

제01장

자바 시작하기

구디아카데미 ▷ 민경태 강사

학습목표

1. 자바프로젝트의 구성요소에 대해서 알 수 있다.
2. 자바프로젝트를 추가하고 실행할 수 있다.
3. 주석문을 작성할 수 있다.
4. 출력문을 작성할 수 있다.

자바 시작하기

```
each: function(e, t, n) {  
    o = e.length,  
    a = M(e);  
    if (n) {  
        if (a) {  
            for (; o > i; i++)  
                if (r = t.apply(e[i], n), r ===  
            ) else  
                for (i in e)  
                    if (r = t.apply(e[i], n), r ===  
        ) else if (a) {  
            for (; o > i; i++)  
                if (r = t.call(e[i], i, e[i])  
            ) else  
                for (i in e)  
                    if (r = t.call(e[i], i, e[i])  
        return e  
    },  
    trim: b && !b.call("\uffff\u00a0") ?  
        return null == e ? "" : b.call(  
    } : function(e) {  
        return null == e ? "" : (e +  
    },  
    makeArray: function(e, t) {  
        var n = t || [];  
        return null != e && (M(Obj  
    },  
    isArray: function(e, t, n) {  
        var r;  
        if (t) {  
            if (m) return m.c  
            for (n = t.length  
                if (n in t  
        }  
    }
```

목차

1. 자바프로젝트의 구성요소
2. 자바프로젝트의 추가와 실행
3. 주석문
4. 출력문

```

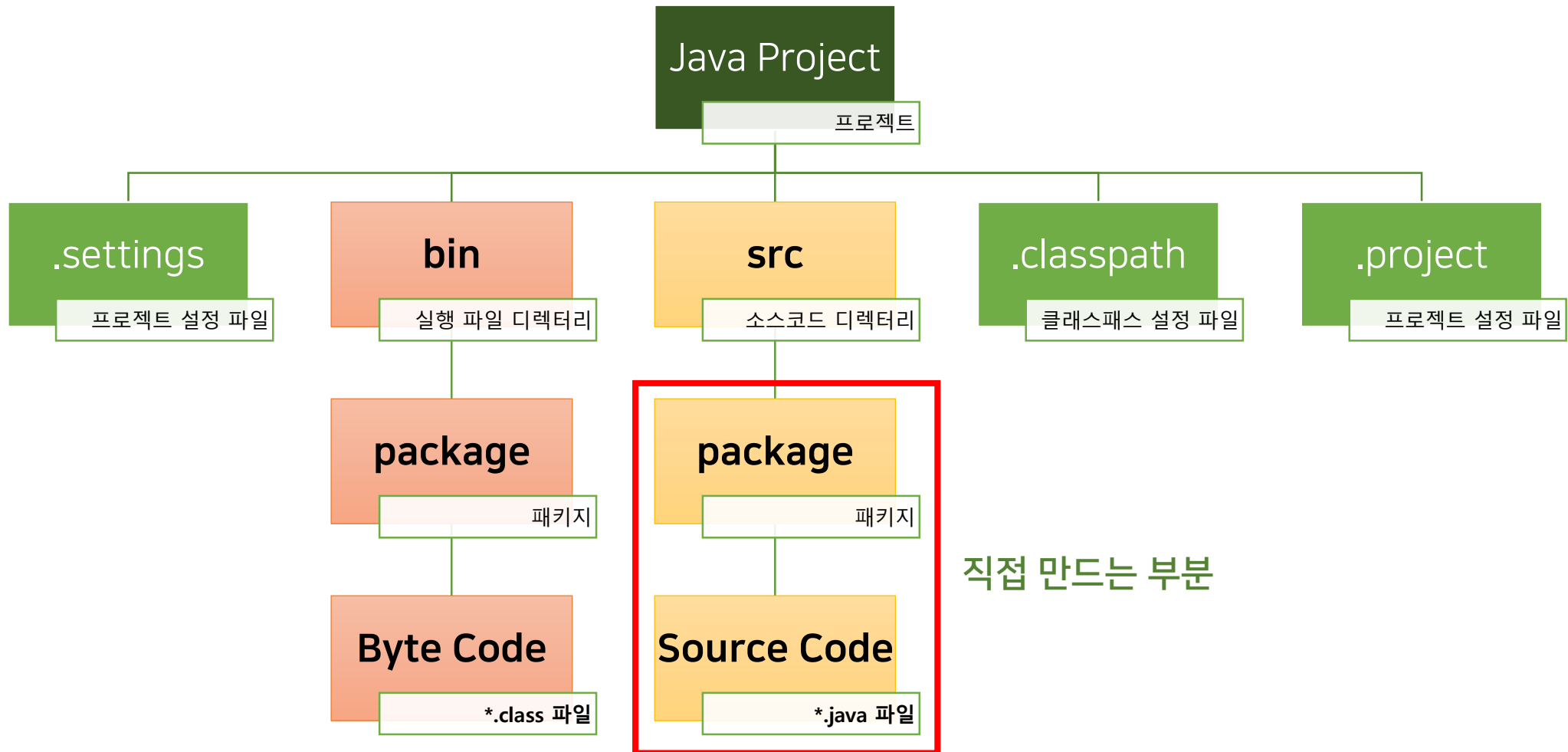
each: function(e, t, n) {
  var r, i = 0,
      o = e.length,
      a = M(e);
  if (n) {
    if (a) {
      for (; o > i; i++)
        if (r = t.apply(e[i], n), r === !1) break;
    } else
      for (i in e)
        if (r = t.apply(e[i], n), r === !1) break;
  } else if (a) {
    for (; o > i; i++)
      if (r = t.call(e[i], e[i]), r === !1) break;
  } else
    for (i in e)
      if (r = t.call(e[i], e[i]), r === !1) break;
  return e;
},
trim: b && !b.call("\uffff\u00a0") ? function(e) {
  return null == e ? "" : b.call(e);
} : function(e) {
  return null == e ? "" : (e + "").replace(C, "");
},
makeArray: function(e, t) {
  var n = t || [];
  return null != e && (M(Object(e)) ? x.merge(n, "string"
),
isArray: function(e, t, n) {
  var r;
  if (t) {
    if (n) return m.call(t, e, n);
    for (r = t.length, r = r ? 0 > n ? Math.max(0, r + n) :
      if (n in t && t[n] === e) return n;
  }
}

