Encapsulation

Lab 4

TA : Changmin Jeon, Minji Kim, Hyunwoo Jung,
Hyunseok Oh, Jingyu Lee, Seungwoo Jo

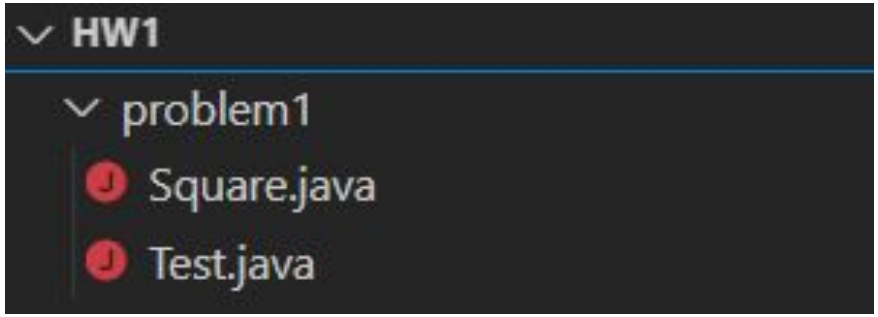
SEOUL NATIONAL UNIVERSITY

Assignment Grading Process

- Let's take a look with an example.
- Implement a program that outputs the square of the number entered into the input.

Directory Structure

- HW1
  - problem1
    - Test.java
    - Square.java



Assignment Grading Process

1. **Unzip** HW1.zip
2. **Swap** Test.java for Testing.
3. **Compile** Test.java
4. **Execute** Test.java with Various Testcases
5. **Compare** outputs

Overview

- Import built-in packages
- Java API Documents
- Problem 1. Implement Dice class
- Problem 2. RockPaperScissors class
- Problem 3. Implement Platform class

Import Built-In Packages

- Package encapsulates a group of classes
- There are various useful built-in Packages.
  - `import java.util.Scanner;`
  - `import java.time.*;`

