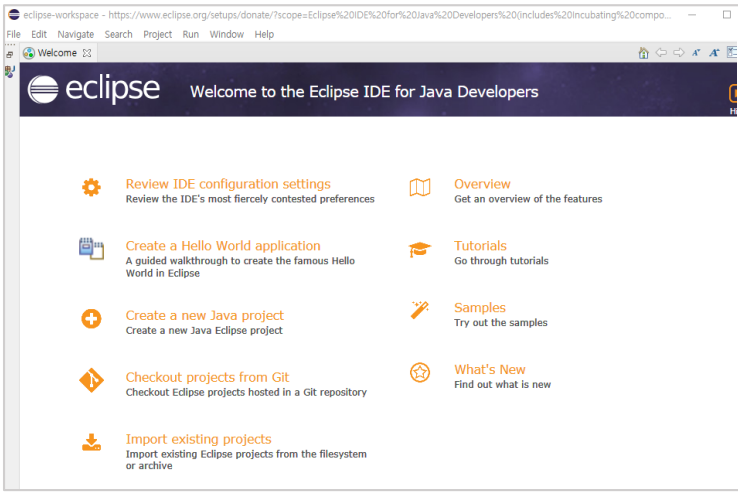
② 시스템에 설치된 JDK 와 Eclipse 를 설치할 폴더를 확인한 후 [INSTALL] 클릭



③ 이클립스를 실행하면 Workspace 를 지정하는 대화상자에 workspace(작업 폴더) 즉, 자신이 사용하는 폴더명을 선택하고 [Launch] 클릭



④ 새로운 workspace 를 시작할 때 나타나는 Welcome 페이지 [닫기]



3. 간단한 출력 프로그램 작성

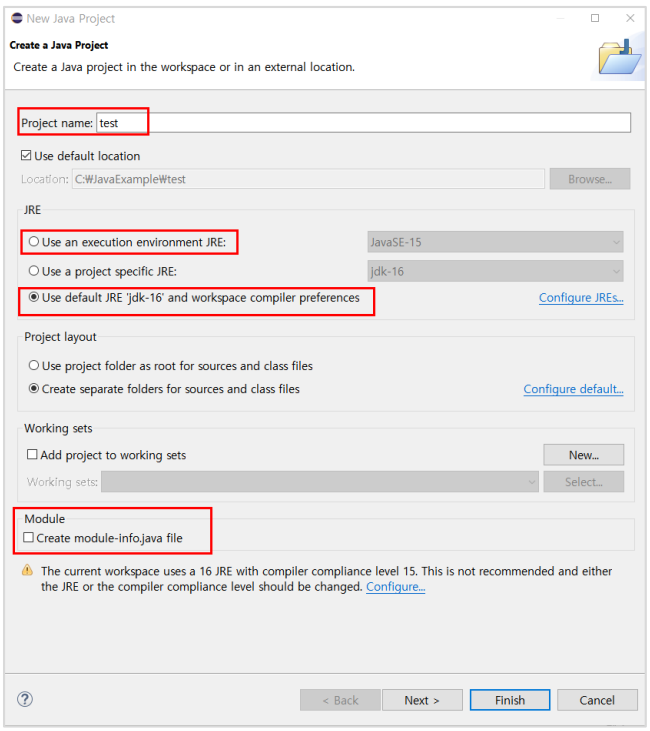
1) 프로젝트 생성

① Package Explorer 에서 [Create a Java Project]를 클릭하거나, [File] - [New] - [Java Project]를 클릭

