자바 기초 문법

Instructor: Park, JoonSeok (pjs50@pusan.ac.kr)

Java 기본 문법 ▶ 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
```

Java 기본 문법 ▶ 변수와 데이터 형식

❖ 주로 사용되는 데이터 형

[표 3-1] Java에서 주로 사용되는 데이터 형

분류	데이터 형	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 정밀도 높음
불리언형	boolean	true 또는 false를 입력

Java 기본 문법 ▶ 조건문

- **if 조건식 / switch()~case 개념**

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

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

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

Java 기본 문법 ▶ 조건문

❖ if 조건식 / switch()~case 예제

[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
```

Java 기본 문법 ▶ 배열

❖ 배열 개념

일차원 배열 : one[4]

10			20
[0]	[1]	[2]	[3]

이차원 배열 : two[3][4]

100			
[0][0]	[0][1]	[0][2]	[0][3]
[1][0]	[1][1]	[1][2]	[1][3]
			200
[2][0]	[2][1]	[2][2]	[2][3]

[그림 3-5] 배열 개념

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

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

Java 기본 문법 ▶ 반복문

- **for,while 개념**

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

```
while( 조건식 ) {
    // 조건식이 true인 동안에 이 부분을 수행
}
```

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

Java 기본 문법 ▶ 반복문

❖ for,while 예제

[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
```

Java 기본 문법 ▶ 메소드와 전역변수, 지역변수

❖ 메소드 예제

[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
```

Java 기본 문법 ▶ 예외 처리

❖ try~catch 예제

[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.
```

Java 기본 문법 ▶ 연산자

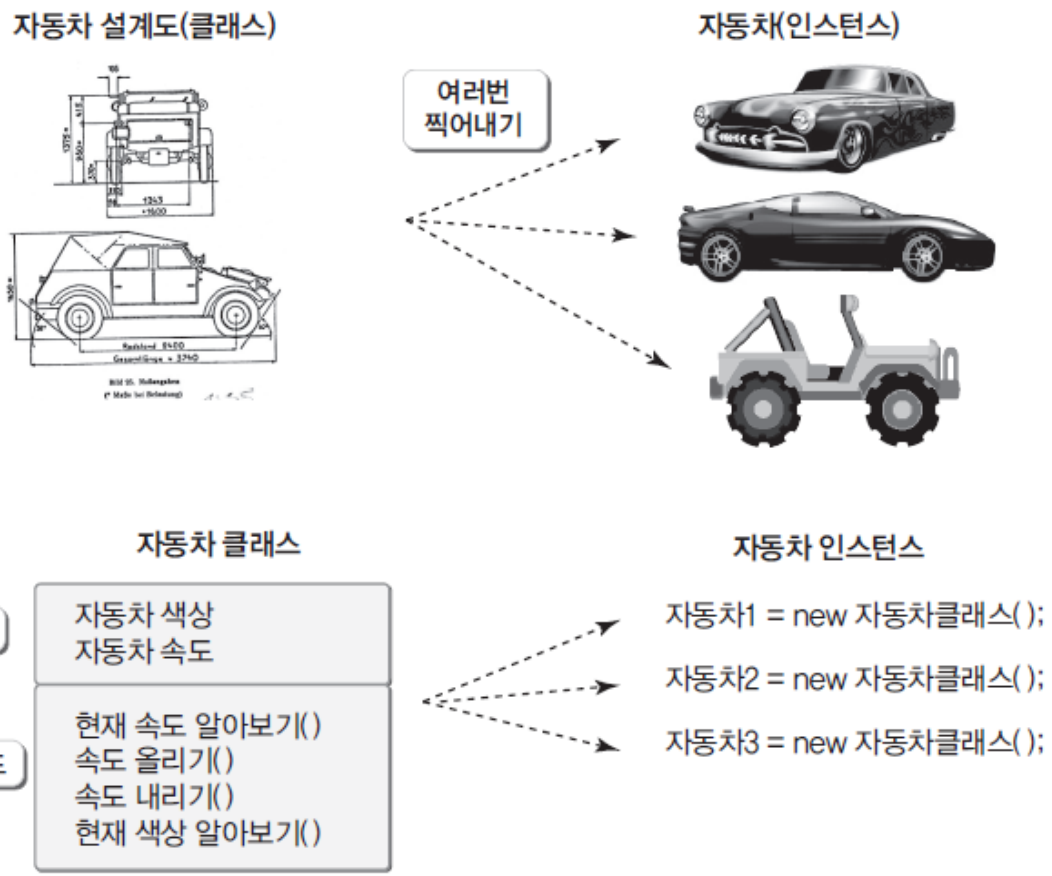
❖ 주요 연산자

[표 3-2] 주로 사용되는 Java 연산자

연산자	설명
+, −, *, /, %	사칙 연산자로, %는 나머지 값을 계산한다.
+, −	부호 연산자로, 변수, 수, 식 앞에 붙일 수 있다.
=	대입 연산자로, 오른쪽을 왼쪽에 대입한다.
++, —	1씩 증가 및 감소시킨다.
==, !=, <, >, >=, <=	비교 연산자로, 결과는 true 또는 false가 되며, 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 이동시킨다.
+=, −=, *=, /=	복합 대입 연산자로, "a +=b"는 "a=a+b"와 동일하다.
(데이터 형)	캐스트(cast) 연산자로, 데이터 형을 강제로 변환시켜 준다. 예로 int a = (int) 3.5는 double형인 3.5 값을 int형으로 강제로 변환시켜서 a에 대입한다. 결국 a에는 3이 대입된다.

클래스와 인스턴스 ▶ 클래스 정의와 인스턴스 생성

❖ 개념



❖ 클래스 정의 예제

[예제 3-7] Car.java – 기본 구조