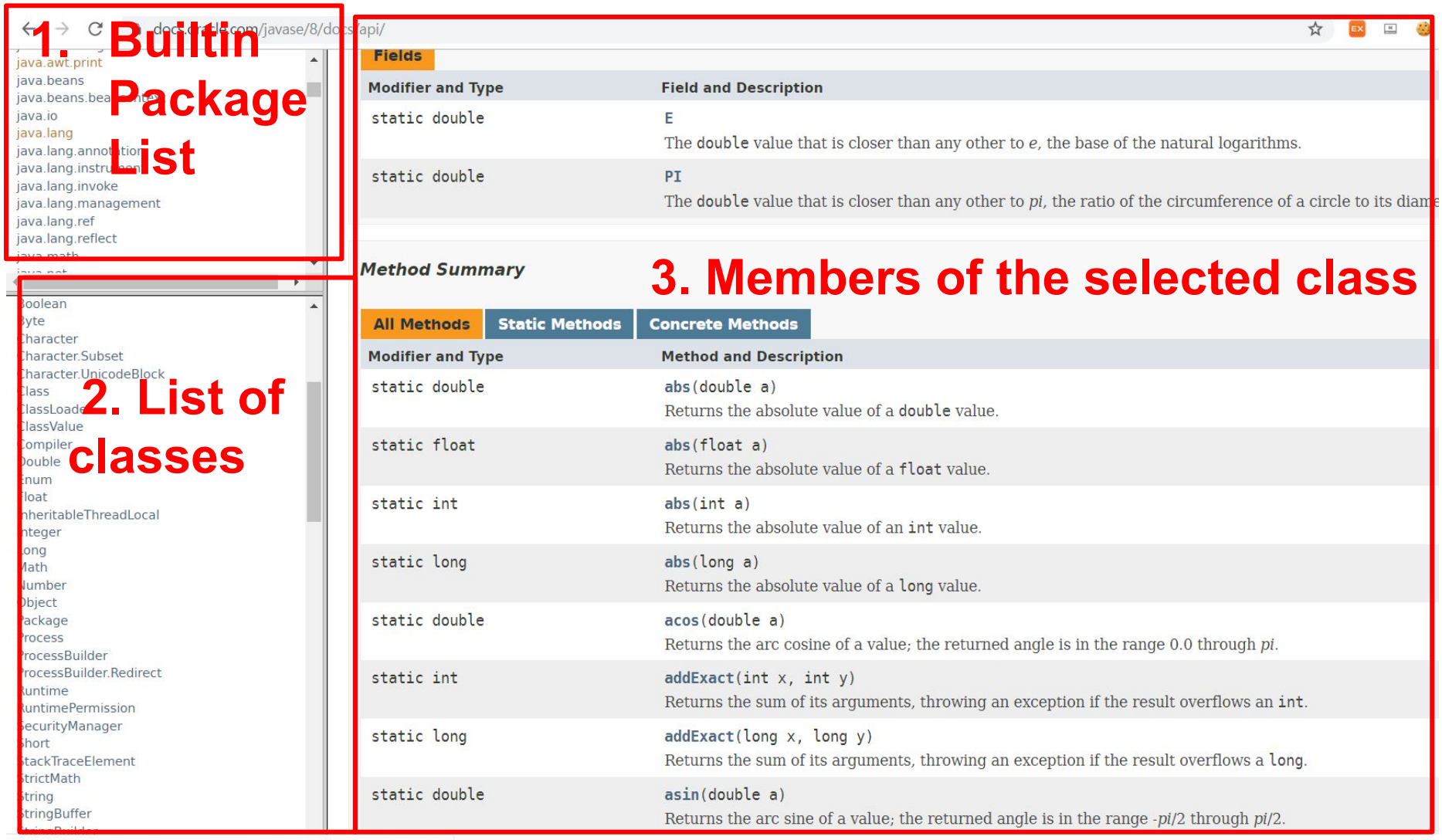
Java API Documents

- The detailed documentation of of Java APIs .
  https://docs.oracle.com/javase/8/docs/api/
- Import useful packages with rich functionality.

Why are they necessary?

- Because it is cumbersome to manually implement all the functionalities we need.
- So we borrow the existing ones.

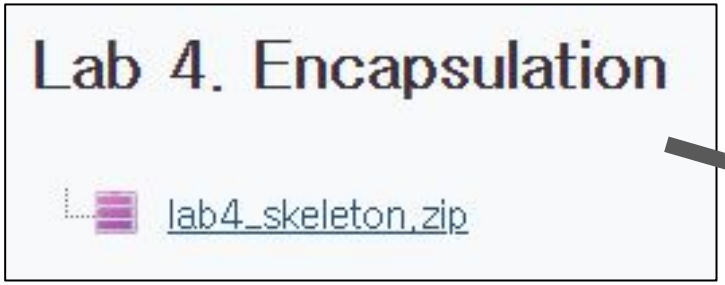
Java API Documents



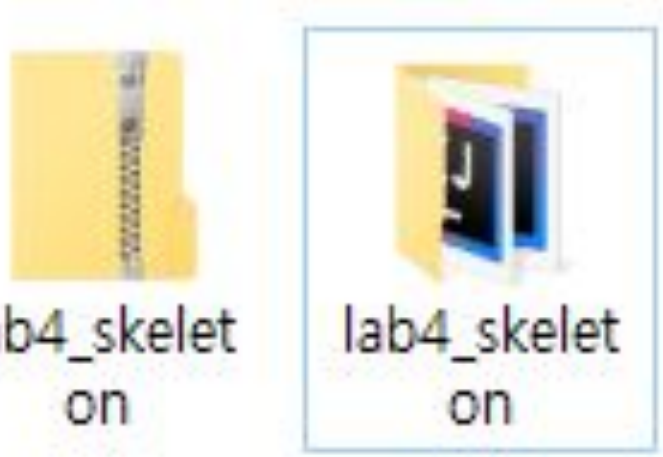
Goal of the Lab

- Understand the concept package.
- Understand the necessity of APIs.

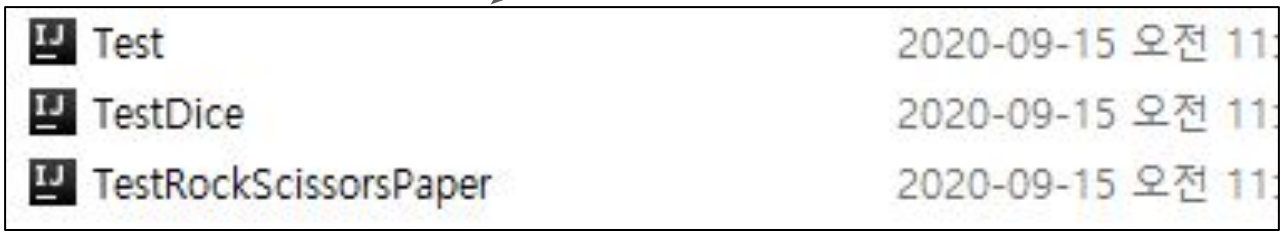
Before going in to the problem..



