

2024년 상반기 K-디지털 트레이닝

TodoApp

[KB] IT's Your Life

✓ 프로젝트 생성

New Project

Search:

New Project

- Java
- Kotlin
- Groovy
- Empty Project

Generators

- Maven Archetype
- Jakarta EE
- Spring Boot
- JavaFX

Name:

Location:

Project will be created in: C:\WKB_Fullstack\W08_Spring\TodoApp

☐ Create Git repository

Build system:

JDK: 17 java version "17"

Gradle DSL:

Advanced Settings

Gradle distribution:

Gradle version: ☒ Auto-select

☒ Use these settings for future projects

GroupId:

ArtifactId:

Compose for Desktop

- HTML
- React
- Express
- Angular CLI
- Vue.js
- Vite

Templates

[More via plugins...](#)

1 TodoApp

settings.gradle

```
rootProject.name = 'TodoApp'
include ':ScoulaCli'
project(':ScoulaCli').projectDir=new File('C:\\KB_Fullstack\\04_Java\\ScoulaLib\\scoulacli')
```


build.gradle

```
...
dependencies {
    implementation project(':ScoulaCli')

    testImplementation platform('org.junit:junit-bom:5.10.0')
    testImplementation 'org.junit.jupiter:junit-jupiter'
}
...
```

✓ Lombok

- mvnrepository.com에서 Lombok 검색



Project Lombok » 1.18.32

Lombok is a Java library that provides annotations to simplify Java development by automating the generation of boilerplate code. Key features include automatic generation of getters, setters, equals, hashCode, and toString methods, as well as a facility for automatic resource management. It aims to reduce the amount of manual coding, thereby streamlining the codebase and reducing potential for errors.

License	MIT
Categories	Code Generators
Tags	lombok codegen code generator
HomePage	https://projectlombok.org
Date	Mar 20, 2024
Files	pom (1 KB) jar (2.0 MB) View All
Repositories	Central NeetGames Public
Ranking	#13 in MvnRepository (See Top Artifacts) #1 in Code Generators
Used By	24,221 artifacts

Maven
Gradle
Gradle (Short)
Gradle (Kotlin)
SBT
Ivy
Grape
Leiningen
Buildr

```
compileOnly 'org.projectlombok:lombok:1.18.32'
```

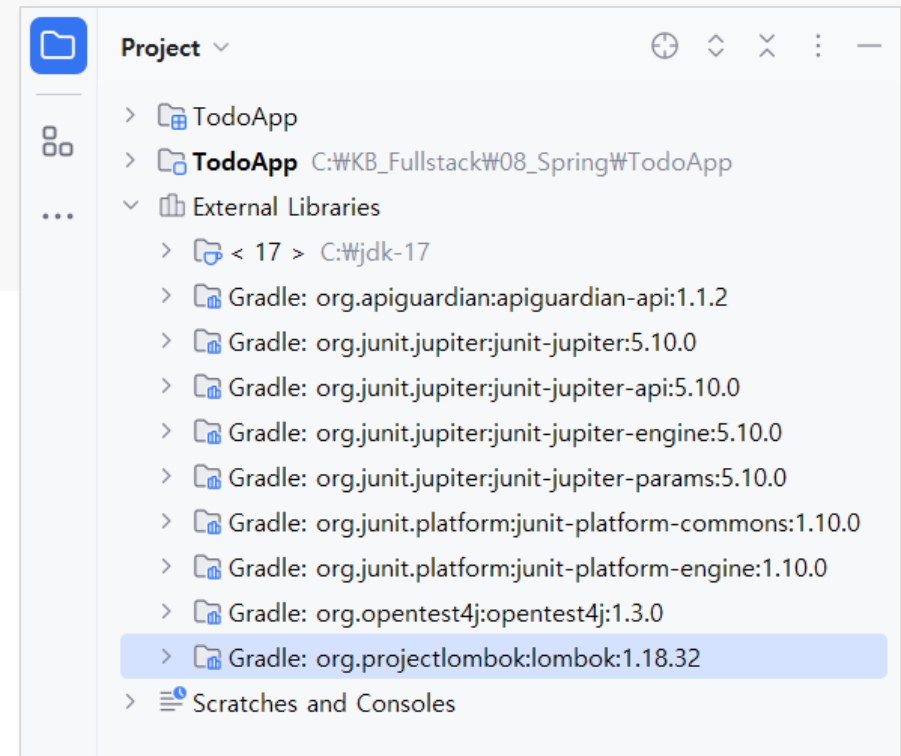
☐ Include comment with link to declaration