```

1. 자바프로젝트의 구성요소

자바프로젝트의 구성요소

■ 자바프로젝트 구성요소



자바프로젝트의 구성요소

■ src

- 소스 코드를 작성하는 폴더
- 패키지를 만든 다음 클래스를 추가하고 클래스 내부에 소스 코드를 작성함

src

└─ package

└─ source code

■ bin

- 소스 코드가 컴파일된 바이트 코드(*)가 저장되는 폴더
- src 폴더에서 작성한 패키지와 클래스를 기반으로 생성됨

* 바이트 코드

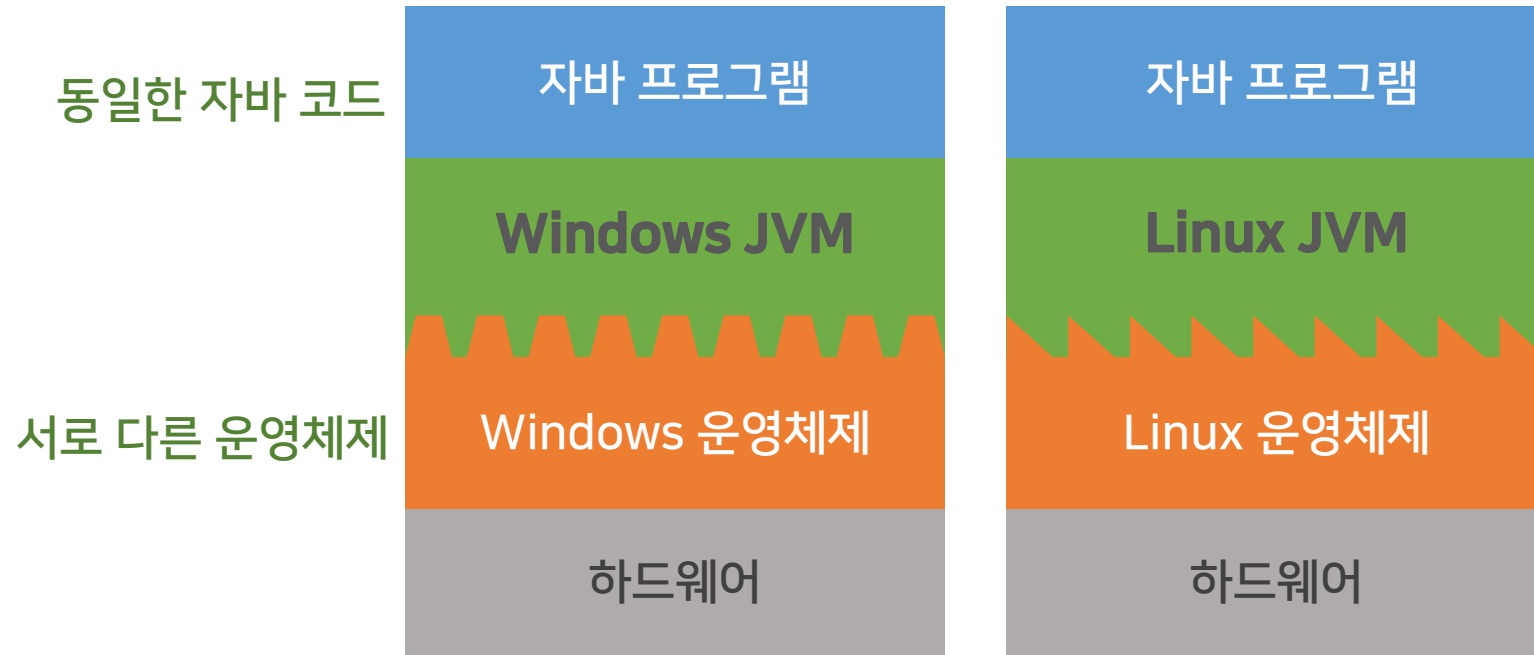
- ▶ 소스 코드를 JVM이 이해할 수 있는 언어로 변환한 상태의 코드. 실행 가능한 상태의 코드를 의미한다.