② [Create a Java Project] 창에서 [Project Name]에 프로젝트 명으로 'test' 입력

③ [Create module-info.java]의 체크를 해제

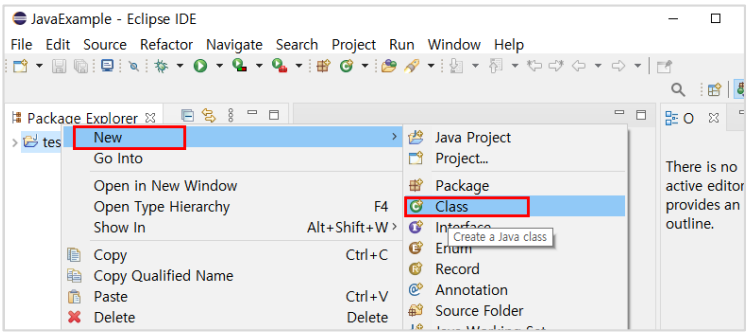
④ [Finish] 클릭



2) 클래스 생성

① [Package Explorer]에서 'test'프로젝트를 선택한 후 마우스 오른쪽 버튼 클릭

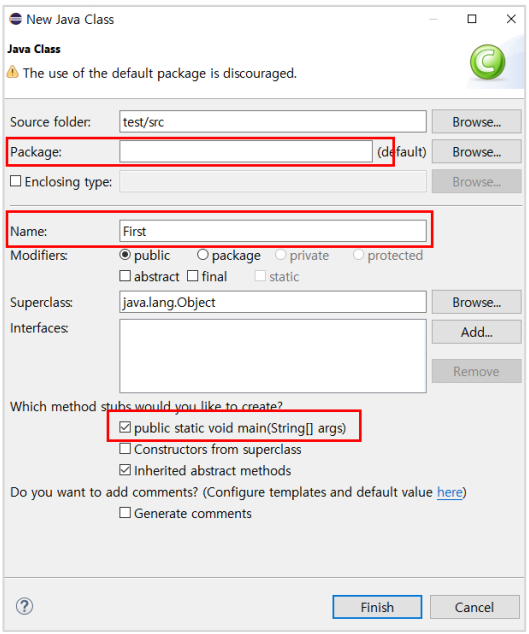
② 팝업 메뉴에서 [New]-[Class] 선택 (or [New]아이콘 - [Class] 선택)



③ [New Java Class] 창에서 [package]는 빈 칸으로 두고, [Name] 에 클래스명으로 'First' 를 입력

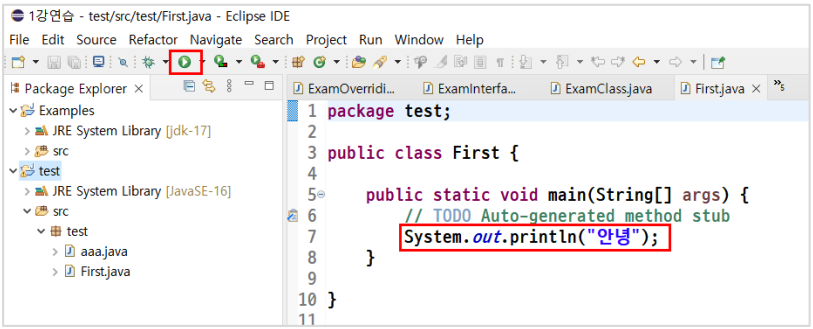
④ [public static void main(String[ ] args)]를 체크

⑤ [Finish] 클릭

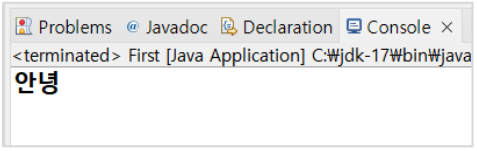


3) main( ) 메소드에 코드 작성

① main( ) 메소드 안에 System.out.println( ) 메소드를 이용하여 "안녕" 을 출력하는 자바 코드 추가

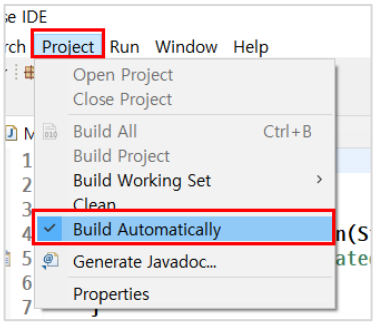


② 저장 후 실행



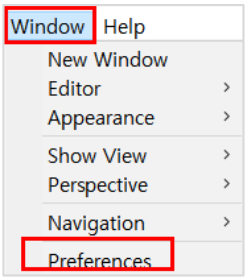
※ 이클립스에서 sysout 을 타이핑한 후 [ctrl] + [space] 를 누르면 System.out.println(); 이 자동으로 작성됨