```
1  public class Car {
2    String color;
3    int speed = 0;
4
5    int getSpeed() {
6      return speed;
7    }
8
9    void upSpeed(int value) {
10     if (speed + value >= 200)
11       speed = 200;
12     else
13       speed = speed + value;
14   }
15
16   void downSpeed(int value) {
17     if (speed - value <= 0)
18       speed = 0;
19     else
20       speed = speed - value;
21   }
22
23   String getColor() {
24     return color;
25   }
26 }
```

❖ 인스턴스 생성 예제

[예제 3-8] exam07.java

```
1  public class exam07 {
2    public static void main(String args[]) {
3      Car myCar1 = new Car();
4      myCar1.color = "빨강";
5      myCar1.speed = 0;
6
7      Car myCar2 = new Car();
8      myCar2.color = "파랑";
9      myCar2.speed = 0;
10
11     Car myCar3 = new Car();
12     myCar3.color = "초록";
13     myCar3.speed = 0;
14
15     myCar1.upSpeed(50);
16     System.out.println("자동차1의 색상은 " + myCar1.getColor()
17         + "이며, 속도는 "
18         + myCar1.getSpeed() + "km입니다.");
19
20     myCar2.downSpeed(20);
21     ~~~~~ 중간 생략 (myCar2 내용 출력) ~~~~~
22     myCar3.upSpeed(250);
23     ~~~~~ 중간 생략 (myCar3 내용 출력) ~~~~~
24   }
25 }
```

```
자동차1의 색상은 빨강이며, 속도는 50km입니다.
자동차2의 색상은 파랑이며, 속도는 0km입니다.
자동차3의 색상은 초록이며, 속도는 200km입니다.
```

❖ 생성자 예제

[예제 3-9] Car.java – 생성자 추가

```
1  public class Car {
2    String color;
3    int speed;
4
5    Car(String color, int speed) {
6      this.color = color;
7      this.speed = speed;
8    }
9    ~~~~ 중간 생략([예제 3-7]의 5행 이하와 동일) ~~~~
```

[예제 3-10] exam07.java – 수정

```
1  public class exam07 {
2    public static void main(String args[]) {
3      Car myCar1 = new Car("빨강",0);
4      Car myCar2 = new Car("파랑",0);
5      Car myCar3 = new Car("초록",0);
6
7      ~~~~ 중간 생략([예제 3-8]의 15행 이하와 동일) ~~~~
```

❖ 메소드 오버로딩 예제

[예제 3-11] Car.java – 메소드 오버로딩 추가