JVM

- Java Virtual Machine
- 컴파일된 바이트 코드를 실행하는 자바 가상 머신
- 자바로 작성된 모든 프로그램은 JVM에 의해서 실행됨
- JVM만 설치되어 있으면 자바 프로그램은 어떤 운영체제에서도 실행 가능

JVM

- 운영체제가 다르더라도 운영체제에 따른 JVM이 별도로 존재하므로 모든 자바 코드는 동일하게 작성할 수 있음




```

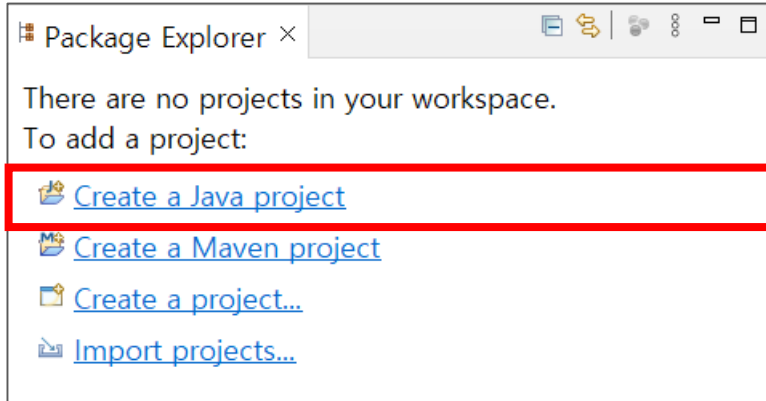
each: function(e, t, n) {
  var r, i = 0,
      o = e.length,
      a = M(e);
  if (n) {
    if (a) {
      for (; o > i; i++)
        if (r = t.apply(e[i], n), r === !1) break;
    } else
      for (i in e)
        if (r = t.apply(e[i], n), r === !1) break;
  } else if (a) {
    for (; o > i; i++)
      if (r = t.call(e[i], e[i]), r === !1) break;
  } else
    for (i in e)
      if (r = t.call(e[i], e[i]), r === !1) break;
  return e;
},
trim: b && !b.call("\uffff\u00a0") ? function(e) {
  return null == e ? "" : b.call(e);
} : function(e) {
  return null == e ? "" : (e + "").replace(C, "");
},
makeArray: function(e, t) {
  var n = t || [];
  return null != e && (M(Object(e)) ? x.merge(n, "string"
),
isArray: function(e, t, n) {
  var r;
  if (t) {
    if (n) return m.call(t, e, n);
    for (r = t.length, r = r ? 0 > n ? Math.max(0, r + n) : n; n in t && t[n] === e) return n;
  }
}

```

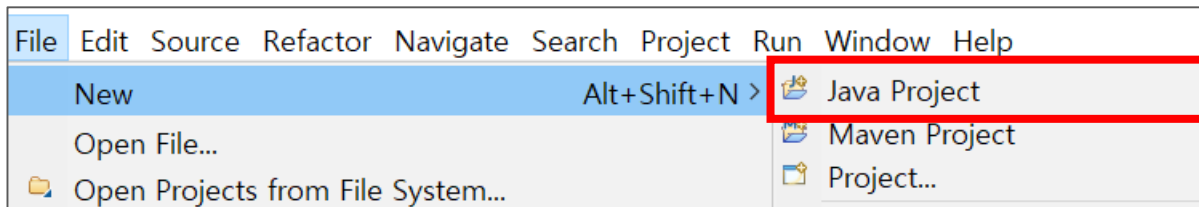
2. 자바프로젝트의 추가와 실행

자바프로젝트 추가

- Create a Java Project



- [File] - [New] - [Java Project]



자바프로젝트 추가

- 프로젝트 이름 입력
lec01_start
- 설치된 자바 버전 선택
JavaSE-17

New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: lec01_start

☒ Use default location

Location: C:\java lec\workspace\lec01_start [Browse...](#)

JRE

☒ Use an execution environment JRE: JavaSE-17

☐ Use a project specific JRE: jre

☐ Use default JRE 'jre' and workspace compiler preferences [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

[?](#) [< Back](#) [Next >](#) [Finish](#) [Cancel](#)

자바프로젝트 추가

- 모듈 생성 해제 후
Next >

① 체크 해제

New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

☐ Use default JRE 'jre' and workspace compiler preferences [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

Working sets

☐ Add project to working sets [New...](#)

Working sets: [Select...](#)

Module

☐ Create module-info.java file

Module name:

☐ Generate comments

[? < Back](#) [Next >](#) [Finish](#) [Cancel](#)

