Java 문법 1

동양미래대학교
DONGYANG MIRAE UNIVERSITY

CONTENTS

자바 개요

1. Java 특징

- Java 의 역사
  - 1991년 선마이크로시스템스(오라클에 인수됨)의 제임스 고슬링이 C 언어를 모델로 연구 시작
  - 1995년 JDK(Java Development Kit) 1.0 발표
  - 1997년 JDK 1.1이 발표되면서 완전한 프로그래밍 언어의 모습을 갖춤

- Java의 특징
  ① 구문이 간결함
  ② 명료한 객체지향 언어
  ③ 이식성이 높고, 기계에 중립적
  ④ 분산 처리 지원
  ⑤ 멀티스레드(Multi-thread) 언어

2. Java 프로그램 작성법

- Java 프로그램 전통적인 작성법
  - 메모장에서 Java 코드를 작성한 후에 *.java로 저장
  - javac.exe를 사용해서 컴파일하면 *.class 파일이 생성
  - java.exe를 사용해서 컴파일된 *.class 파일을 실행

→ **개발자들은 대부분 이클립스 환경에서 Java 개발**

2. Java 프로그램 작성법

- 실습 3-1 Eclipse 환경에서 Java 개발하기
  - (1) http://www.eclipse.org/downloads/eclipsepackages/에서 'Eclipse IDE for Java Developers' 다운로드
  - (2) eclipse.exe 실행 → [Eclipse IDE Launcher] 창에서 새 디렉터리 입력
    → 〈Launch〉 클릭 → [Eclipse IDE] 창에서 [Welcome] 창 닫기
    → [File]-[New]-[Java Project] 선택
  - (3) [Create a Java Project] 창에 'Project3 _1' 입력 → 'Use a project specific JRE:' 선택 → 'Create Module-info.java file' 체크버튼 끄기
    - 나머지 : 디폴트로 둔 후 〈Finish〉 클릭

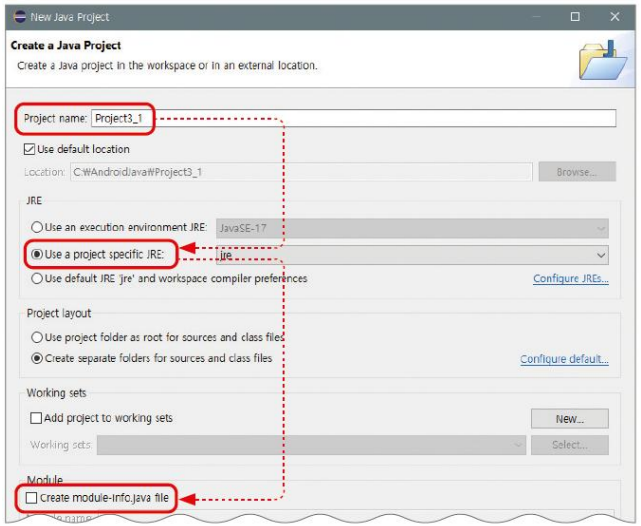


그림 3-1 Java 프로젝트 생성

- 실습 3-1 Eclipse 환경에서 Java 개발하기
  - (4) Package Explorer의 Project3_1/src 폴더에서 마우스 오른쪽 버튼 클릭하고 [New]-[Class] 선택