1 TodoApp

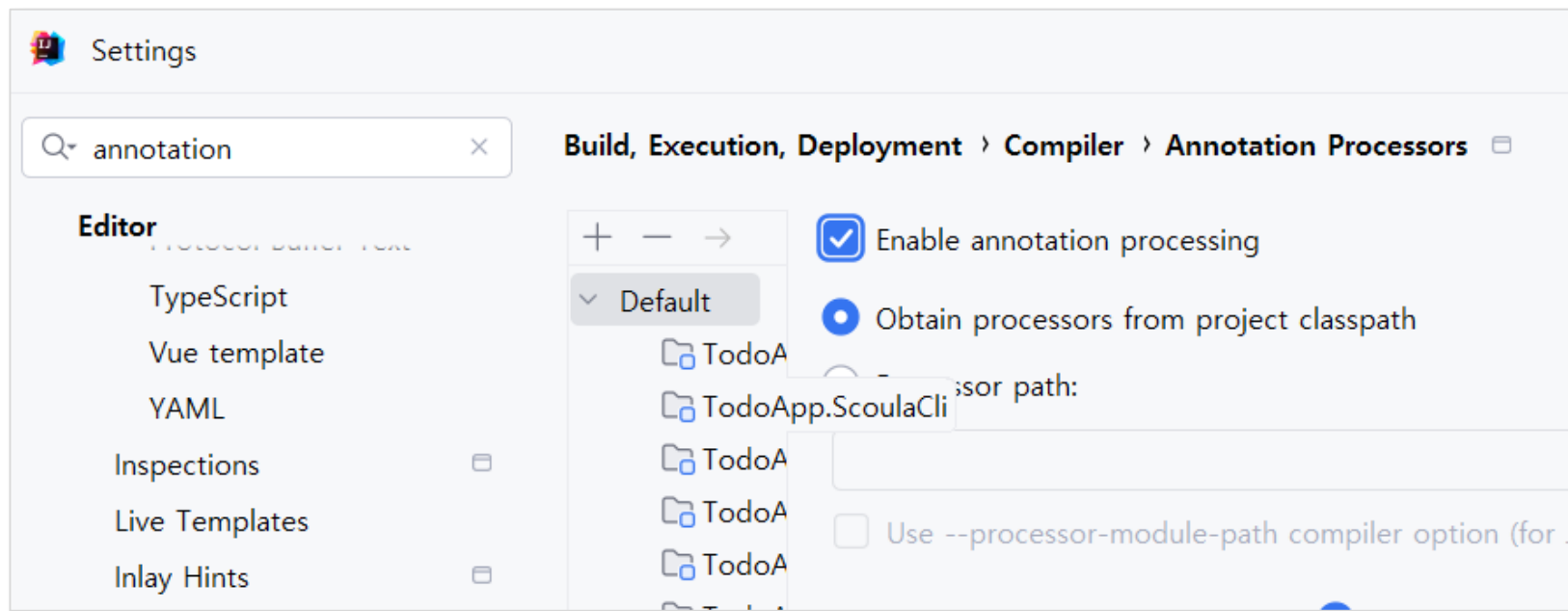
build.gradle

```
...
dependencies {
    implementation project(':ScoulaCli')
    compileOnly 'org.projectlombok:lombok:1.18.32'
    annotationProcessor 'org.projectlombok:lombok:1.18.32'

    testImplementation platform('org.junit:junit-bom:5.10.0')
    testImplementation 'org.junit.jupiter:junit-jupiter'
}
...
```