※ 컴파일을 별도로 하지 않고 run 만 누르면 실행되는 이유는 이클립스의 [Project] 메뉴에 [Build Automatically] 가 체크되어 있기 때문이며, [Build Automatically] 를 체크하면 자동으로 컴파일 후에 실행됨



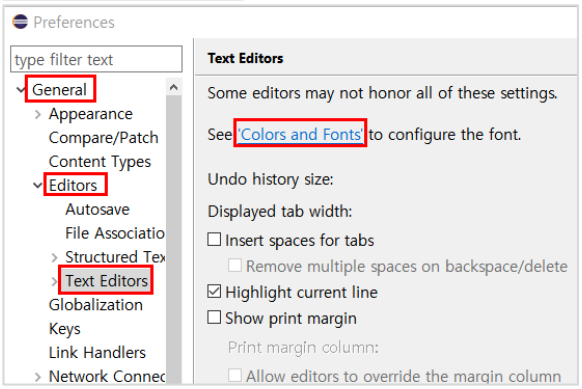
3) Eclipse 폰트 변경

① Eclipse 메뉴에서 [Window]-[Preferences] 선택

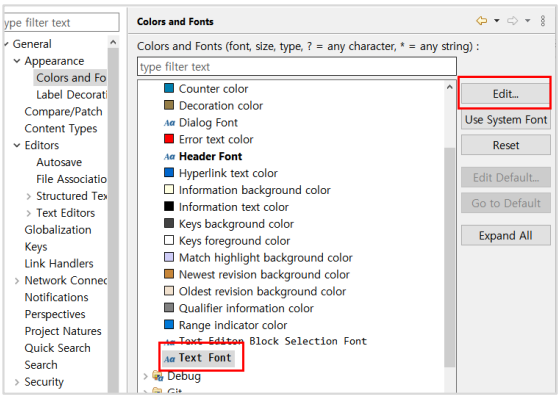


② [Preferences]-[General]-[Editors]-[Text Editors]선택

③ 'Colors and Fonts'클릭

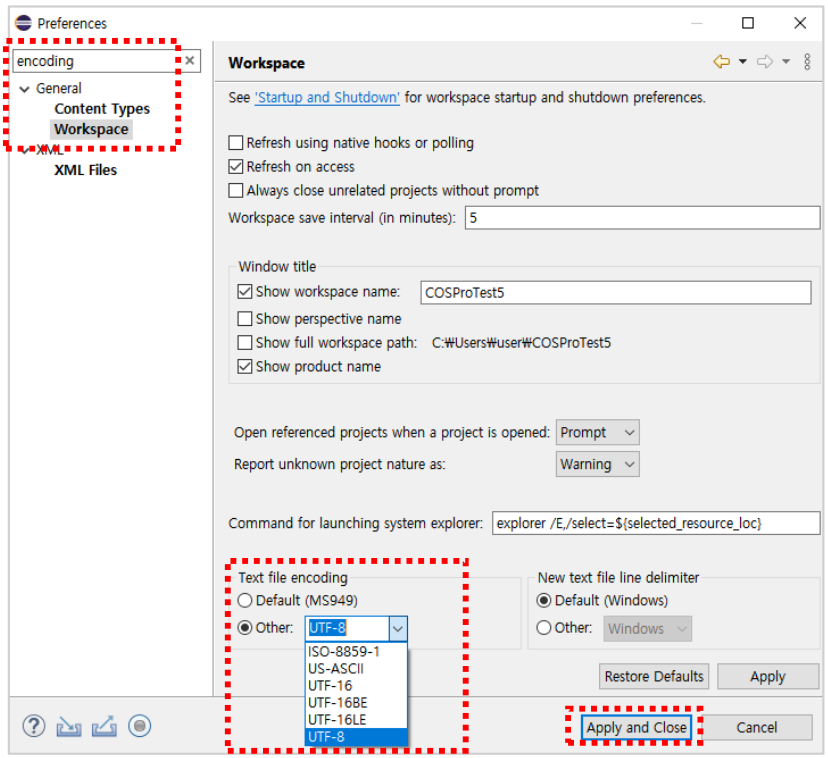


④ [Text Font] 클릭하여 폰트를 변경



※ 팁 : 이클립스에서 encoding 변경하는 법

① Eclipse 메뉴에서 [Window]-[Preferences] 선택
② [Preferences]창의 검색 란에 'encoding' 입력
③ [General]-[Workspace]선택
④ [Text file encoding]항목값을 'Default' →'Other'로 변경하고 그 값을 'UTF-8'로 지정
⑤ [Apply and Close] 클릭



※ 이클립스 단축키 모음

- ctrl + +, - : 폰트 크기
- ctrl + D : 한 줄 삭제
- ctrl + / : 주석 토글
- ctrl + space : 자동 완성
- ctrl + i : 자동 들여쓰기
- 이클립스의 에러 부분에서 ctrl + shift + o ➔ 자동으로 import 됨
- sysout ➔ ctrl + space : System.out.println()

2 변수

1. 기본 자료형

1) 정수

- 많이 사용되는 정수형 자료형은 int 와 long 이 있음
- int 타입은 4 바이트(32 비트) 만큼의 메모리를 할당하여 정수를 표현하는데, 그 범위가 -2,147,483,648 ~ 2,147,483,647 까지의 값을 나타낼 수 있음