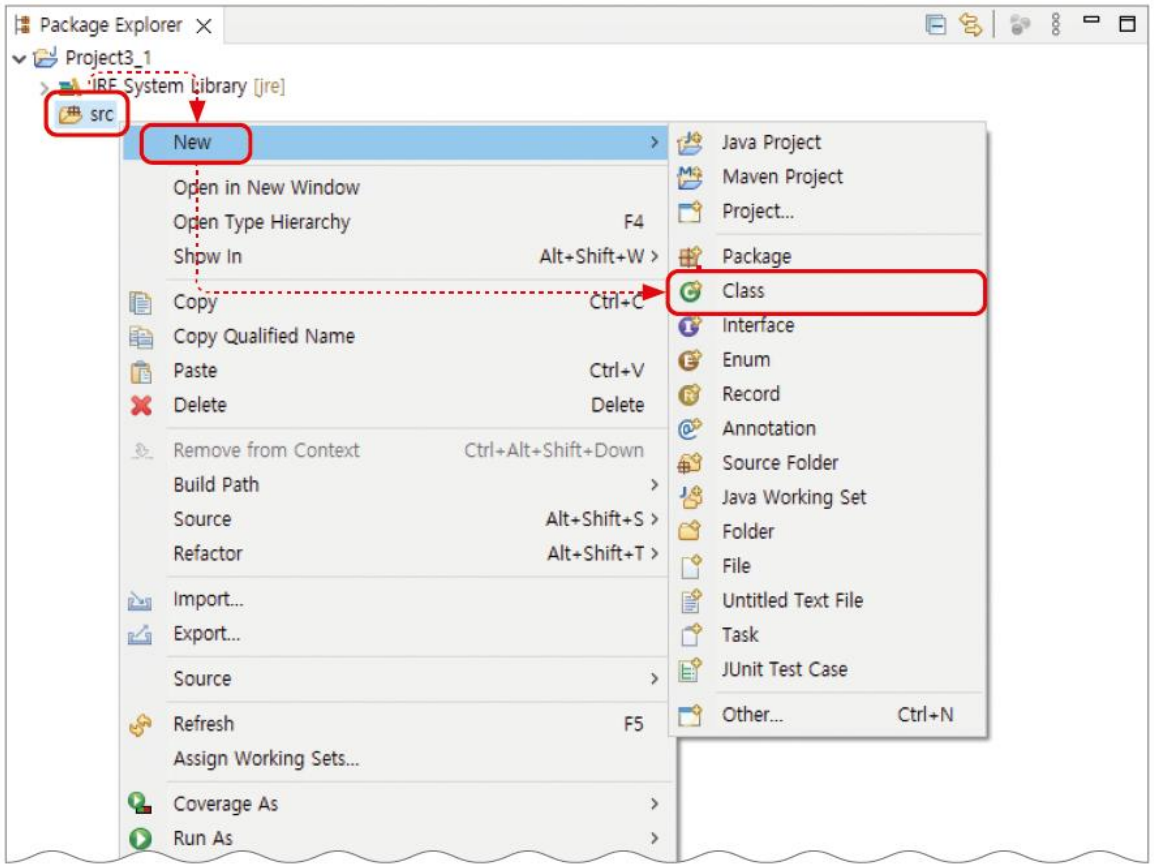


그림 3-2 Java 소스 추가 1

- 실습 3-1 Eclipse 환경에서 Java 개발하기
  - (5) [Java Class] 창에서 Name에 'exam01' 입력하고 public static void main(String[]args)를 체크한 후 〈Finish〉 클릭