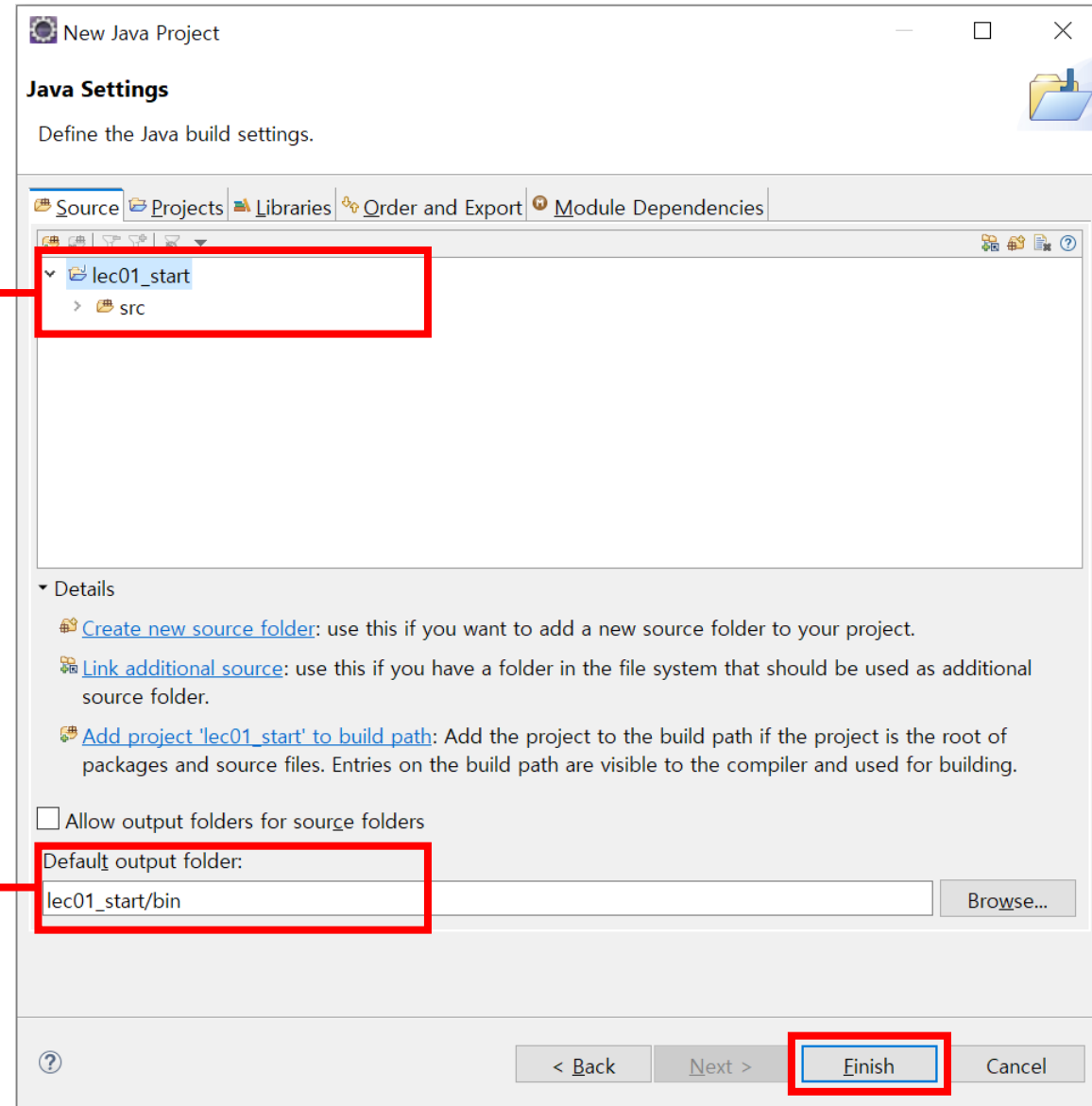
② 클릭

자바프로젝트 추가

■ Finish

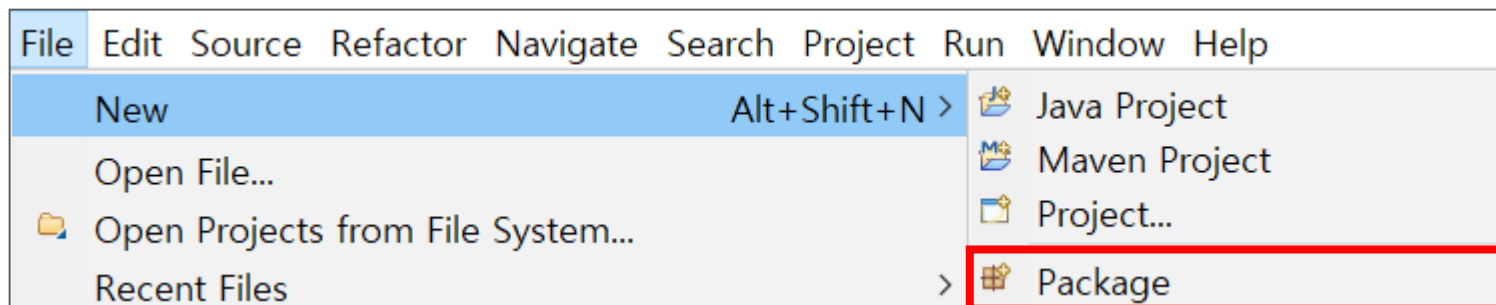
src 폴더

bin 폴더



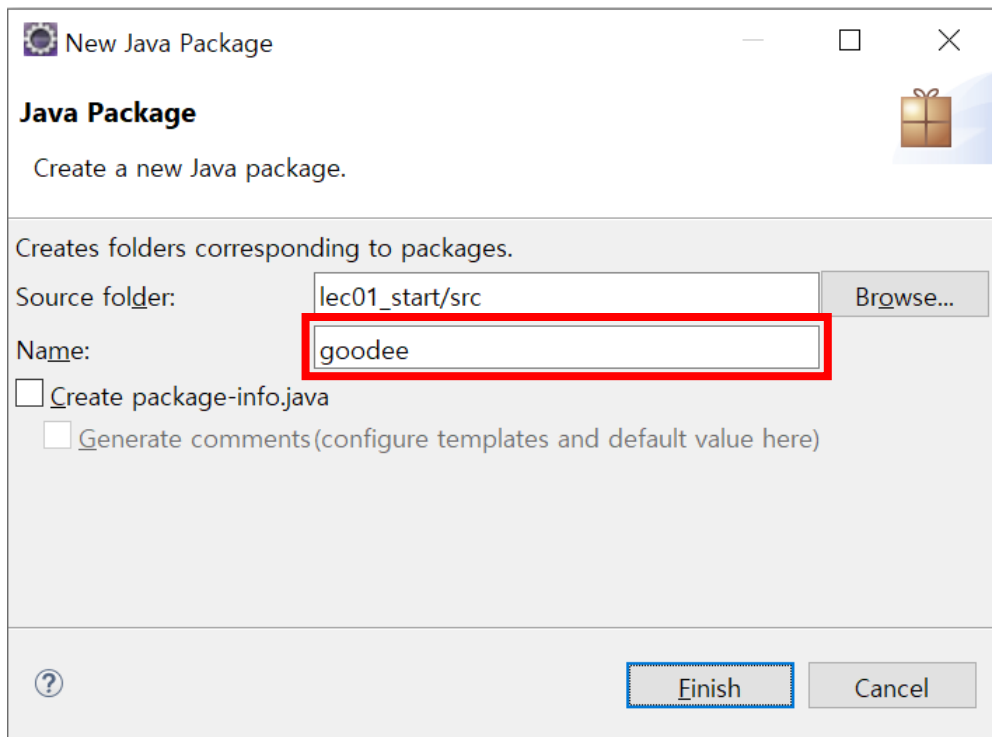
패키지 추가

- 패키지는 자바클래스를 저장하는 폴더의 개념
- 자바클래스의 역할에 따라 패키지를 구분해서 저장함
- [File] - [New] - [Package]