```
1  public class Car {
2    String color;
3    int speed;
4
5    Car(String color, int speed) {
6      this.color = color;
7      this.speed = speed;
8    }
9
10   Car(int speed) {
11     this.speed = speed;
12   }
13
14   Car() {
15   }
16
17   ~~~~ 중간 생략([예제 3-7]의 5행 이하와 동일) ~~~~
```

클래스와 인스턴스 ▶ 정적 필드, 정적 메소드, 상수 필드

❖ 예제

[예제 3-12] Car.java – 정적 구성 요소 추가

```
1  public class Car {
2    String color;
3    int speed;
4    static int carCount = 0;
5    final static int MAXSPEED = 200;
6    final static int MINSPEED = 0;
7
8    static int currentCarCount() {
9      return carCount;
10   }
11
12   Car(String color, int speed) {
13     this.color = color;
14     this.speed = speed;
15     carCount ++;
16   }
17
18   ~~~~ 중간 생략([예제 3-11]의 10행 이하와 동일) ~~~~
```

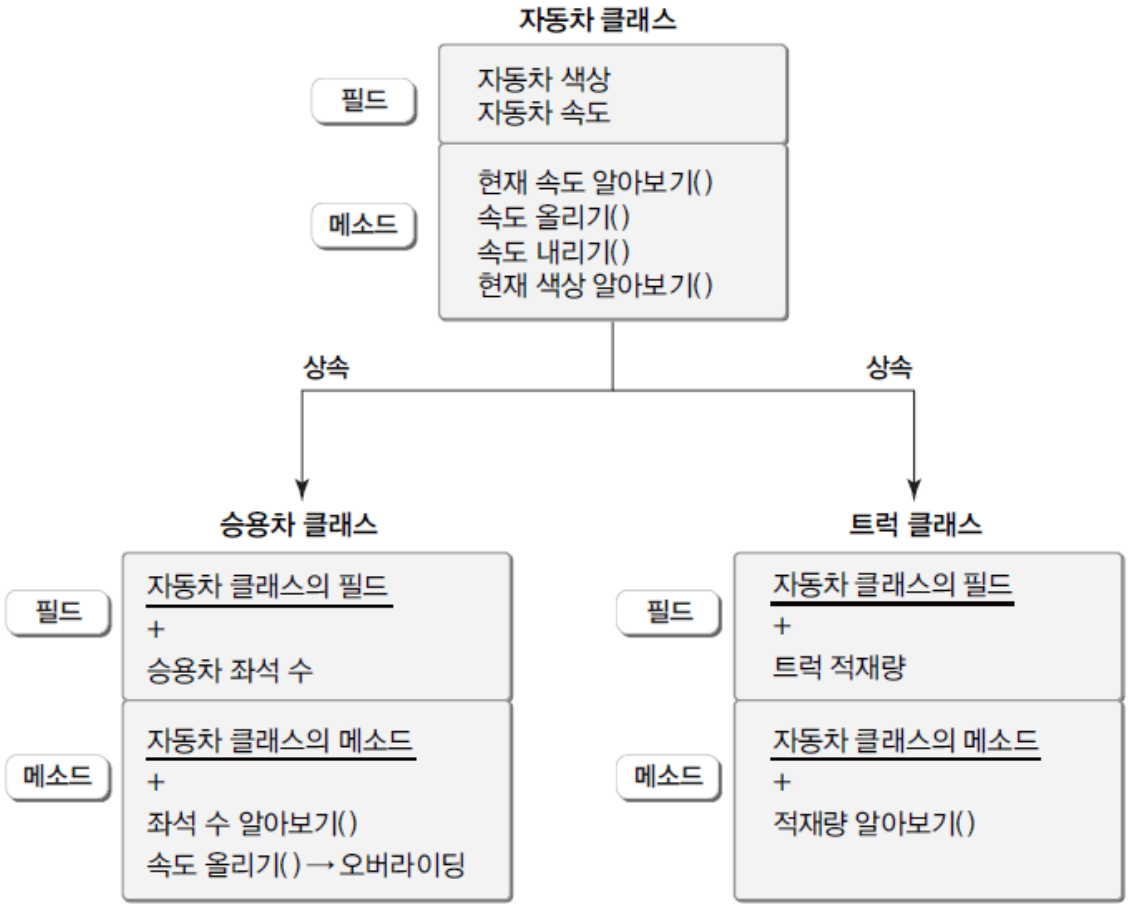
[예제 3-13] exam08.java