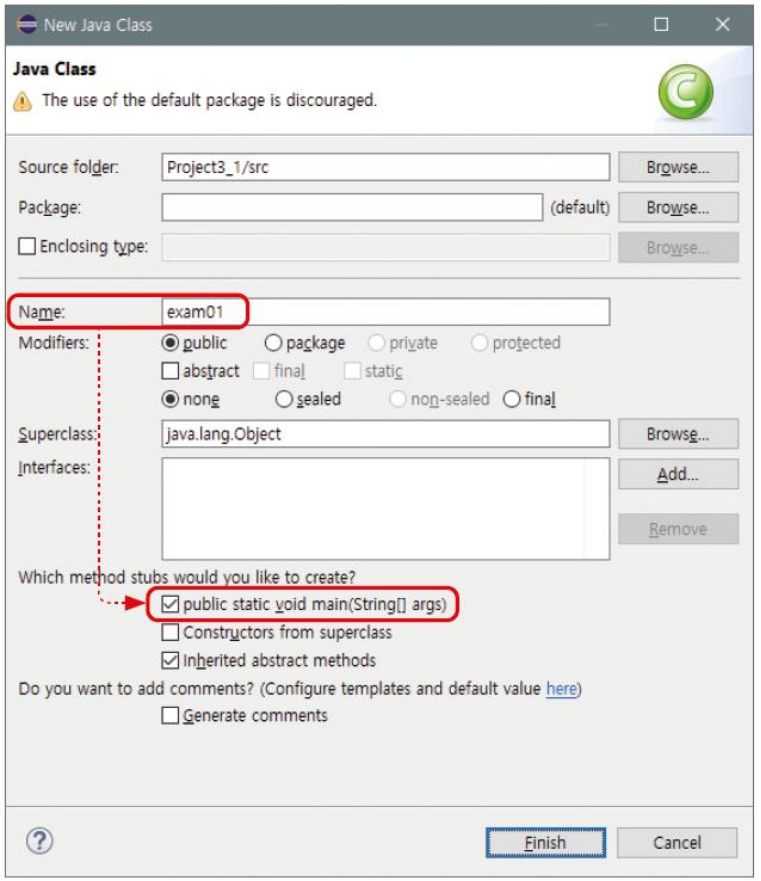


그림 3-3 Java 소스 추가 2

2. Java 프로그램 작성법

- 실습 3-1 Eclipse 환경에서 Java 개발하기
  - (6) 간단한 예제 코딩

예제 3-1 exam01.java

```
1  public class exam01 {
2    public static void main(String args[]) {
3      System.out.println("안드로이드를 위한 Java 연습");
4    }
5  }
```

2. Java 프로그램 작성법

- 실습 3-1 Eclipse 환경에서 Java 개발하기
  - (7) [File]-[Save]를 선택해서 저장한 후[Run]-[Run] 선택 혹은 [ctrl]+[F11] 눌러 실행
    - 실행 결과 : 아래쪽 콘솔에 바로 출력
      - 이후 예제는 예제는 Project3_1/src 폴더에서 마우스 오른쪽 버튼을 클릭하여 [New]-[Class]를 선택하고 exam02, exam03, … 등으로 입력해서 코딩함