- long 타입은 8 바이트(64 비트) 만큼의 메모리를 할당하여 수를 표현하는데, 그 범위가 -9,223,362,036,854,775,808 ~ 9,233,372,036,854,775,807 까지의 값을 나타낼 수 있음

2) 실수

- 많이 사용되는 실수형 자료는 float 와 double 이 있음
- float 타입은 4 바이트(32 비트) 만큼의 메모리를 할당하여 실수를 표현하는데, 그 범위가 $\pm(1.40 * 10^{-45} \sim 3.4 * 10^{38})$ 까지의 값을 소수점 아래로 7 자리까지 나타낼 수 있음
- double 타입은 8 바이트(64 비트) 만큼의 메모리를 할당하여 실수를 표현하는데, 그 범위가 $\pm(4.94 * 10^{-324} \sim 1.79 * 10^{308})$ 까지의 값을 소수점 아래 15 자리까지 나타낼 수 있음
- 좀 더 정밀도가 높은 값을 사용하는 경우는 double 을 사용
- 변수의 데이터 타입을 결정할 때는 사용할 데이터의 최댓값을 고려해서 지정해야 오버플로우가 발생하지 않음

3) 문자

- 문자를 표현하기 위해 char 를 사용
- 2 바이트(16 비트)를 사용해서 문자에 대한 유니코드 값을 저장하므로 덧셈, 뺄셈 연산이 가능

4) 논리값

- 논리값을 표현하기 위해 boolean 을 사용
- 1 바이트(8 비트)를 사용하며 true/false 값이 존재

자료형	데이터	크기	범위
boolean	참, 거짓	1 byte	true, false
char	문자	2byte	유니코드 문자
byte	정수	1byte	-128 ~ 127
short		2byte	-32,768 ~ 32,767
int		4byte	-2,147,483,648 ~ 2,147,483,647
long		8byte	-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807
float	실수	4byte	$\pm(1.40 \times 10^{-45} \sim 3.40 \times 10^{38})$
double		8byte	$\pm(4.94 \times 10^{-324} \sim 1.79 \times 10^{308})$