```
1  import java.lang.Math;
2
3  public class exam08 {
4    public static void main(String args[]) {
5      Car myCar1 = new Car("빨강", 0);
6      Car myCar2 = new Car("파랑", 0);
7      Car myCar3 = new Car("초록", 0);
8
9      System.out.println("생산된 차의 대수(정적 필드) ==> " + Car.carCount);
10     System.out.println("생산된 차의 대수(정적 메소드) ==> " + Car.currentCarCount());
11     System.out.println("차의 최고 제한 속도 ==> " + Car.MAXSPEED);
12
13     System.out.println("PI의 값 ==> " + Math.PI);
14     System.out.println("3의 5제곱 ==> " + Math.pow(3, 5));
15   }
16 }
```

```
생산된 차의 대수(정적 필드) ==> 3
생산된 차의 대수(정적 메소드) ==> 3
차의 최고 제한 속도 ==> 200
PI의 값 ==> 3.141592653589793
3의 5제곱 ==> 243.0
```

클래스의 상속 ▶ 클래스 상속과 메소드 오버라이딩

❖ 상속 개념



[그림 3-7] 클래스의 상속 개념도

❖ 상속 예제

[예제 3-14] Automobile.java

```
1  public class Automobile extends Car {
2     int seatNum;
3
4     int getSeatNum() {
5        return seatNum;
6     }
7
8     void upSpeed(int value) {
9        if (speed + value >= 300)
10          speed = 300;
11       else
12          speed = speed + (int) value;
13    }
14 }
```

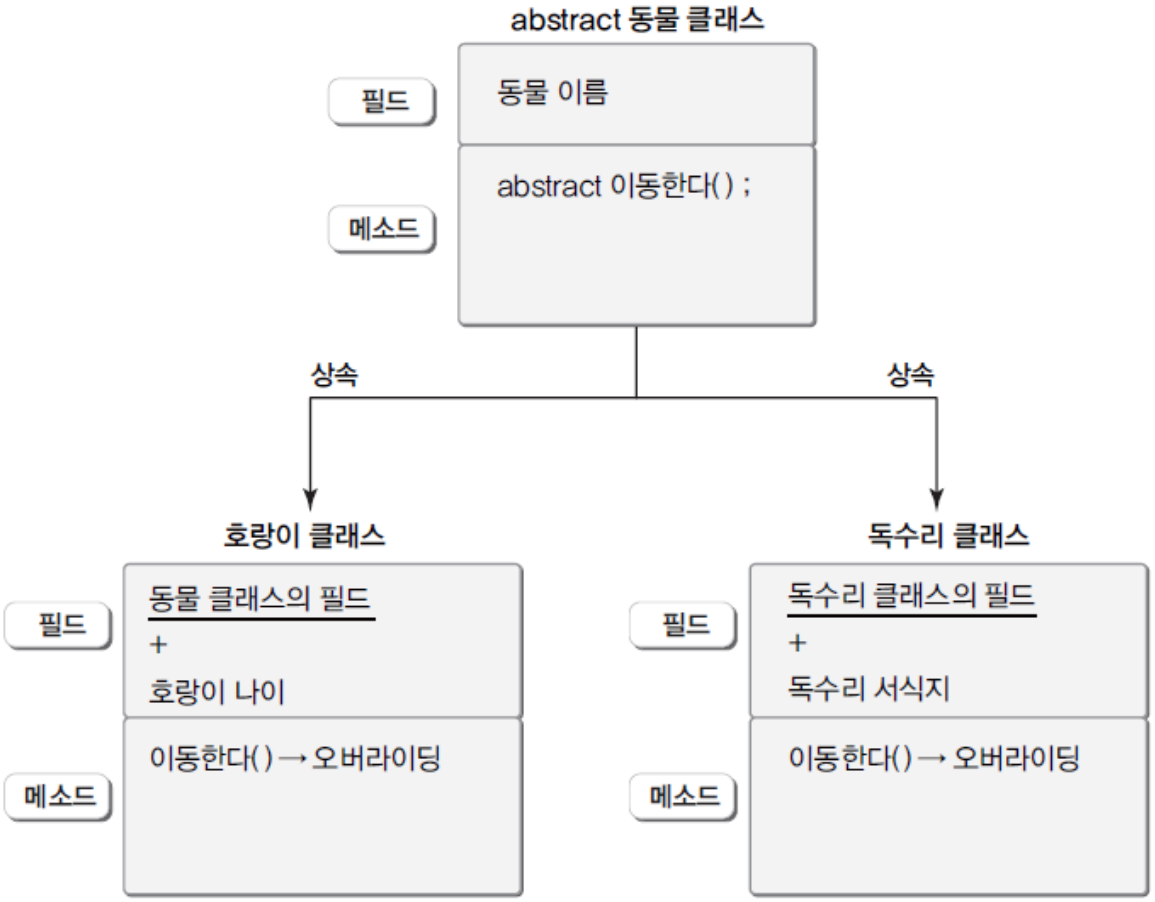
[예제 3-15] exam09.java