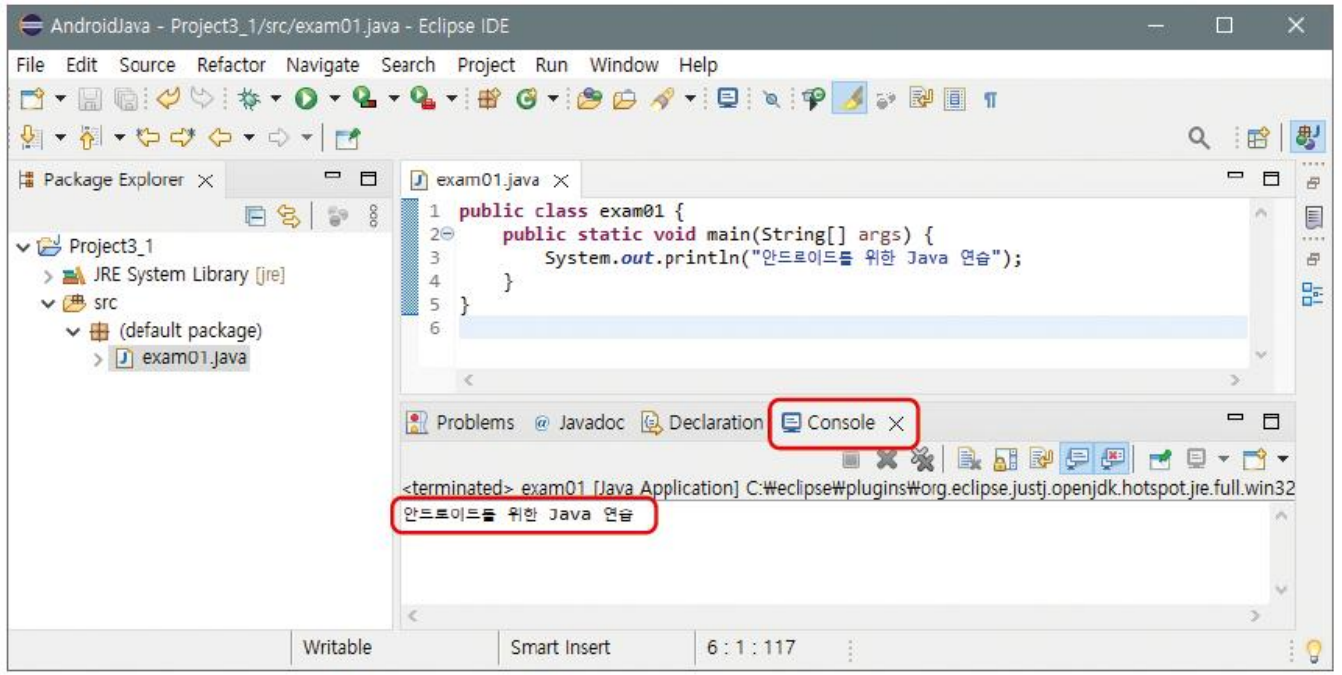


그림 3-4 Java 실행 및 결과 확인

자바 기본문법

1. 변수와 데이터 형식

- 변수 선언 예제

예제 3-2 exam02.java

```
1  public class exam02 {
2      public static void main(String args[]) {
3          int var1 = 10;
4          float var2 = 10.1f;
5          double var3 = 10.2;
6          char var4 = '안';
7          String var5 = "안드로이드";
8          System.out.println(var1);
9          System.out.println(var2);
10         System.out.println(var3);
11         System.out.println(var4);
12         System.out.println(var5);
13     }
14 }
```

```
10
10.1
10.2
안
안드로이드
```

1. 변수와 데이터 형식

- Java에서 많이 사용되는 기본적인 데이터 형식

표 3-1 Java에서 주로 사용되는 데이터 형식

데이터 형식		설명
문자형	char	2byte를 사용하며 한글 또는 영문 1개만 입력
	String	여러 글자의 문자열을 입력
정수형	byte	1byte를 사용하며 −128~+127까지 입력
	short	2byte를 사용하며 −32768~+32767까지 입력
	int	4byte를 사용하며 약 −21억~+21억까지 입력
	long	8byte를 사용하며 상당히 큰 정수까지 입력 가능
실수형	float	4byte를 사용하며 실수를 입력
	double	8byte를 사용하며 실수를 입력. float보다 정밀도가 높음
불리언형	boolean	true 또는 false를 입력

2. 조건문: if, switch()~case

- if문
  - 조건이 true, false인지에 따라서 어떤 작업을 할 것인지를 결정

```
if(조건식) {
// 조건식이 true일 때 이 부분 실행
}
```

```
if(조건식) {
// 조건식이 true일 때 이 부분 실행
} else {
// 조건식이 false일 때 이 부분 실행
}
```

2. 조건문: if, switch()~case

- switch()~case문
  - 여러 가지 경우에 따라 어떤 작업을 할 것인지를 결정

```
switch( 값 ) {
case 값1:
      // 값1이면 이 부분 실행
      break;
case 값2:
      // 값2이면 이 부분 실행
      break;
⋮
default:
      // 어디에도 해당하지 않으면 이 부분 실행
      break;
}
```

2. 조건문: if, switch()~case

예제 3-3 exam03.java

```
1  public class exam03 {
2     public static void main(String args[]) {
3        int count = 85;
4        if (count >= 90) {
5           System.out.println("if문: 합격 (장학생)");
6        } else if (count >= 60) {
7           System.out.println("if문: 합격");
8        } else {
9           System.out.println("if문: 불합격");
10       }
11
12       int jumsu = (count / 10) * 10;
13       switch (jumsu) {
14       case 100:
15       case 90:
16          System.out.println("switch문: 합격(장학생)");
17          break;
18       case 80:
19       case 70:
20       case 60:
21          System.out.println("switch문: 합격");
22          break;
23       default:
24          System.out.println("switch문: 불합격");
25       }
26    }
27 }
```

```
if문: 합격
switch문: 합격
```

