

# IntelliJ IDEA 단축키 15가지

## 1. Alt + Enter

```
private final String[] stringArray = new String[] { "Hello", "World" };
private List<Converter> converters;

private void error() {
    List<String> strings = new HashMap<Integer, String>();
}

public void lambdas() {
    //Remove redundant types
    Function<Function, Function> f3 =
        (Function function) -> function.compose(function)

    //Lambda can be replaced with method reference
}
```

- 무엇이든 수정 가능하다.
  - 코드에서 오류가 발견된 경우 해당 오류 위에 커서를 놓고 Alt + Enter를 누르면 문제에 대한 수정 제안 목록이 표시됨
  - 경고 및 제안이 표시되는 곳에서도 수정 제안 목록이 표시됨
  - 오류, 경고, 제안이 없는 코드에서도 Alt + Enter를 사용하여 좀더 좋은 코드로 수정해주기도 함

## 2. F2

```
private void error() {  
    List<String> strings  
        = new HashMap<Integer, String>();  
}  
  
public void lambdas() {  
    //Lambda can be replaced with method reference  
    Arrays.sort(stringArray, (s1, s2) -> s1.compareToIgnoreCase(s2));  
  
    //Replace with forEach on foo  
    ArrayList<String> foo = getStrings();  
    for (String s : foo) {
```

- 다음 오류, 경고, 제안으로 이동한다.

### 3. Cmd + 1(Mac) 또는 Alt + 1(Windows)

```

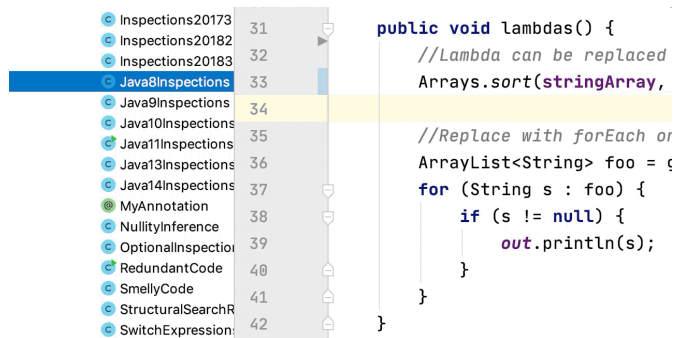
29     }
30
31     public void lambdas() {
32         //Lambda can be replaced with method reference
33         Arrays.sort(stringArray, String::compareToIgnoreCase);
34
35         //Replace with forEach on foo
36         ArrayList<String> foo = getStrings();
37         for (String s : foo) {
38             if (s != null) {
39                 out.println(s);
40             }
41         }
42     }
43 }

```

- 프로젝트 창을 열수있다.

- 평소에 전체 화면으로 코드 에디터를 사용하신다면 Cmd+1 또는 Alt+1 을 눌러 프로젝트 창을 열고 초점을 이동할 수 있음.
- 방향키를 사용해 트리를 탐색하고, 타이핑을 하여 검색할 수 있음.

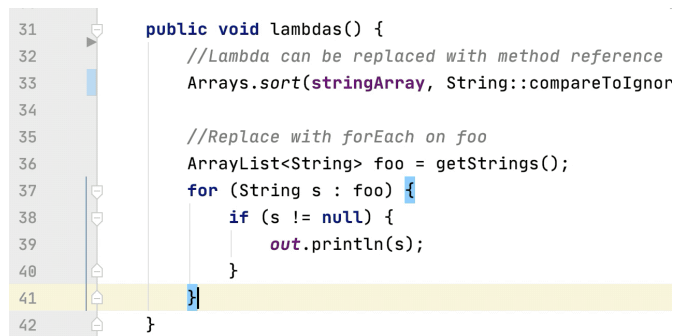
#### 4. Esc



- 에디터에 다시 포커스가 위치하게 하려면 Esc를 누르면된다.

- 평소에 전체 화면으로 코드 에디터를 사용하신다면 Cmd+1 또는 Alt+1 을 눌러 프로젝트 창을 열고 초점을 이동할 수 있음.
- 방향키를 사용해 트리를 탐색하고, 타이핑을 하여 검색할 수 있음.