✓ Annotation Processing 활성화



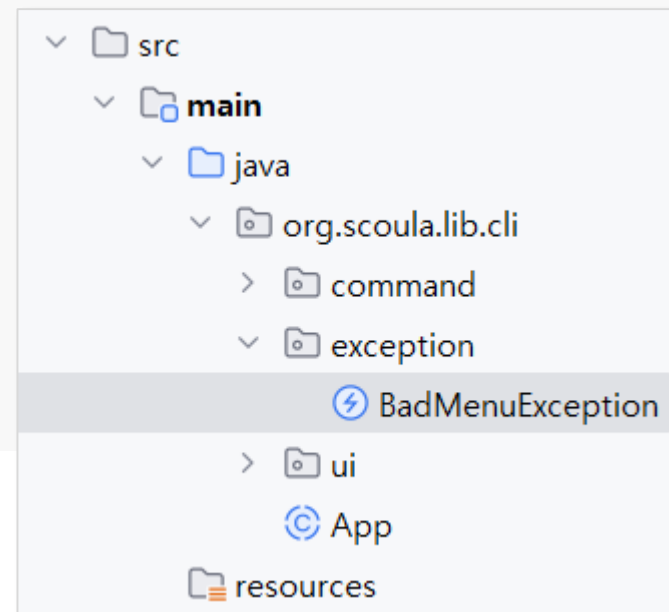
✓ 기본 라이브러리에 예외 추가

- 메뉴 선택 시 `NumberFormatException`, `ArrayIndexOutOfBoundsException` 예외 발생 가능
- 사용자 정의 예외 클래스 `BadMenuException` 운영

BadMenuException.java

```
package org.scoula.lib.cli.exception;

public class BadMenuException extends Exception{
    public BadMenuException() {
        super("잘못된 메뉴 선택입니다.");
    }
    public BadMenuException(String message) {
        super(message);
    }
}
```



App.java

```
public abstract class App {  
    ...  
  
    public void run() {  
        init();  
  
        while(true) {  
            try {  
                menu.printMenu();  
                Command command = menu.getSelect();  
                command.execute();  
            } catch (Exception e) {  
                e.printStackTrace();  
                // System.out.println("에러: " + e.getMessage());  
            }  
        }  
    }  
}
```

✓ TodoApp의 메뉴

1.목록 | 2.상세 | 3.추가 | 4.수정 | 5.삭제 | 6.종료 |

App.java

```
package org.scoula.todo;

import org.scoula.lib.cli.App;
import org.scoula.lib.cli.ui.Menu;

public class TodoApp extends App {
    @Override
    public void createMenu(Menu menu) {
        super.createMenu(menu);

        menu.add(new MenuItem("목록", null));
        menu.add(new MenuItem("상세", null));
        menu.add(new MenuItem("추가", null));
        menu.add(new MenuItem("수정", null));
        menu.add(new MenuItem("삭제", null));
    }

    public static void main(String[] args) {
        App app = new TodoApp();
        app.run();
    }
}
```

Todo.java

```
package org.scoula.todo.domain;

...
import java.util.Date;

@NoArgsConstructor
@AllArgsConstructor
@Data
public class Todo implements Cloneable { // 복제(clone)을 허용하는 인터페이스
    private static int gid = 1; // Todo Id 발급을 위한 스택틱 변수

    private int id;
    private String title;
    private String description;
    private boolean done;
    private Date date; // импорт 주의! java.util.Date임. 실수로 java.sql.Date를 импорт 하는 경우 있음

    @Builder
    public Todo(String title, String description, boolean done) {
        this(gid++, title, description, done, new Date());
    }
}
```

Todo.java

```
@Override
public Object clone() {    // 부모에서는 protected 였던 것을 public으로 변경
    try {
        return super.clone(); // 얇은 복사
    } catch (CloneNotSupportedException e) {
        throw new RuntimeException(e);
    }
}

public String getRegDate() {
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    return sdf.format(date);
}
}
```

✓ Dao 클래스

- Data Access Object
- 데이터에 대한 CRUD 처리 담당
 - 데이터 소스(배열, List, 데이터베이스 등)에 대한 처리를 상위 계층(비즈니스 계층)으로 부터 분리