Download the skeleton

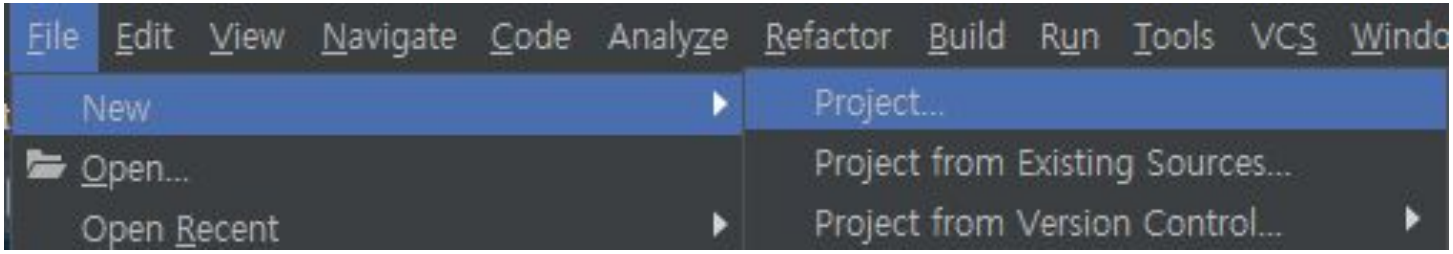


Extract the Zip file

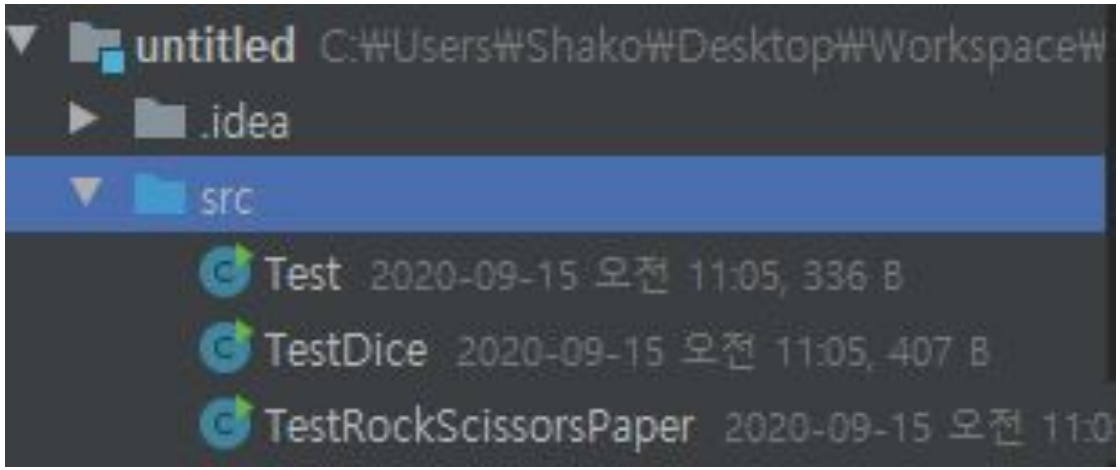


Check the .java files

Before going in to the problem..