2. 변수 선언

- 데이터를 저장하기 위해 이름을 지정한 메모리 공간
- 프로그램 안에서 변수 선언을 먼저 해야 사용 가능
- 변수명은 알파벳, 숫자, 밑줄(_), 달러 기호($)를 사용할 수 있지만 변수의 첫 번째 글자로 숫자는 사용 불가

```
int a1 ;
a1 = 10;

double marketRate = 12.5 ;
```

- 자료형을 먼저 표기한 후 변수명을 지정
- 변수명은 소문자로 표기하는 것이 일반적
- 두 단어 이상으로 구성된 변수명은 새로운 단어의 첫 글자를 대문자로 표기

- 상수(Constants) : 값이 변하지 않는 수

```
final int MAX_SIZE=100;
```

Final 선언을 추가하면 상수가 되며, 한번만 값을 할당할 수 있고, 한번 할당된 값은 변경 불가

- 리터럴(Literal) : 프로그램 안에서 사용되는 값 자체

- 타입에 따른 리터럴
  - 정수형 리터럴 : 100, -35 와 같이 숫자와 부호로 표시된 수
  - 실수형 리터럴 : 3.14, -19.1 과 같이 소수점이 있는 숫자와 부호로 표시된 수
  - 논리형 리터럴 : true, false 로 표시되는 값
  - 문자형 리터럴 : 'A', 'z' 와 같이 작은 따옴표(')로 감싼 문자
  - 문자열 리터럴 : "Happy" 와 같이 큰 따옴표(")로 감싼 문자열

- 리터럴 타입 접미사

타입 접미사	리터럴 타입	예시
L 또는 l	long 형	15789L
F 또는 f	float 형	3.14f
D 또는 d (생략 가능)	double 형	1.24690325d

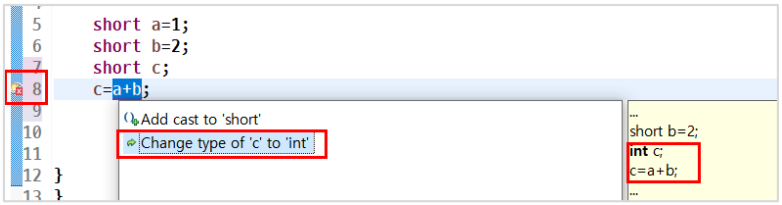
```
boolean b = true;   //true,false

char c='A';
String str = "AB";

byte b=128;   //에러
short s = 10;
int i = 4;        //10_000
long l =10L;   //L(l) 가능, 생략 불가
```

```
float f = 1.2f;    //F(f) 가능, 생략 불가
double d=1.2d; //D(d) 가능, 생략 가능
double d=1.2;
double d=1e2;  // 100.0
```

<기본 자료형의 크기>

1byte	boolean, byte
2byte	char, short
4byte	int, float
8byte	long, double

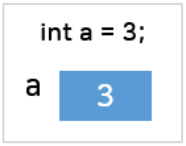
※ 주의 사항



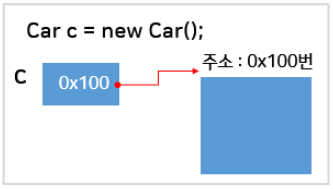
- 자바 기본 정수 연산은 int
- a,b,c 변수 모두 short 타입이나, 더하기 정수 연산을 한 경우 전부 int 타입으로 변경한 후 계산하기 때문에 결과인 c 변수는 int 이어야 한다. 크기가 작은 변수(short)에 크기가 큰 변수(int)를 집어 넣을 수 없기 때문이며, 연산을 동반한 경우에는 byte 나 short 로 선언해도 메모리가 절약되지 않고 오히려 데이터 변환 과정만 추가되므로 기본 int 로 하는 것이 적당함

3. 자바의 자료형

- **기본형(Primitive type)**
  - boolean, char, byte, short, int, long, float, double
  - 값을 저장



- **참조형(Reference type)**
  - 메모리 주소를 저장
  - 객체의 주소를 저장하는 방식



4. 형 변환(type conversion)

1) 묵시적 형 변환(자동 형 변환, Implicit Conversion)