TodoDao.java

```
package org.scoula.todo.dao;

...
public class TodoDao {
    private static TodoDao instance = new TodoDao();

    public static TodoDao getInstance() {
        return instance;
    }

    private List<Todo> list;

    private TodoDao() {
        list = new ArrayList<>();
        for(int i=0; i<10; i++){ // 임시 테스트 데이터 구성
            Todo todo = Todo.builder()
                .title("Todo " + i)
                .description("Description " + i)
                .done(false)
                .build();
            list.add(todo);
        }
    }
}
```

TodoDao.java

```
public List<Todo> getList() {  
    return list;  
}  
  
public Todo getTodo(int id) {  
    for(Todo todo : list){  
        if(todo.getId() == id){  
            return todo;  
        }  
    }  
    return null;  
}  
  
public void add(Todo todo) {  
    list.add(todo);  
}
```


TodoDao.java

```
public void update(Todo todo) {
    for(int i=0; i<list.size(); i++){
        if(todo.getId() == list.get(i).getId()){
            list.set(i, todo);
        }
    }
}

public void delete(int id) {
    for(int i=0; i<list.size(); i++){
        if(list.get(i).getId() == id){
            list.remove(i);
            return;
        }
    }
}
```

✓ 목록 보기

○ PrintTodoCommand

1.목록 |2.상세 |3.추가 |4.수정 |5.삭제 |6.종료 |

선택> 1

- 1] Todo 0
- 2] Todo 1
- 3] Todo 2
- 4] Todo 3
- 5] Todo 4
- 6] Todo 5
- 7] Todo 6
- 8] Todo 7
- 9] Todo 8
- 10] Todo 9

PrintTodoCommand.java

```
package org.scoula.todo.command;

import org.scoula.lib.cli.command.Command;
import org.scoula.todo.dao.TODODao;
import org.scoula.todo.domain.TODO;

public class PrintTodoCommand implements Command {
    TODODao dao = TODODao.getInstance();

    @Override
    public void execute() {
        for(TODO todo: dao.getList()) {
            String line = "%2d] %s".formatted(todo.getId(), todo.getTitle());
            System.out.println(line);
        }
        System.out.println();
    }
}
```

TodoApp.java

```
public class TodoApp extends App {
    @Override
    public void createMenu(Menu menu) {
        super.createMenu(menu);

        menu.add(new MenuItem("목록", new PrintTodoCommand()));
        menu.add(new MenuItem("상세", null));
        menu.add(new MenuItem("추가", null));
        menu.add(new MenuItem("수정", null));
        menu.add(new MenuItem("삭제", null));
    }

    public static void main(String[] args) {
        App app = new TodoApp();
        app.run();
    }
}
```

☑ 상세 보기

○ DetailTodoCommand

1.목록 |2.상세 |3.추가 |4.수정 |5.삭제 |6.종료 |

선택> 2

Todo Id: 4

[Todo 상세보기]-----

ID : 4

제목 : Todo 3

설명 : Description 3

완료여부: false

등록일 : 2024-06-27 15:27:41

DetailTodoCommand.java

```
package org.scoula.todo.command;

...

public class DetailTodoCommand implements Command {
    TodoDao dao = TodoDao.getInstance();

    @Override
    public void execute() {
        int id = Input.getInt("Todo Id: ");
        Todo todo = dao.getTodo(id);

        System.out.println("[Todo 상세보기]-----");
        System.out.println("ID      : " + todo.getId());
        System.out.println("제목    : " + todo.getTitle());
        System.out.println("설명     : " + todo.getDescription());
        System.out.println("완료여부: " + todo.isDone());
        System.out.println("등록일   : " + todo.getRegDate());
        System.out.println("-----");
        System.out.println();
    }
}
```

TodoApp.java

```
public class TodoApp extends App {
    @Override
    public void createMenu(Menu menu) {
        super.createMenu(menu);

        menu.add(new MenuItem("목록", new PrintTodoCommand()));
        menu.add(new MenuItem("상세", new DetailTodoCommand()));
        menu.add(new MenuItem("추가", null));
        menu.add(new MenuItem("수정", null));
        menu.add(new MenuItem("삭제", null));
    }

    public static void main(String[] args) {
        App app = new TodoApp();
        app.run();
    }
}
```

✓ 삭제

○ DeleteTodoCommand

1.목록 |2.상세 |3.추가 |4.수정 |5.삭제 |6.종료 |

선택> 5

삭제할 Todo Id: 4

1.목록 |2.상세 |3.추가 |4.수정 |5.삭제 |6.종료 |

선택> 1

1] Todo 0

2] Todo 1

3] Todo 2

5] Todo 4

6] Todo 5

...