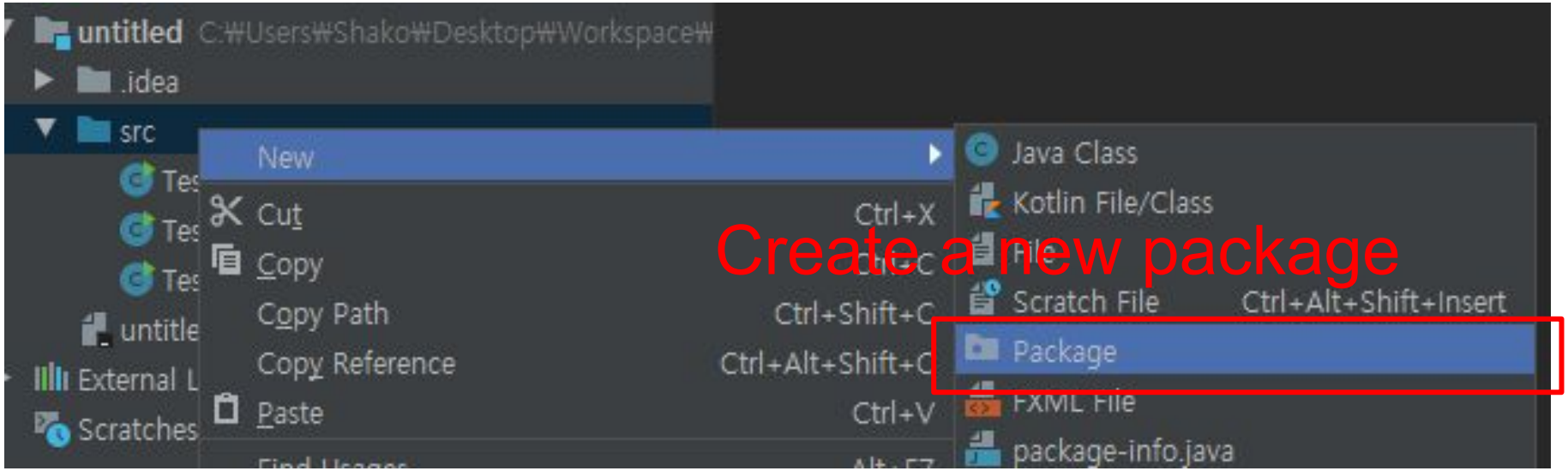


Create a new project

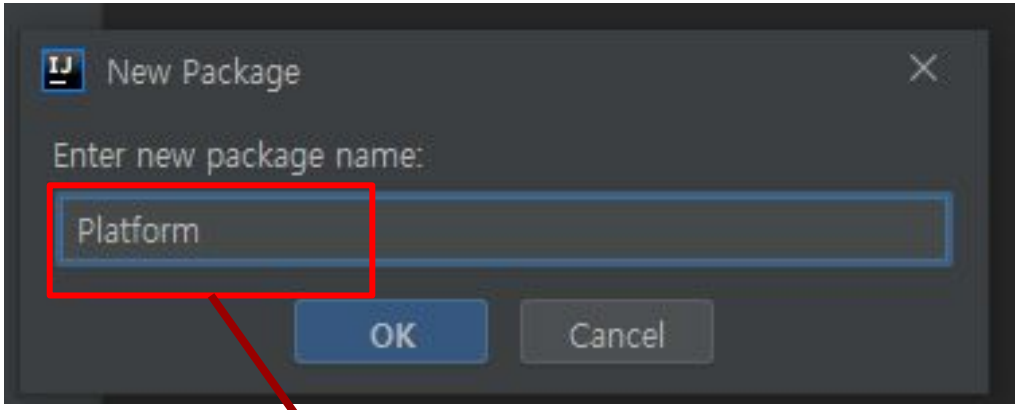


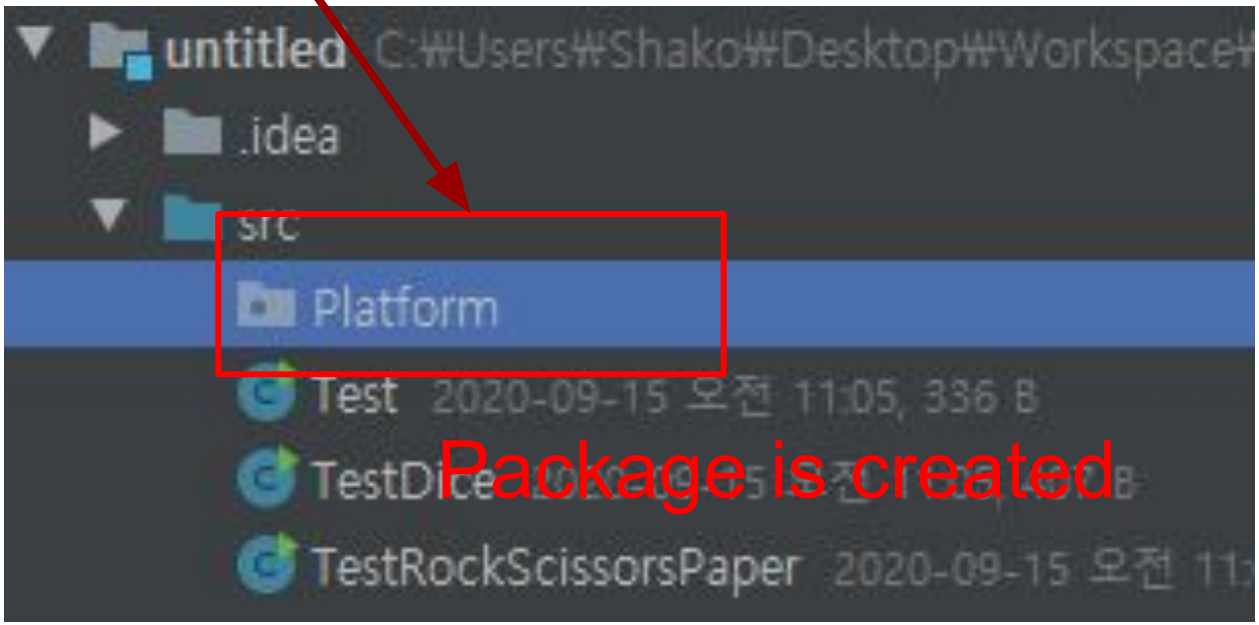
Paste the skeleton code classes to the src dir.