3. 배열

- 배열
  - 여러 데이터를 한 변수에 저장하는 데 사용

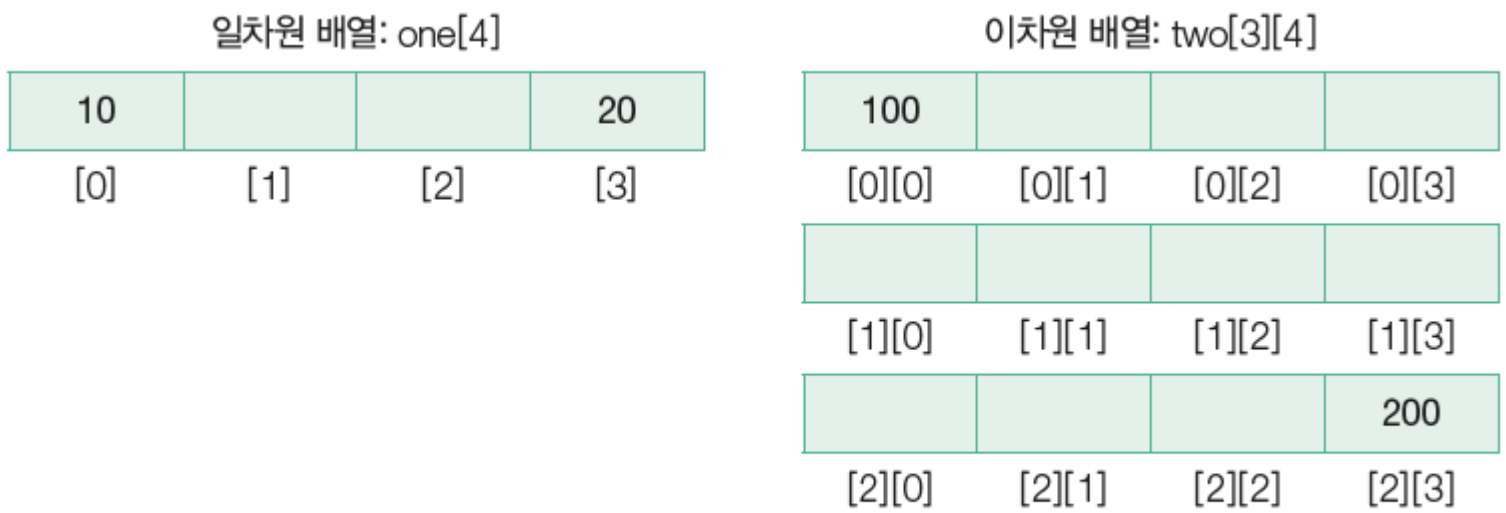


그림 3-5 **배열의 개념**

3. 배열

- 일차원 배열

```
int one[] = new int[4];
one[0] = 10;
one[3] = 20;
```

- 이차원 배열

```
int two[][] = new int[3][4];
two[0][0] = 100;
two[2][3] = 200;
```

3. 배열

- 배열 선언하면서 바로 값 대입하기

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

- 배열 크기 확인하기
  - '배열.length' 사용

4. 반복문: for, while

- for문
  - 조건문과 함께 프로그래밍의 필수 요소임

```
for(초기식; 조건식; 증감식) {
// 이 부분을 반복 실행
}
```

- 배열을 지원하는 for문의 형식
  - 배열의 내용이 하나씩 변수에 대입된 후 for문 내부가 실행됨
  - 결국 배열의 개수만큼 for문이 반복됨

```
for(변수형 변수 : 배열명) {
// 이 부분에서 변수 사용
}
```

4. 반복문: for, while

- while문

```
while( 조건식 ) {
// 조건식이 true인 동안 이 부분을 실행
}
```

4. 반복문: for, while

예제 3-4 exam04.java

```
1  public class exam04 {
2     public static void main(String args[]) {
3        int one[] = new int[3];
4        for (int i = 0; i < one.length; i++) {
5           one[i] = 10 * i;
6        }
7
8        String two[] = { "하나", "둘", "셋" };
9        for (String str : two) {
10          System.out.println(str);
11       }
12
13       int j=0;
14       while( j < one.length ) {
15          System.out.println(one[j]);
16          j++;
17       }
18    }
19 }
```

```
하나
둘
셋
0
10
20
```

5. 메소드와 전역변수, 지역변수

- 변수
  - 전역변수(global variable) : 모든 메소드에서 사용 가능함
  - 지역변수(local variable) : 메소드 내부에서만 사용 가능함