DeleteTodoCommand.java

```
package org.scoula.todo.command;

import org.scoula.lib.cli.command.Command;
import org.scoula.lib.cli.ui.Input;
import org.scoula.todo.dao.TODODao;
import org.scoula.todo.domain.TODO;

public class DeleteTodoCommand implements Command {
    TODODao dao = TODODao.getInstance();

    @Override
    public void execute() {
        int id = Input.getInt("삭제할 TODO Id: ");
        dao.delete(id);

        System.out.println();
    }
}
```

TodoApp.java

```
public class TodoApp extends App {
    @Override
    public void createMenu(Menu menu) {
        super.createMenu(menu);

        menu.add(new MenuItem("목록", new PrintTodoCommand()));
        menu.add(new MenuItem("상세", new DetailTodoCommand()));
        menu.add(new MenuItem("추가", null));
        menu.add(new MenuItem("수정", null));
        menu.add(new MenuItem("삭제", new DeleteTodoCommand()));
    }

    public static void main(String[] args) {
        App app = new TodoApp();
        app.run();
    }
}
```

✓ 추가

○ AddTodoCommand

 1.목록 |2.상세 |3.추가 |4.수정 |5.삭제 |6.종료 |

선택> 3

[새 Todo 추가]-----

제목: 프레임워크 공부

설명: 내가 만든 Cli 프레임워크 라이브러리를 익힌다.

 1.목록 |2.상세 |3.추가 |4.수정 |5.삭제 |6.종료 |

선택> 1

1] Todo 0

...

10] Todo 9

11] 프레임워크 공부

Input.java

```
package org.scoula.lib.cli.ui;

import java.util.Scanner;

public class Input {
    static Scanner scanner = new Scanner(System.in);

    public static int getInt(String title) {
        System.out.print(title);
        return Integer.parseInt(scanner.nextLine());
    }

    public static String getLine(String title) {
        System.out.print(title);
        return scanner.nextLine();
    }
}
```

AddTodoCommand.java

```
package org.scoula.todo.command;

...
public class AddTodoCommand implements Command {
    TodoDao dao = TodoDao.getInstance();

    @Override
    public void execute() {
        System.out.println("[새 Todo 추가]-----");
        String title = Input.getLine("제목: ");
        String description = Input.getLine("설명: ");
        System.out.println("-----");

        Todo todo = Todo.builder()
            .title(title)
            .description(description)
            .done(false)
            .build();
        dao.add(todo);

        System.out.println();
    }
}
```

TodoApp.java

```
public class TodoApp extends App {
    @Override
    public void createMenu(Menu menu) {
        super.createMenu(menu);

        menu.add(new MenuItem("목록", new PrintTodoCommand()));
        menu.add(new MenuItem("상세", new DetailTodoCommand()));
        menu.add(new MenuItem("추가", new AddTodoCommand()));
        menu.add(new MenuItem("수정", null));
        menu.add(new MenuItem("삭제", new DeleteTodoCommand()));
    }

    public static void main(String[] args) {
        App app = new TodoApp();
        app.run();
    }
}
```

✓ 수정

○ UpdateTodoCommand

1.목록 |2.상세 |3.추가 |4.수정 |5.삭제 |6.종료 |

선택> 4

수정할 Id: 3

[Todo 수정하기]-----

ID : 3

제목(Todo 2): 제목 수정

설명(Description 2): 설명 수정

완료여부 (y/N):

변경하지 않는 경우 그냥 엔터

1.목록 |2.상세 |3.추가 |4.수정 |5.삭제 |6.종료 |

선택> 2

Todo Id: 3

[Todo 상세보기]-----

ID : 3

제목 : 제목 수정

설명 : 설명 수정

완료여부: false

등록일 : 2024-06-27 15:39:36

Input.java

```
public class Input {
    ...

    public static String getLine(String title, String defaultValue) {
        // 이름(김길동):
        System.out.printf("%s(%s): ", title, defaultValue);
        String answer = scanner.nextLine();

        // 그냥 엔터 쳤으면 defaultValue 리턴, 입력값이 있으면 answer 리턴
        return answer.isEmpty() ? defaultValue : answer;
    }
}
```