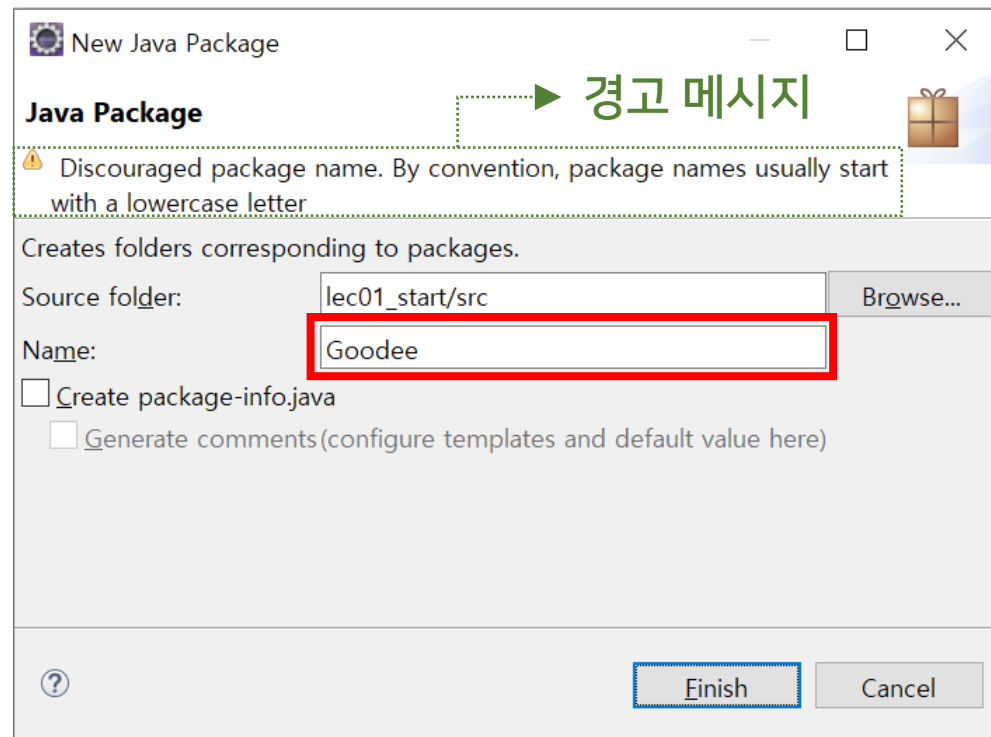


패키지 추가

- 패키지는 소문자로 시작하는 이름을 가져야 함



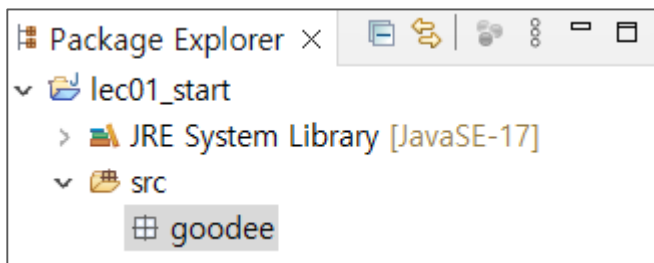
문제 없음



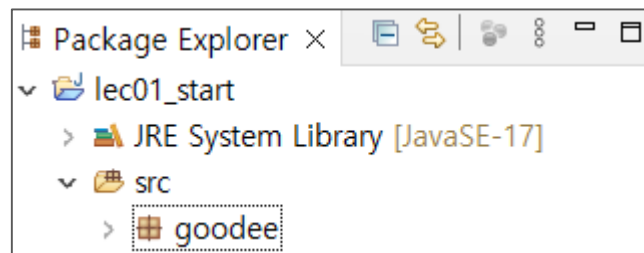
대문자로 시작하기 때문에 경고 메시지가 나타남

패키지 추가

- 패키지에 포함된 자바클래스 유무에 따라 색상이 다르게 표시됨