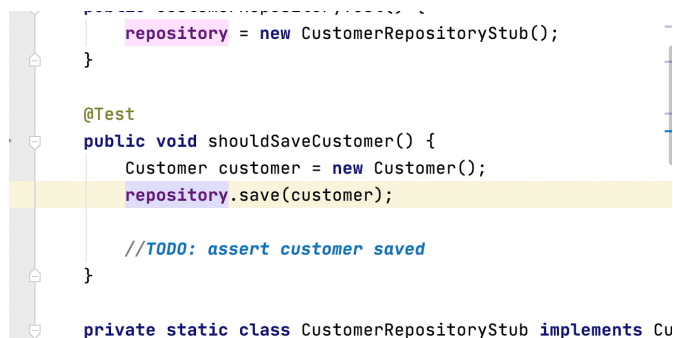
#### 5. Cmd+E(Mac) 또는 Ctrl+E(Windows)



- 가장 최근 열었던 파일을 확인할 수 있다.

- 이 단축키를 누르면 최근 파일 대화상자가 열려 방향키로 파일을 탐색할 수 있음.
- 이곳에서 도구 창을 열 수도 있으며, 키보드 단축키가 없는 도구 창도 열 수 있음.

#### 6. Cmd+B(Mac) 또는 Ctrl+B (Windows)



○ 심볼의 선언으로 이동할 수 있다.

- 필드에서 Cmd+B 또는 Ctrl+B(Windows)를 누르면 커서가 필드 선언으로 이동함.
- 클래스 이름에서 Cmd+B 또는 Ctrl+B를 누르면 클래스 파일로 이동함.
- Alt+Cmd+B(Mac) 또는 Ctrl+Alt+B(Windows)
- 구현을 탐색하는 것도 가능하다.

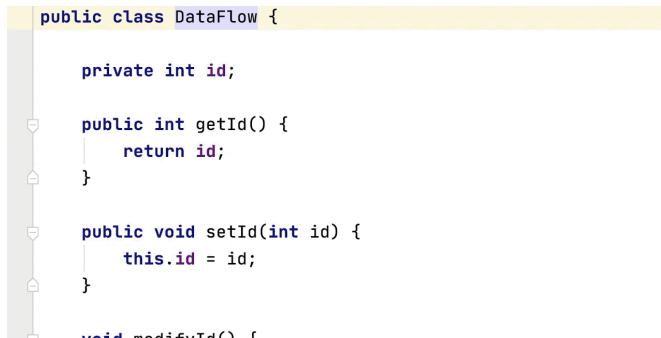
## 7. Alt+F7



○ 모든 사용 위치가 표시된다.

- 예를 들어, 인터페이스 이름에서 Alt+F7 를 누르면 필드 선언 또는 인터페이스를 구현한 클래스를 비롯하여 해당 인터페이스가 사용된 모든 위치가 검색창에 표시 됨.

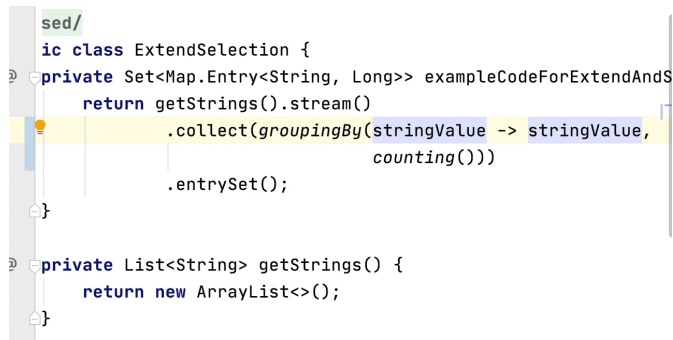
## 8. Ctrl 두 번



○ 무엇이든 실행 가능하다.

- Ctrl 을 두 번 누르면 Run Anything 창이 열림.(기본적으로 이 창에서는 최근 실행한 실행 구성 (run configurations)목록이 표시됨.)