- 크기가 작은 자료형의 데이터를 큰 자료형의 변수에 대입하는 경우 컴파일러가 자동으로 형 변환을 수행
- 자료형의 크기에 상관없이 정수 자료형보다 실수 자료형이 우선
  - byte(1) ➔ short(2), char(2) ➔ int(4) ➔ long(8) ➔ float(4) ➔ double(8)

```
float n = 10   // int 10은 float로 자동 형변환
30L + 12.8     // long형 30은 double형으로 자동 형 변환
```

2) 명시적 형 변환(강제 형 변환, Explicit Conversion)

- 크기가 큰 자료형의 데이터를 더 작은 크기의 자료형 변수로 할당할 때 데이터 손실이 일어나기 때문에 강제적인 형 변환이 필요

```
long num1=200L;
int num2=(int)num1;  // num1을 int로 강제 형변환
```

5. 연산자

1) 대입 연산자와 증감 연산자

연산자	의미
A += B	A 에 B 를 더한 값을 A 에 할당
A -= B	A 에서 B 를 뺀 값을 A 에 할당
A *= B	A 에 B 를 곱한 값을 A 에 할당
A /= B	A 를 B 로 나눈 값을 A 에 할당
A %= B	A 를 B 로 나눈 나머지를 A 에 할당
A++	A 를 1 만큼 증가
A--	A 를 1 만큼 감소

2) 논리 연산자

연산자	의미
A && B	A 와 B 모두 참이면 참(AND 연산)
A || B	A 와 B 중 하나라도 참이면 참(OR 연산)
!A	A 가 참이면 거짓, A 가 거짓이면 참

※ 기본적인 사칙 연산자와 비교 연산자는 일반 수학 연산자와 동일

종류	연산자
산술 연산자	+, - , *, /, %, (<<, >>)
비교 연산자	>, <, >=, <=, ==, !=
논리 연산자	&&, ||, !, (&, |, ^, ~)

3 배열

1. 1 차원 배열

1) 1 차원 배열의 선언

- 자료형[ ] 변수명 ;
- 자료형 변수명[ ];

```
int[] arr;
int arr[];
```

2) 1 차원 배열 선언 후 생성

- 자료형[ ] 변수명 = new 자료형[배열의 길이]

```
int[] arr = new int[3];
```

- 자료형[ ] 변수명;
  변수명 = new 자료형[배열의 길이];

```
int[] arr;
arr=new int[3];
```

3) 1 차원 배열 생성 & 초기값 지정

- 자료형[ ] 변수명 = {값 1, … , 값 2}
- 자료형[ ] 변수명 = new 자료형[ ]{값 1, … , 값 2}

```
int[] arr = {1,2,3};

int[] arr = new int[] {1,2,3};
int[] arr = new int[3] {1,2,3}; //에러
```

※ 마지막 문장처럼 배열 크기와 함께 초기값을 주면 에러
자료형[ ] 변수명 = new 자료형[배열 크기]{값 1, … , 값 2}

4) 1 차원 배열 항목 참조

- 인덱스를 지정해서 배열의 개별 항목 참조
- 인덱스 값 : 0 부터 (배열의 크기-1)까지의 값을 가짐
  예시) **a[0], a[1], a[2]**

5) 배열 항목의 기본 값

배열이 생성될 때 기본 모든 요소는 0 혹은 null 로 초기화 된다.

```
int[] arr=new int[3];          // 모든 요소 0으로 초기화
String[] str= new String[5];   // 모든 요소 null로 초기화
```

그 외의 초기값은 아래 표 참고.