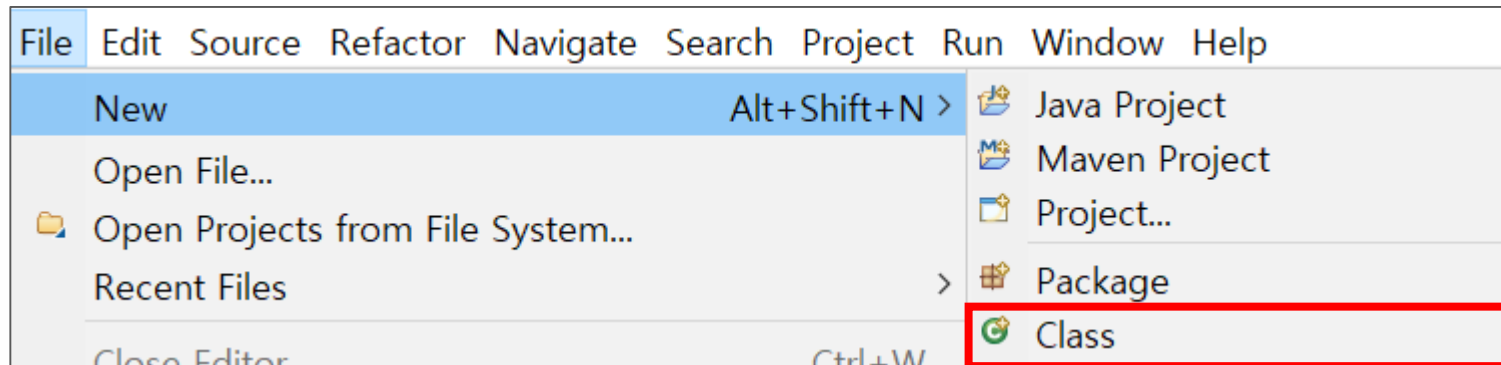
비어 있는 패키지



자바클래스가 포함된 패키지

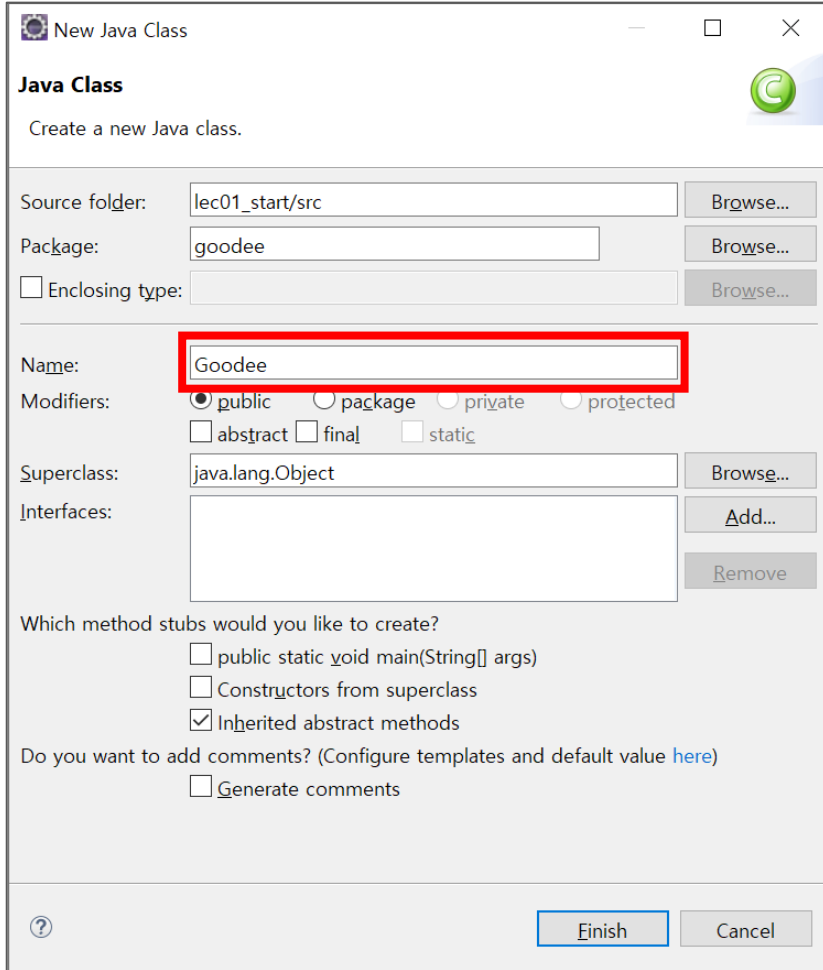
클래스 추가

- 자바는 모든 소스 코드를 클래스 내부에 작성해야 함
- 클래스를 만들면 클래스 이름과 동일한 자바파일(*.java)이 생성됨
- [File] - [New] - [Class]