## 9. Alt+위/아래 방향키(Mac) 또는 Ctrl+W, Ctrl+Shift+W (Windows)



```

sed/
ic class ExtendSelection {
private Set<Map.Entry<String, Long>> exampleCodeForExtendAndS
return getStrings().stream()
.collect(groupingBy(stringValue -> stringValue,
counting()))
.entrySet();
}

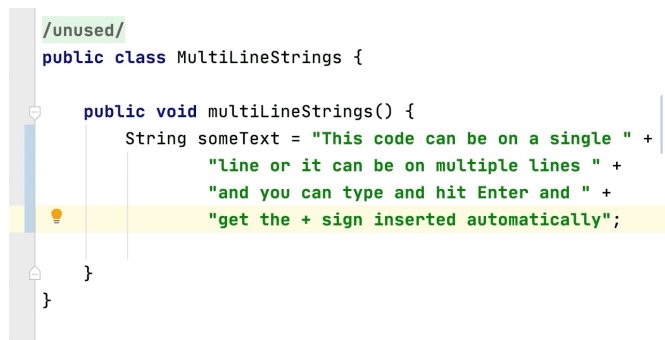
private List<String> getStrings() {
return new ArrayList<>();
}

```

○ 커서 주변의 코드 섹션을 확장하거나 축소할 수 있다.

- 선택 영역 확장을 사용하면 IntelliJ IDEA가 확장된 섹션의 유효한 표현식을 자동으로 선택함.
- Alt+아래 방향키 또는 Ctrl+Shift+W
  - 유효한 Java 블록을 통해 선택 영역이 축소되어 커서로 돌아온다.

## 10. Cmd+/(Mac) 또는 Ctrl+/(Windows)



```

/unused/
public class MultiLineStrings {

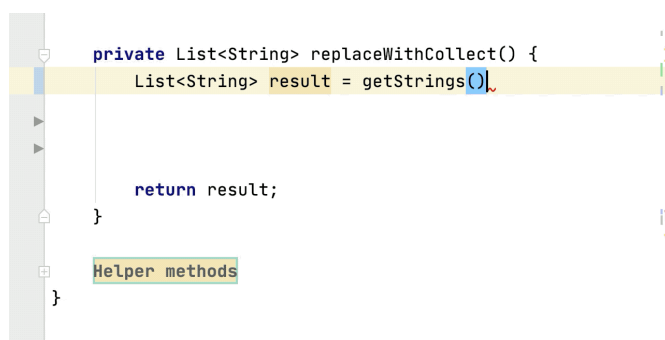
public void multilineStrings() {
String someText = "This code can be on a single " +
"line or it can be on multiple lines " +
"and you can type and hit Enter and " +
"get the + sign inserted automatically";
}
}

```

○ 간편하게 주석을 추가할 수 있다.

- 줄 어디서든 Cmd+/ (Windows의 경우 Ctrl+/)를 누르면 코드가 줄 주석으로 처리됨.
- 이미 주석 처리된 줄에서 같은 단축키를 다시 누르면 주석 처리가 해제됨
- Alt+Cmd+/(Mac) 또는 Shift+Ctrl+/(Windows)
  - 블록 주석을 추가할 수 있다.
  - 이 단축키를 다시 누르면 블록 주석이 제거됨.

## 11. Shift+Cmd+Enter(Mac) 또는 Shift+Ctrl+Enter(Windows)



```

private List<String> replaceWithCollect() {
List<String> result = getStrings();

return result;
}

Helper methods
}

```

○ 현재 구문을 완성시켜준다.

- 코드 끝에 세미콜론을 추가시켜줌.
- for 루프를 작성 할 때에 Shift+Cmd+Enter 를 누르면 IntelliJ IDEA가 중괄호를 추가하고 커서를 적절한 위치에 배치함.
- if 문의 경우 괄호 및 중괄호를 추가하고 커서가 올바른 위치에 놓임.

## 12. Alt+Cmd+L(Mac) 또는 Ctrl+Alt+L(Windows)

```
}  
  
public void horriblyFormattedMethod () {  
    System.out.println("First line");  
    System.out.println("Second line");  
    System.out.println("Third line");  
    for (int i = 0; i < 3; i++)  
        System.out.println("Where?");  
}
```

○ 간편하게 현재 파일의 서식을 프로젝트 표준에 맞춰준다.

- 변경이 된 줄의 서식을 지정하는 데 사용될 수 있음.