Create Packages

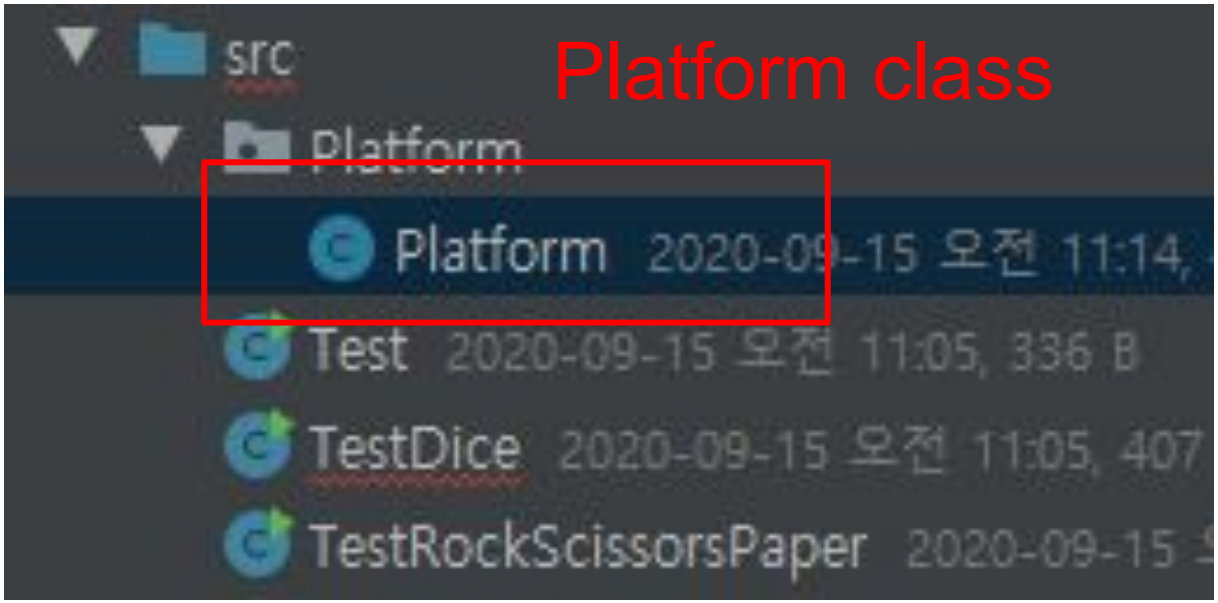


Create Packages

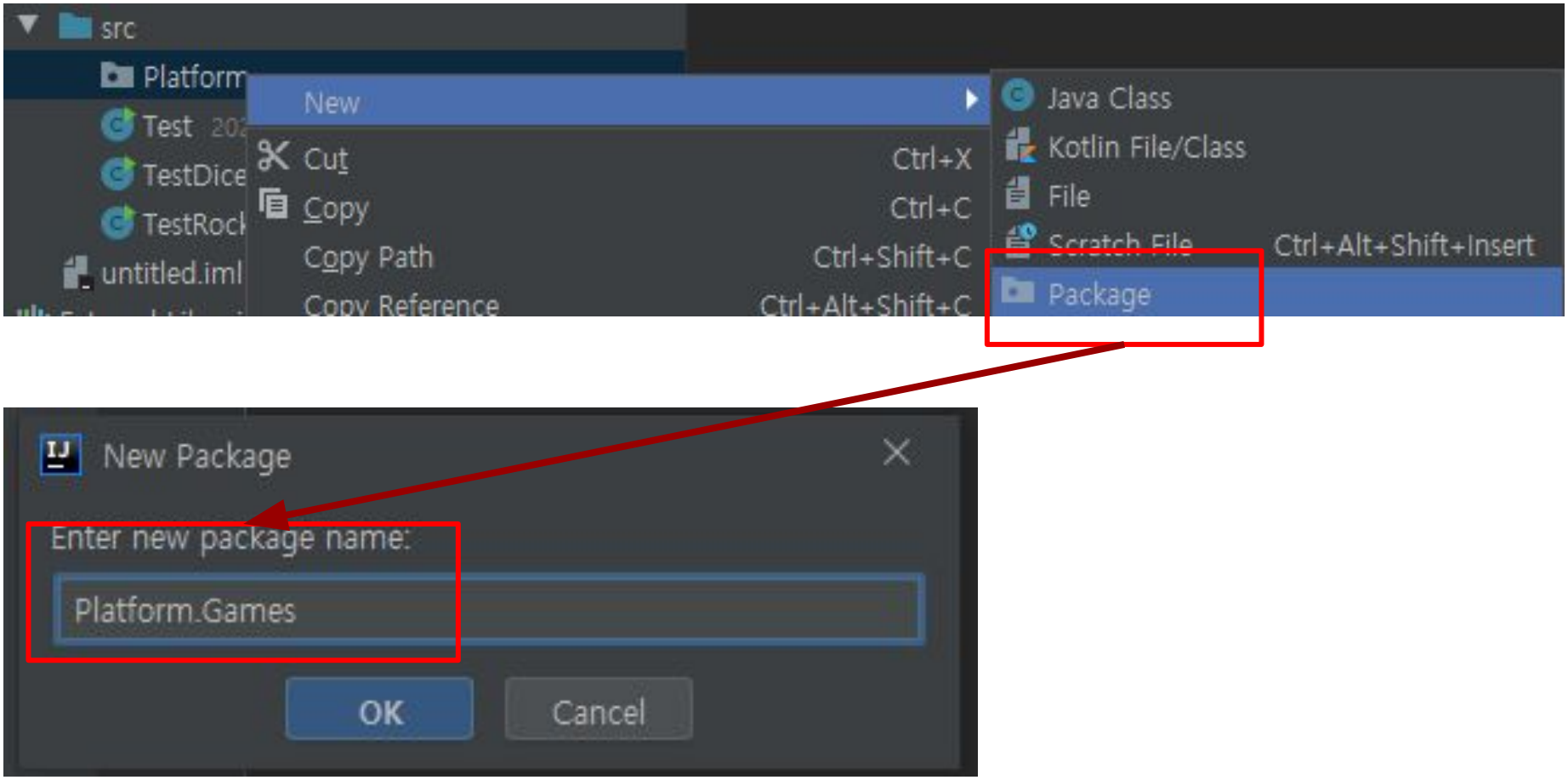




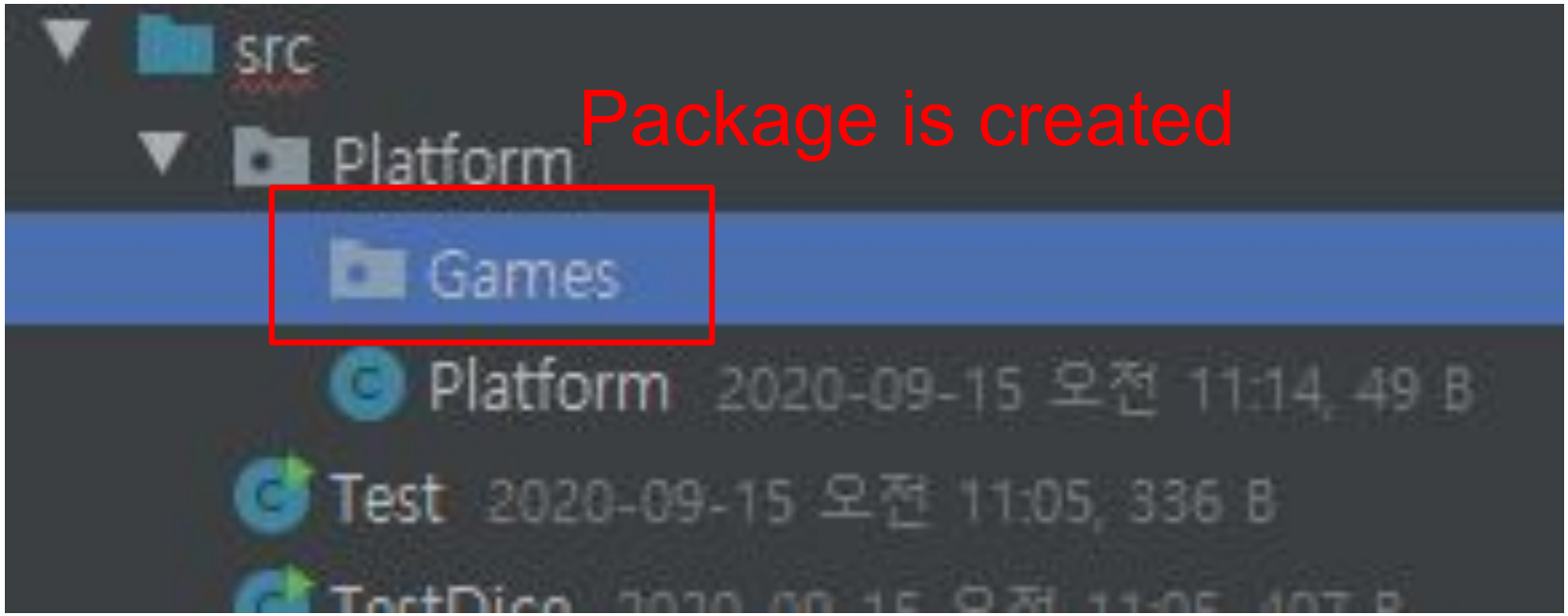
Create Class under the New Package



Create More Packages

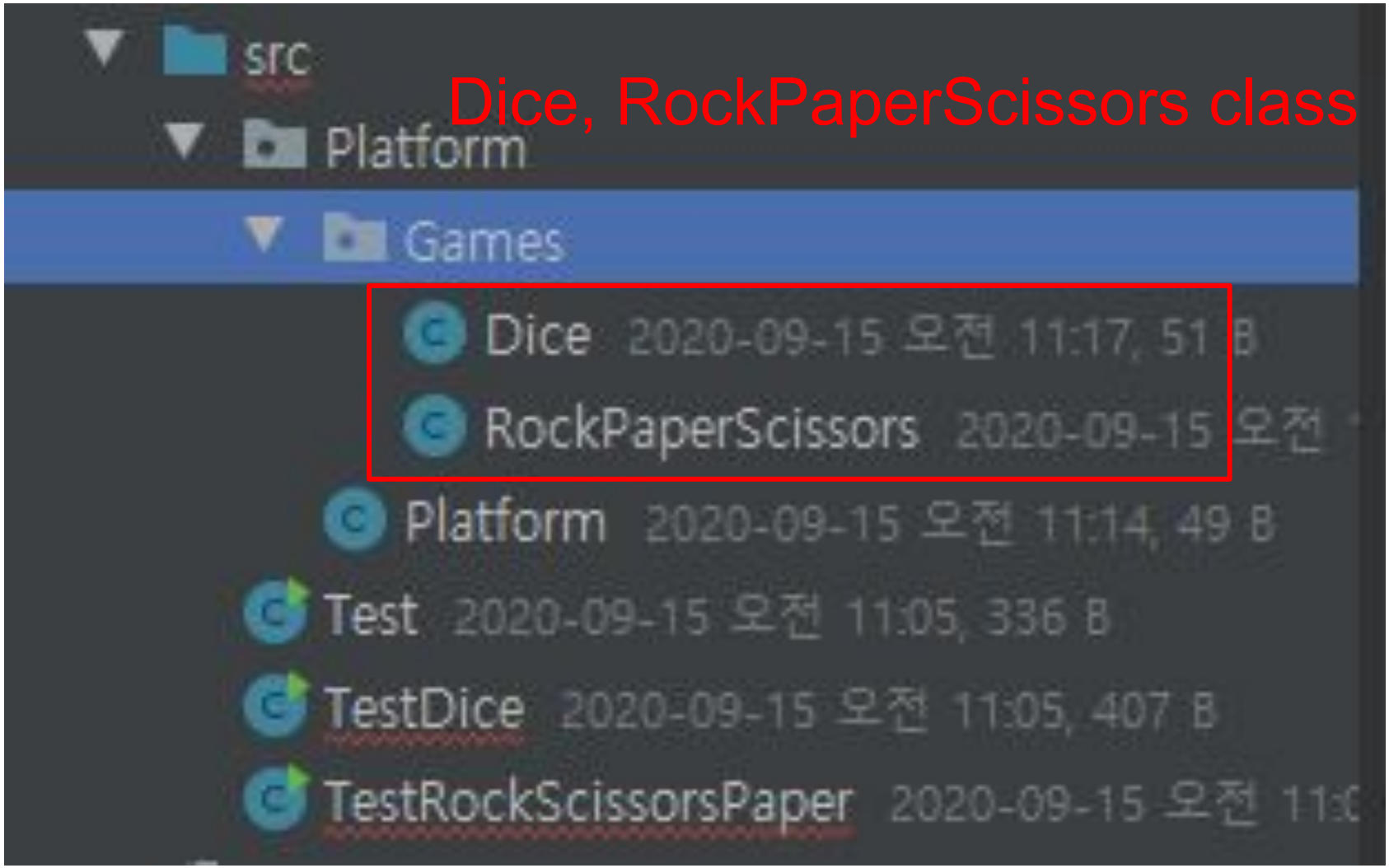


Create More Packages



Package is created

Create Class under the Package



Dice, RockPaperScissors class

Let’s Build the Layout!

```
public class Dice {
    public int playGame() { return -1; }
}
```

```
public class RockPaperScissors {
    public int playGame() { return -1; }
}
```

```
 public class Platform {
    public float run(){ return -0.0f; }
    public void setRounds(int num) { }
 }
```

Goal of the Problem 1

- Understand the concept of packages.
- Use already implemented functionality (API)
  - Math.random, Scanner

Problem 1. Implement Dice Class

- Objective: Implement `public int playGame` method of `Dice`.
- Description of the method
  - The dice randomly outputs int from 0 to 99.
    - Use `Math.random()` function wisely.
  - The user and the opponent rolls the dice one time.
    - If user's dice int is large than the opponent(win), return 1.
    - If user's dice int is smaller than the opponent(lose), return -1.