```
1  public class exam09 {
2     public static void main(String args[]) {
3        Automobile auto = new Automobile();
4
5        auto.upSpeed(250);
6        System.out.println("승용차의 속도는 "
7              + auto.getSpeed() + "km입니다.");
8     }
9  }
```

```
승용차의 속도는 250km입니다.
```

클래스의 상속 ▶ 추상 클래스와 추상 메소드

❖ 추상 클래스 개념



[그림 3-8] 추상 클래스의 상속 개념도

클래스의 상속 ▶ 추상 클래스와 추상 메소드

❖ 추상 클래스 예제

[예제 3-16] Animal.java

```
1  abstract class Animal {
2     String name;
3     abstract void move();
4  }
```

[예제 3-17] Tiger.java

```
1  class Tiger extends Animal {
2     int age;
3     void move() {
4        System.out.println("네 발로 이동한다.");
5     }
6  }
```

[예제 3-18] Eagle.java

```
1  class Eagle extends Animal {
2     String home;
3     void move() {
4        System.out.println("날개로 이동한다.");
5     }
6  }
```

[예제 3-19] exam10.java

```
1  public class exam10 {
2     public static void main(String args[]) {
3        Tiger tiger1 = new Tiger();
4        Eagle eagle1 = new Eagle();
5
6        tiger1.move();
7        eagle1.move();
8     }
9  }
```

```
네 발로 이동한다.
날개로 이동한다.
```

클래스의 상속 ▶ 클래스 변수의 다형성

❖ 다형성 예제

[예제 3-20] exam11.java

```
1  public class exam11 {
2    public static void main(String args[]) {
3      Animal animal;
4
5      animal = new Tiger();
6      animal.move();
7
8      animal = new Eagle();
9      animal.move();
10   }
11 }
```

```
네 발로 이동한다.
날개로 이동한다.
```

클래스의 상속 ▶ 인터페이스와 다중 상속

❖ 인터페이스와 다중 상속 예제

[예제 3-21] exam12.java

```
1  interface iAnimal {
2    abstract void eat();
3  }
4
5  public class exam12 {
6    public static void main(String args[]) {
7      iCat cat = new iCat();
8      cat.eat();
9
10     iTiger tiger = new iTiger();
11     tiger.move();
12     tiger.eat();
13   }
14
15   static class iCat implements iAnimal {
16     public void eat() {
17       System.out.println("생선을 좋아한다.");
18     }
19   }
20
21   static class iTiger extends Animal implements iAnimal {
22     void move() {
23       System.out.println("네 발로 이동한다.");
24     }
25
26     public void eat() {
27       System.out.println("멧돼지를 잡아 먹는다.");
28     }
29   }
30 }
```