자료형	기본값
정수(int, long, byte, …)	0
실수(float, double)	0.0
문자(char)	0
boolean	false

6) 배열의 값 초기화

- for 문을 이용하여 초기화

```
int[] k1= {1,2,3};
```

```
int[] k1= new int[3];

for(int i=0; i<k1.length ; i++) {
    k1[i]=10;
    System.out.print(k1[i]+"\t");
}
```

- fill() 사용하기

```
import java.util.Arrays;
Arrays.fill(배열, 초기값);
```

```
int[] k2= new int[3];
Arrays.fill(k2,20);
```

7) 배열 정렬하기

- import java.util.Arrays;
- 오름차순 정렬 : Arrays.sort(배열);

```
int[] k3= new int[] {49,5,42};
Arrays.sort(k3);
```

- 오름차순 부분 정렬 : Arrays.sort(배열,시작 인덱스,끝 인덱스);

```
Arrays.sort(k3,0,2);
```

- 내림차순 정렬 : Arrays.sort(배열,Collections.reverseOrder());

  Wrapper 타입만 가능

```
Integer [] k4= {49,5,42};
Arrays.sort(k4,Collections.reverseOrder());
```

2. 2 차원 배열

1) 2 차원 배열의 선언

- 2 차원 배열은 행과 열을 갖는 표 형태로 데이터를 저장할 수 있는 자료 구조
- 자료형[ ][ ] 변수명;

```
int[][] arr1;
```

2) 2 차원 배열의 선언 & 생성

- 자료형[ ][ ] 변수명 = new 자료형[행의 길이][열의 길이];

```
int[][] arr = new int[3][3];
```

```
0       0       0
0       0       0
0       0       0
```

- 자료형[ ][ ] 변수명 = {
  {항목값, … 항목값}
  {항목값, …, 항목값}
  }

```
int[][] arr= {
        {1,2,3},
        {4,5,6},
        {7,8,9}
};
```

```
1       2       3
4       5       6
7       8       9
```

➔ 안쪽 중괄호로 지정된 {1, 2, 3} 이 한 행을 나타냄

3) 2 차원 배열의 형태

	열0	열1	열2
행0	[0][0]	[0][1]	[0][2]
행1	[1][0]	[1][1]	[1][2]

< 크기가 [2][3]인 2차원 배열>

- 0 번 행의 1 번 열에 있는 2 차원 배열 항목을 참조하려면, 배열의 인덱스를 '배열변수[0][1]' 로 지정

4) 2 차원 배열의 출력

```
① for(int i=0; i<arr.length; i++) {
②     for(int j=0; j<arr[i].length; j++) {
          System.out.print(arr[i][j]+"\t");
      }
③     System.out.println();
  }
```

① 행을 지정하기 위해 사용하는 for 문 : 인덱스 변수 i 는 2 차원 배열의 행을 지정
② 열을 지정하기 위해 사용하는 for 문 : 인덱스 변수 j 는 2 차원 배열의 열을 지정
③ 한 행의 데이터를 모두 출력한 후 다음 줄로 넘어가기 위해 줄바꿈 문자 출력

3. 배열의 활용

1) 배열의 길이(=배열의 항목 개수)

- 배열변수명.length 필드는 배열의 항목 개수를 나타냄

```
for (int i=0; i<arr.length; i++)
    System.out.println(arr[i]);
```

➔ 인덱스 변수 i가 0부터 배열의 개수보다 작은 동안 1씩 증가하며 배열 arr의 항목을 출력

2) Enhanced for

- 배열 항목값을 지정한 변수로 차례대로 받아와서 반복하는 for 문
- 사용 형식

for(자료형 변수명 : 배열변수명)
{
**   배열의 길이만큼 반복할 명령**
}

```
int[] ar={1,2,3,4,5};

for(int e:ar){
    System.out.println(e);
}
```

➔ 배열 arr 의 0 번 인덱스 항목부터 마지막 항목까지 하나씩 차례대로 변수 a 로 받아오며 for 문 안에 있는 명령을 반복

※ **참고 : Arrays.toString( ) 메소드**

- 배열 항목을 한 번에 한 줄의 문자열로 출력하는 메소드
- Arrays.toString( ) 메소드를 사용하려면 먼저 java.util 패키지를 import 해야 함

```
int a[] = new int[] {3,5,7,9};
System.out.println(Arrays.toString(a));
```