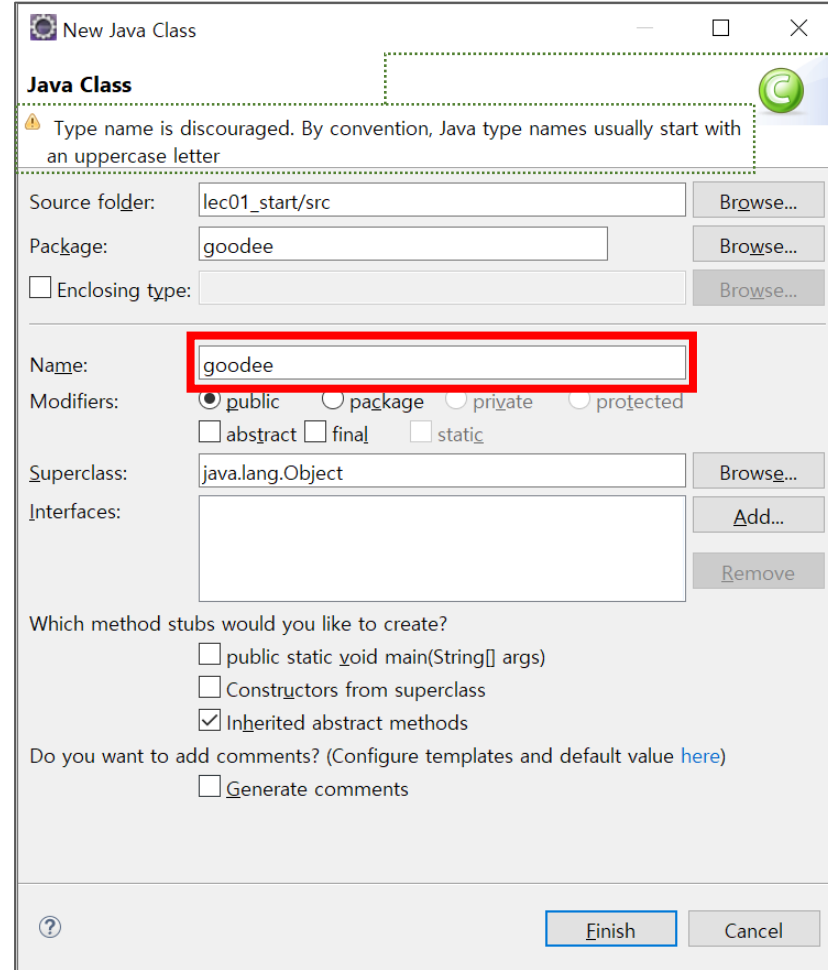
클래스 추가

- 클래스는 대문자로 시작하는 이름을 가져야 함



The dialog box 'New Java Class' is shown. The 'Name' field contains 'Goodee' and is highlighted with a red rectangle. The 'Package' field contains 'goodee'. The 'Source folder' is 'lec01_start/src'. The 'Enclosing type' is empty. The 'Superclass' is 'java.lang.Object'. The 'Interfaces' field is empty. The 'Which method stubs would you like to create?' section has 'Inherited abstract methods' checked. The 'Do you want to add comments?' section has 'Generate comments' unchecked. The 'Finish' button is highlighted.

문제 없음



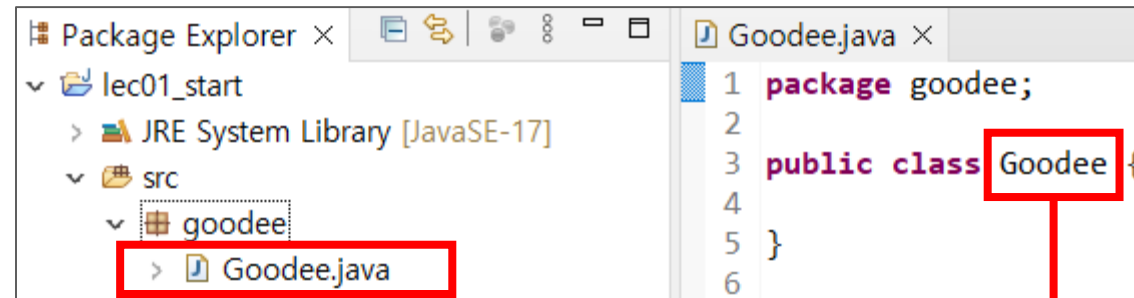
The dialog box 'New Java Class' is shown. The 'Name' field contains 'goodee' and is highlighted with a red rectangle. The 'Package' field contains 'goodee'. The 'Source folder' is 'lec01_start/src'. The 'Enclosing type' is empty. The 'Superclass' is 'java.lang.Object'. The 'Interfaces' field is empty. The 'Which method stubs would you like to create?' section has 'Inherited abstract methods' checked. The 'Do you want to add comments?' section has 'Generate comments' unchecked. The 'Finish' button is highlighted. A warning message is displayed at the top: 'Type name is discouraged. By convention, Java type names usually start with an uppercase letter'. A green dashed box highlights the warning message, and a green arrow points from the text '경고 메시지' to it.

경고 메시지

소문자로 시작하기 때문에 경고 메시지가 나타남

클래스 추가

- 클래스가 추가된 자바프로젝트



파일 이름과 클래스 이름이 같아야 한다.

```

each: function(e, t, n) {
    var r, i = 0,
        o = e.length,
        a = M(e);
    if (n) {
        if (a) {
            for (; o > i; i++)
                if (r = t.apply(e[i], n), r === !1) break;
        } else
            for (i in e)
                if (r = t.apply(e[i], n), r === !1) break;
    } else if (a) {
        for (; o > i; i++)
            if (r = t.call(e[i], e[i]), r === !1) break;
    } else
        for (i in e)
            if (r = t.call(e[i], e[i]), r === !1) break;
    return e;
},
trim: b && !b.call("\uffff\u00a0") ? function(e) {
    return null == e ? "" : b.call(e);
} : function(e) {
    return null == e ? "" : (e + "").replace(C, "");
},
makeArray: function(e, t) {
    var n = t || [];
    return null != e && (M(Object(e)) ? x.merge(n, "string"
),
isArray: function(e, t, n) {
    var r;
    if (t) {
        if (n) return m.call(t, e, n);
        for (r = t.length, r = n ? 0 > n ? Math.max(0, r + n) : r; r--;)
            if (n in t && t[r] === e) return r;
    }
}