Input.java

```
public static boolean confirm(String title) {
    return confirm(title, true);
}

public static boolean confirm(String title, boolean defaultValue) {
    String yesNo = defaultValue ? "(Y/n)" : "(y/N)";
    System.out.printf("%s %s: ", title, yesNo);

    String answer = scanner.nextLine();
    if (answer.isEmpty())
        return defaultValue;

    return answer.equalsIgnoreCase("y");
}
```

UpdateTodoCommand.java

```
package org.scoula.todo.command;

...

public class UpdateTodoCommand implements Command {
    TodoDao dao = TodoDao.getInstance();

    @Override
    public void execute() {
        int id = Input.getInt("수정할 Id: ");
        Todo todo = dao.getTodo(id);

        System.out.println("[Todo 수정하기]-----");
        System.out.println("ID      : " + todo.getId());
        String title = Input.getLine("제목", todo.getTitle());
        String description = Input.getLine("설명", todo.getDescription());
        Boolean done = Input.confirm("완료여부", todo.isDone());
        System.out.println("-----");
        System.out.println();
    }
}
```

UpdateTodoCommand.java

```
    Todo updateTodo = (Todo)todo.clone();
    updateTodo.setTitle(title);
    updateTodo.setDescription(description);
    updateTodo.setDone(done);

    dao.update(updateTodo);
}
```

TodoApp.java

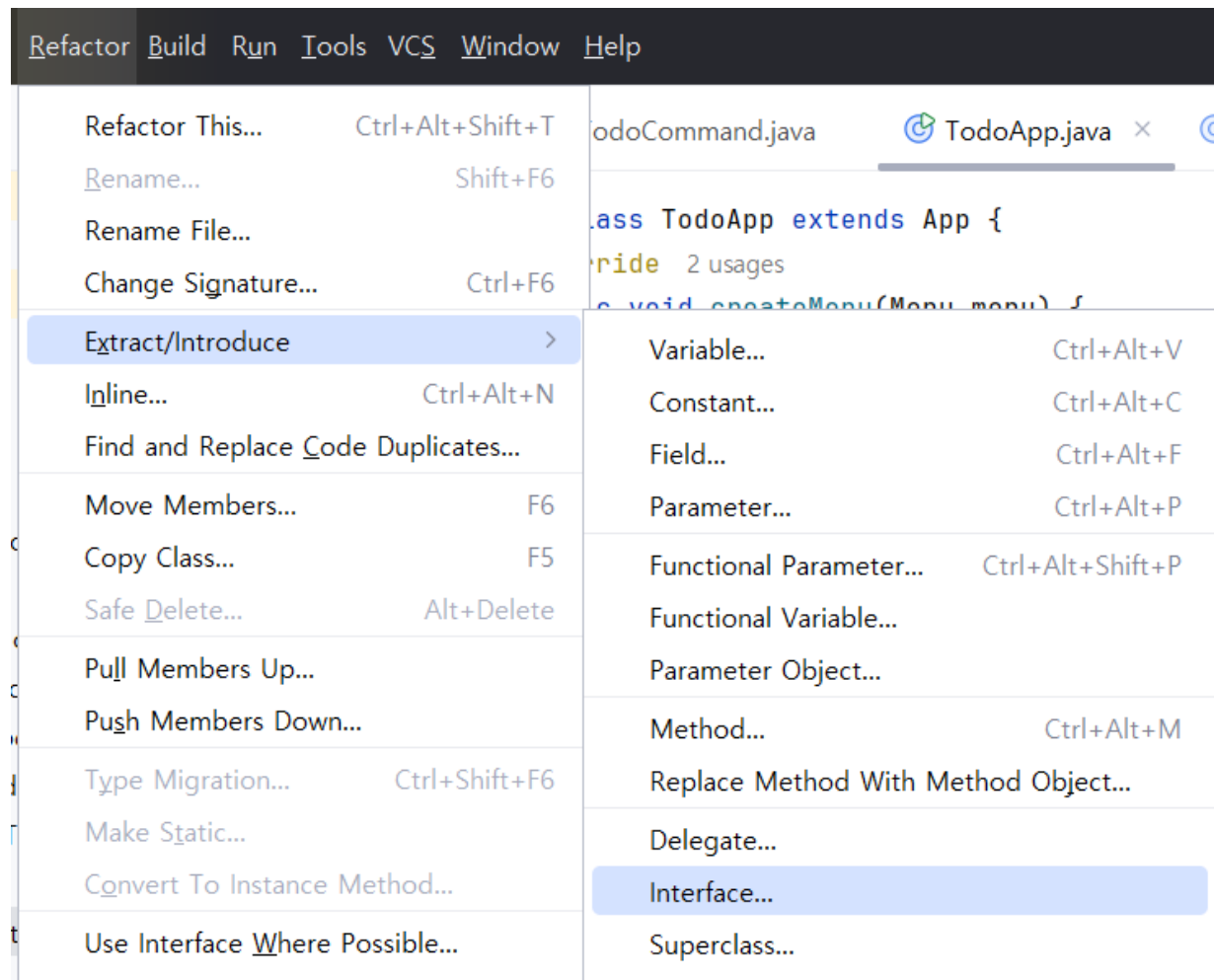
```
public class TodoApp extends App {
    @Override
    public void createMenu(Menu menu) {
        super.createMenu(menu);

        menu.add(new MenuItem("목록", new PrintTodoCommand()));
        menu.add(new MenuItem("상세", new DetailTodoCommand()));
        menu.add(new MenuItem("추가", new AddTodoCommand()));
        menu.add(new MenuItem("수정", new UpdateTodoCommand()));
        menu.add(new MenuItem("삭제", new DeleteTodoCommand()));
    }

    public static void main(String[] args) {
        App app = new TodoApp();
        app.run();
    }
}
```