## 13. Ctrl+T(Mac) 또는 Shift+Ctrl+Alt+T(Windows)

```
String hotkey = "Ctrl/Cmd+Alt+M";  
  
String[] steps = new String[5];  
steps[0] = "First select a block of code";  
steps[1] = "Then press " + hotkey;  
steps[2] = "Give the method a fullName";  
steps[3] = "Assign it a visibility";  
steps[4] = "Apply the refactoring";  
  
System.out.println(s);
```

○ 사용 가능한 리팩토링 옵션이 표시된다.

- 방향 키를 사용해 하나를 선택하여 입력하거나, 리팩토링 왼쪽에 있는 숫자를 사용해 선택할 수 있음.

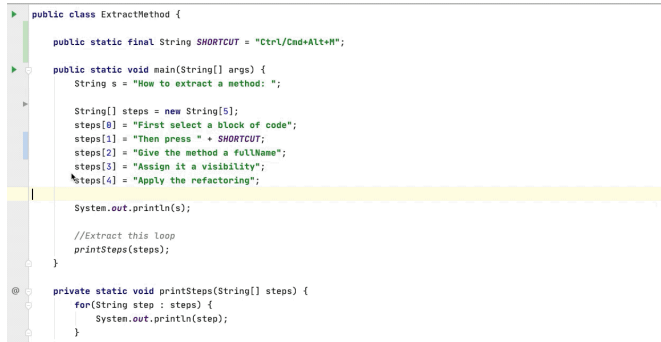
## 14. Shift+Cmd+A(Mac) 또는 Shift+Ctrl+A(Windows)

```
package com.jethrains.refactoring;  
  
public class ExtractMethod {  
    public static final String SHORTCUT = "Ctrl/Cmd+Alt+H";  
  
    public static void main(String[] args) {  
        String s = "How to extract a method: ";  
  
        String[] steps = new String[5];  
        steps[0] = "First select a block of code";  
        steps[1] = "Then press " + SHORTCUT;  
        steps[2] = "Give the method a fullName";  
        steps[3] = "Assign it a visibility";  
        steps[4] = "Apply the refactoring";  
  
        System.out.println(s);  
  
        //Extract this loop  
        printSteps(steps);  
    }  
  
    private static void printSteps(String[] steps) {  
        for (String step : steps) {  
            System.out.println(step);  
        }  
    }  
}
```

○ IntelliJ IDEA에서 모든 액션을 검색할 수 있습니다.

- 드롭다운 메뉴에 액션뿐만 아니라 단축키도 표시되므로 이 단축키를 배우고 연습할 수 있음.
- 액션 검색을 통해 액션뿐만 아니라 설정도 검색할 수 있으므로, 여기서 바로 설정을 변경할 수 있음.
- 도구 창을 검색하고 열 수도 있음.

## 15. Shift 두 번



```
public class ExtractMethod {  
    public static final String SHORTCUT = "Ctrl/Cmd+Alt+H";  
  
    public static void main(String[] args) {  
        String s = "How to extract a method: ";  
  
        String[] steps = new String[5];  
        steps[0] = "First select a block of code";  
        steps[1] = "Then press " + SHORTCUT;  
        steps[2] = "Give the method a full name";  
        steps[3] = "Assign it a visibility";  
        steps[4] = "Apply the refactoring";  
  
        System.out.println(s);  
  
        //Extract this loop  
        printSteps(steps);  
    }  
  
    private static void printSteps(String[] steps) {  
        for(String step : steps) {  
            System.out.println(step);  
        }  
    }  
}
```

○ Shift 키를 두 번 누르면 모든 항목을 검색할 수 있는 검색 상자가 열립니다.

- 검색할 항목을 입력하면 클래스, 파일, 심볼 및 액션에 대한 결과가 표시됨.

---

## 참고한 곳

- [https://blog.jetbrains.com/ko/2020/03/11/top-15-intellij-idea-shortcuts\\_ko/](https://blog.jetbrains.com/ko/2020/03/11/top-15-intellij-idea-shortcuts_ko/)
- [https://www.youtube.com/watch?v=QYO5\\_riePOQ](https://www.youtube.com/watch?v=QYO5_riePOQ)