    - else (draw), return 0.
  - Before returning, `println` the user's int and the opponent's int with **a single space** in between them.
    - e.g.

```
47 11
```

```
40 42
```

  - You can test it with the `TestDice` class in the skeleton.

Dice Class Skeleton

```
package Platform.Games;
public class Dice {
    public int playGame() {
        // TODO Lab 4. Problem 1.
    }
}
```

Goal of the Problem 2

- Implement the class with similar interface.
  - Concept of abstraction needed!

Problem 2. RockPaperScissors Class

- Objective: Implement `public int playGame` method of `RockPaperScissors` class.
- Description of the method:
  - User console inputs one of words “scissor”, “rock”, “paper”
    - Beware of the case. If other word inputs, the User loses. (return -1)
  - opponent randomly choose one of “scissor”, “rock”, “paper”.
    - Use `Math.random()` function wisely.
  - User & Opponent does the Rock Paper Scissors game.
    - If User wins, return 1. If draw, return 0. Else, return -1.
  - Before returning, print**ln** the user’s choice and the opponent’s choice with **a single space** in between them.
    - e.g.

```
scissor paper
```

```
rock paper
```

  - Test it with the `TestRockScissorsPaper` class in the skeleton.

RockPaperScissors Class Skeleton

```
package Platform.Games;

public class RockPaperScissors {
    public int playGame() {
        // TODO Lab 4. Problem 2.
    }
}
```

Goal of the Problem 3

- Understand Inter-package import
- Understand the necessity of getter/setter
- Understand the necessity of the generalization of the class. (Will be presented in Inheritance)

Problem 3. Implement Platform Class

- Objective: Implement `public float run` and `public void setRounds` method of `Platform` class.
- Description of the method:
  - `setRounds` sets a number of game rounds per `run()` call.
    - After the first call of this, the consequent setRounds **should not** be able to set the game rounds.
    - The initial number of rounds should be 1.
  - `run()` first console inputs an integer 0 or 1.
    - if 0, play `Dice.playGame` for the number of rounds `setRounds` has set, and return the win rate in **float**.
    - if 1, play `RockPaperScissors.playGame` for the number of rounds `setRounds` has set, and return the win rate in **float**.
    - Win rate : (number of wins) / (total number of rounds)
  - You can test it with the `Test` class in the skeleton.