✓ TodoDao 추상화

- TodoDao 이름 변경: TodoListDao
- Interface 추출하기



✓ ToDoDao 추상화

- interface name: **ToDoDao**
- 인터페이스에 추가할 메서드 선택

Extract Interface

Extract interface from:

org.scoula.todo.dao.ToDoListDao

☒ Extract interface
 ☐ Rename original class and use interface where possible

Interface name:

ToDoDao

Package for new interface:

org.scoula.todo.dao

Target destination directory:

Leave in same source root

Members To Form Interface

	Member	Make Abstract
<input type="checkbox"/>	getInstance():ToDoListDao	<input type="checkbox"/>
<input checked="" type="checkbox"/>	getList():List<Todo>	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	getTodo(id:int):Todo	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	add(todo:Todo):void	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	update(todo:Todo):void	<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/>	delete(id:int):void	<input checked="" type="checkbox"/>

JavaDoc

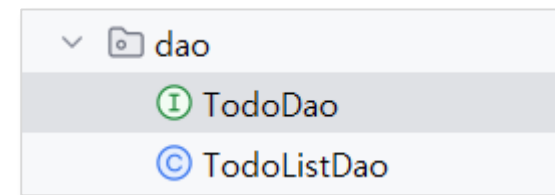
☒ As is
 ☐ Copy
 ☐ Move

?

Refactor

Preview

Cancel



TodoDao.java

```
package org.scoula.todo.dao;

import org.scoula.todo.domain.TODO;

import java.util.List;

public interface TodoDao {
    List<TODO> getList();

    TODO getTODO(int id);

    void add(TODO todo);

    void update(TODO todo);

    void delete(int id);
}
```

TodoListDao.java

```
public class TodoListDao implements TodoDao {  
    private static TodoListDao instance = new TodoListDao();  
  
    public static TodoDao getInstance() {  
        return instance;  
    }  
  
    ...  
}
```


AddTodoCommand.java

```
public class AddTodoCommand implements Command {  
    TodoDao dao = TodoListDao.getInstance();  
    ...  
}
```

- 다른 커맨드로 TodoListDao를 TodoDao 인터페이스 타입으로 변경