```
생선을 좋아한다.
네 발로 이동한다.
멧돼지를 잡아 먹는다.
```

클래스의 상속 ▶ 익명 내부 클래스

❖ 익명 내부 클래스 예제

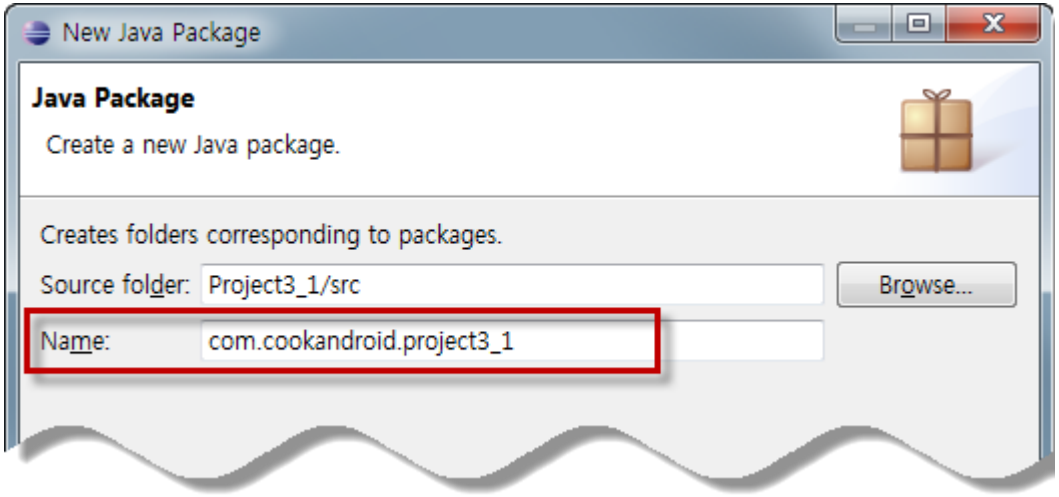
[예제 3-22] exam13.java

```
1  interface  clickListener {
2    public void print();
3  }
4
5  public class exam13 {
6    public static void main(String args[]) {
7
8      clickListener listener =
9        (new clickListener() {
10         public void print() {
11           System.out.println("클릭 리스너입니다. ");
12         }
13       });
14
15     listener.print();
16   }
17 }
```

```
클릭 리스너입니다.
```

기타 알아둘 Java 문법 ▶ 패키지

- **패키지 개요**
  - 관리를 위해서 클래스 및 인터페이스를 묶는 단위
  - 패키지 추가를 위해서 [New]-[Package] 선택
  - Java 파일 첫행에 “package 패키지명;” 추가



기타 알아둘 Java 문법 ▶ 제너릭스

❖ 제너릭스 개요

- 데이터 형식의 안전성을 보장

```
ArrayList strList = new ArrayList();
strList.add("첫 번째");
strList.add("두 번째");
strList.add(3);
```

```
ArrayList<String> strList = new ArrayList<String>();
strList.add("첫 번째");
strList.add("두 번째");
strList.add(3);
```

- 컴파일 오류를 발생시킴
- <String>외에도 다른 형식도 사용 가능함

기타 알아둘 Java 문법 ▶ 데이터 변환 등

- **데이터 변환, 문자열 비교, 날짜 형식**
  - ✓ **날짜를 표현하기 위해 DateFormat 클래스를 사용**
  - ✓ **SimpleDateFormat을 사용하면 “연월일”이나 “시분초” 같은 표현 가능**

```
int a = Integer.parseInt("100");
double b = Double.parseDouble("100.123");
```

```
String str = "안녕하세요";
if (str.equals((String)"안녕하세요" )) {
      // 문자열이 같으면 이곳을 수행
}
```

```
Date now = new Date();
SimpleDateFormat sFormat;

sFormat = new SimpleDateFormat("yyyyMMdd");
System.out.println(sFormat.format(now)); // 20121131 형식으로 출력

sFormat = new SimpleDateFormat("HH:mm:ss");
System.out.println(sFormat.format(now)); // 23:15:21 형식으로 출력
```