```

3. 주식문

- comment
- 소스코드의 설명을 작성하기 위한 부분으로 자바에 의해 실행되지 않는 영역
- 자바는 3가지 형식의 주석을 가짐
 1. single-line comment
 2. multi-line comment
 3. documentation comment

single-line comment

- 주석을 한 줄만 작성하는 경우에 사용
- `//` 로 시작

```
public class Goodee {  
    // single-line comment  
}
```

multi-line comment

- 주석을 여러 줄로 작성하는 경우에 사용
- `/*`로 시작하고 `*/`로 끝남

```
public class Goodee {  
    /*  
     * multi-line comment  
     */  
}
```

documentation comment

- 클래스, 메소드, 인수 등에 관한 자바 API 문서를 작성하기 위해서 사용하고 내부에서는 전용 Annotation을 사용할 수 있음
- `/**`로 시작하고 `*/`로 끝남

```
public class Goodee {  
    /**  
     * documentation comment  
     */  
}
```



```

each: function(e, t, n) {
    var r, i = 0,
        o = e.length,
        a = M(e);
    if (n) {
        if (a) {
            for (; o > i; i++)
                if (r = t.apply(e[i], n), r === !1) break;
        } else
            for (i in e)
                if (r = t.apply(e[i], n), r === !1) break;
    } else if (a) {
        for (; o > i; i++)
            if (r = t.call(e[i], e[i]), r === !1) break;
    } else
        for (i in e)
            if (r = t.call(e[i], e[i]), r === !1) break;
    return e;
},
trim: b && !b.call("\uffff\u00a0") ? function(e) {
    return null == e ? "" : b.call(e);
} : function(e) {
    return null == e ? "" : (e + "").replace(C, "");
},
makeArray: function(e, t) {
    var n = t || [];
    return null != e && (M(Object(e)) ? x.merge(n, "string"
    ),
inArray: function(e, t, n) {
    var r;
    if (t) {
        if (m) return m.call(t, e, n);
        for (r = t.length, r = r ? 0 > n ? Math.max(0, r + n) : n; r in t && t[r] === e) return r;
    }
}

```

4. 출력문

main 메소드

- JVM이 자바프로그램의 실행을 시작하는 시작포인트
- main 메소드가 없으면 JVM은 자바프로그램을 실행하지 않음
- 형식

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

출력문

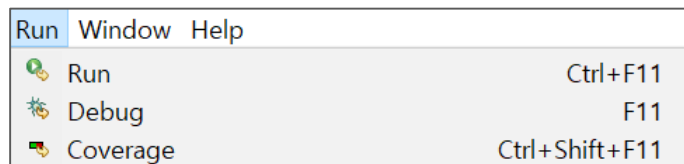
- Console View에 실행 결과를 나타낼 때 사용함
- `System.out.println()`
가장 많이 사용하는 출력문으로 출력 후 자동으로 줄 바꿈 처리됨
- `System.out.print()`
출력 후 자동으로 줄 바꿈 처리가 되지 않음
- `System.out.printf()`
각종 format을 활용하여 출력 형식을 지정할 때 사용함

출력문

■ 코드

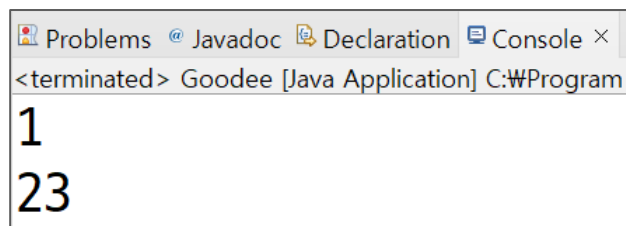
```
public class Goodee {  
    public static void main(String[] args) {  
        System.out.println(1);           // 1 출력 후 줄 바꿈  
        System.out.print(2);             // 2 출력  
        System.out.printf("%d", 3);      // 3 출력 (%d는 정수를 의미)  
    }  
}
```

■ 실행



[Run] - [Run] 또는 Ctrl + F11

■ 실행결과



이클립스 주요 단축키

■ 자동 완성

- Ctrl + Space Bar
- 사용 예시
 - sysout System.out.println()
 - try try { } catch(Exception e) { }

■ 주석

- Ctrl + / 한 줄 주석(//) 설정 및 해제
- Ctrl + Shift + / 여러 줄 주석(/* */) 설정
- Ctrl + Shift + W 여러 줄 주석(/* */) 해제

■ 실행

- Ctrl + F11 실행 (에러 발생 시 디버깅 안 함)
- F11 실행 (에러 발생 시 디버깅 함)

이클립스 주요 단축키

■ 저장

- Ctrl + s
- Ctrl + Shift + s

현재 파일 저장

열려 있는 모든 파일 저장

■ 소스코드 편집

- Ctrl + z
- Ctrl + y
- Ctrl + d
- Ctrl + Shift + x
- Ctrl + Shift + y
- Ctrl + Shift + f
- Ctrl + Shift + o
- Alt + ↑ ↓
- Ctrl + Alt + ↑ ↓

작업 취소

작업 취소의 취소 (다시 작업)

한 줄 삭제

대문자로 변환

소문자로 변환

소스코드 자동 정렬

import 자동 정리

소스코드 위/아래로 이동

소스코드 위/아래로 복사