Problem 3. Implement Platform Class

- Expected Console I/O of the Test class

```
Choose 0 for Dice, 1 for
RockPaperScissors
0
73 38
58 10
95 26
69 39
2 65
38 77
0.6666667
```

```
Choose 0 for Dice, 1 for
RockPaperScissors
1
scissor
scissor scissor
paper
paper rock
rock
rock rock
paper
paper scissor
scissor
scissor paper
scissor
scissor rock
0.33333334
```

```
Choose 0 for Dice, 1 for
RockPaperScissors
1
scissor
scissor paper
paper
paper rock
paeper
paper
paper scissor
rock
rock paper
rock
rock paper
0.33333334
```

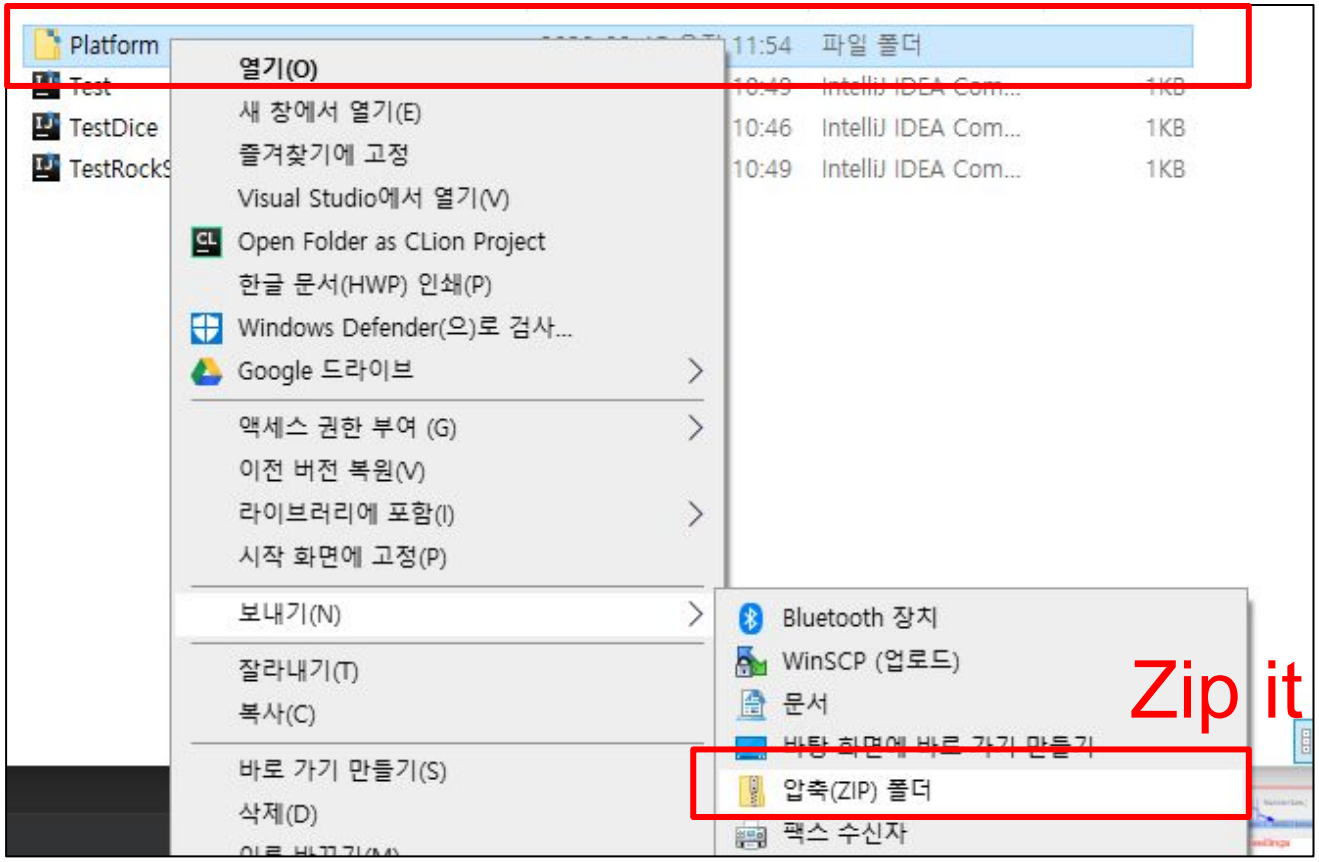
Platform Class Skeleton

```
package Platform;
public class Platform {
  public float run(){
    // TODO Lab 4. Problem 3.
  }
  public void setRounds(int num) {
    // TODO Lab 4. Problem 3.
  }
}
```

Submission

- Compress your `Platform` package directory into a zip file.
  - After unzip, the 'Platform' directory must appear.
- Rename your zip file as 20XX-XXXXX_{name}.zip - for example, 2020-12345_KimMinji.zip
- Upload it to eTL - Lab 4 assignment.
- Your program should contain `Platform.Platform` class, `Games.Dice` and `Games.RockPaperScissors` that does not prompt compile error on the skeleton code.

Submission


Platform
Test
TestDice
TestRockS
11:54 파일 폴더
10:46 IntelliJ IDEA Com... 1KB
10:49 IntelliJ IDEA Com... 1KB
열기(O)
새 창에서 열기(E)
즐겨찾기에 고정
Visual Studio에서 열기(V)
Open Folder as CLion Project
한글 문서(HWP) 인쇄(P)
Windows Defender(으)로 검사...
Google 드라이브
액세스 권한 부여 (G)
이전 버전 복원(V)
라이브러리에 포함(I)
시작 화면에 고정(P)
보내기(N)
잘라내기(T)
복사(C)
바로 가기 만들기(S)
삭제(D)
Bluetooth 장치
WinSCP (업로드)
문서
압축(ZIP) 폴더
팩스 수신자
Zip it


Platform