예제 3-5 exam05.java

```
1  public class exam05 {
2     static int var = 100;
3     public static void main(String args[]) {
4        int var = 0;
5        System.out.println(var);
6
7        int sum = addFunction(10, 20);
8        System.out.println(sum);
9     }
10
11    static int addFunction(int num1, int num2) {
12       int hap;
13       hap = num1 + num2 + var;
14       return hap;
15    }
16 }
```

```
0
130
```

6. 예외 처리: try~catch

- try~catch
  - 프로그램 실행 중에 발생하는 오류를 Java는 try~catch문을 통해 처리

예제 3-6 exam06.java

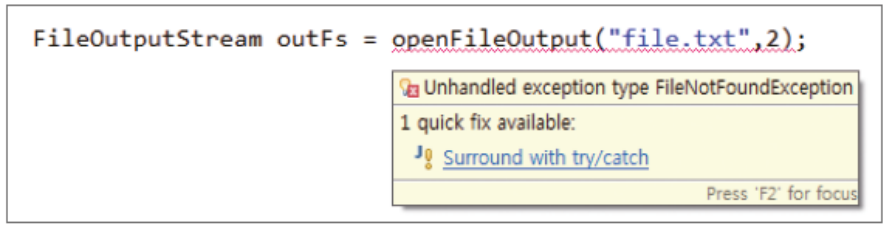
```
1  public class exam06 {
2     static int var = 100;
3     public static void main(String args[]) {
4        int num1 = 100, num2 = 0;
5        try {
6           System.out.println(num1/num2);
7        }
8        catch (java.lang.ArithmeticException e) {
9           System.out.println("계산에 문제가 있습니다.");
10       }
11    }
12 }
```

```
계산에 문제가 있습니다.
```

6. 예외 처리: try~catch

여기서 잠깐 try~catch문 자동 완성

안드로이드에서는 예외 처리에 자동 완성을 많이 사용한다. 다음 그림에서 밑줄이 쳐진 openFileOuput( )에 마우스를 가져가면 팝업 창이 나오는데, 여기서 'Surround with try/catch'를 클릭하면 된다.



try~catch문이 자동 완성된 결과는 다음과 같다. 자세한 내용은 8장에서 살펴보겠다.

```
try {
    FileOutputStream outFs = openFileOutput("file.txt",2);
} catch (FileNotFoundException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

7. 연산자

- Java에서 주로 사용되는 연산자

표 3-2 주로 사용되는 Java 연산자

연산자	설명
+, −, *, /, %	사칙 연산자로 %는 나머지값을 계산한다.
+, −	부호 연산자로 변수, 수, 식 앞에 붙일 수 있다.
=	대입 연산자로 오른쪽을 왼쪽에 대입한다.
++, −−	1씩 증가 또는 감소시킨다.
==, !=, <, >, >=, <=	비교 연산자로 결과는 true 또는 false이며, if문이나 반복문의 조건식에 주로 사용된다.
&&, ||	논리 연산자로 and, or를 의미한다.
&, |, ^, ~	비트 연산자로, 비트 단위로 and, or, exclusive or, not 연산을 한다.
<<, >>	시프트 연산자로, 비트 단위로 왼쪽 또는 오른쪽으로 이동한다.
+=, −=, *=, /=	복합 대입 연산자로, 'a+=b'는 'a=a+b'와 동일하다.
(데이터 형식)	캐스트(cast) 연산자로, 데이터 형식을 강제로 변환한다. 예를 들어 int a = (int) 3.5는 double형인 3.5 값을 int형으로 강제로 변환하여 a에 대입한다. 결국 a에 3이 대입된다.

7. 연산자

- 캐스트 연산자
  - 안드로이드 프로그래밍에서 클래스형 데이터의 강제 형식 변환에도 상당히 많이 사용됨
  - 안드로이드 프로그래밍에서 캐스트 연산자 사용 예
    - View 클래스형을 Button형으로 변환

```
Button button1;
button1 = (Button) findViewById(R.id.btn1);
```