

# Encapsulation

Lab 03

# Overview

- Announcement
- Project Path
- Import built-in packages
- Encapsulation Example : RSP (Rock Scissors Paper) Game System
  - Packaging
  - Access Control

# Announcement - 1

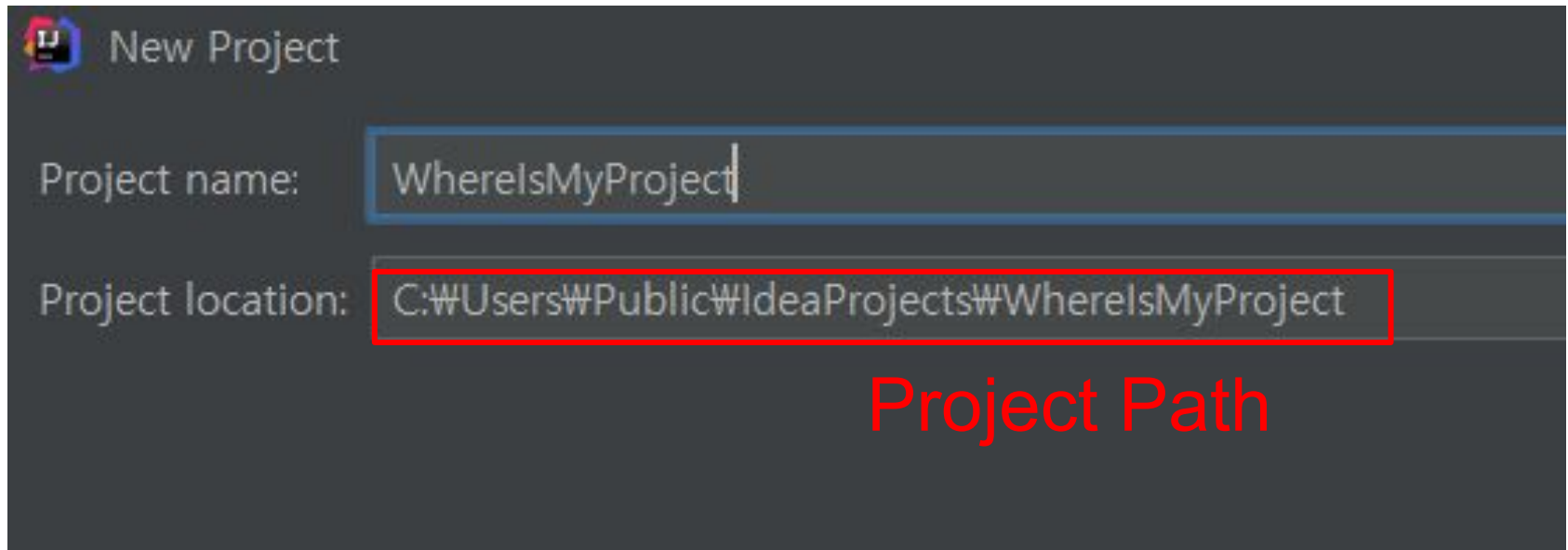
- Homework will be assigned in every 2~3 weeks, combining related programming subjects.
- FAQ will be uploaded on the ETL announcement board, and not replied individually. FAQ will be continuously updated. Please check the board before email or ask to TAs.
- Labtest 1 is re-evaluated. Please check them in the ETL, and ask TAs if you have any problems.

# Announcement - 2

- We evaluated Labtest 1 leniently this time. From now on, we will evaluate Labtests strictly without any exception. Please read Labtest specifications carefully.
- If you want to change your Lab Pair, 1) ask for consents of the desired partner, current partner, and the current partner of the desired partner, and 2) email us with all four students' names until 10/1 (Tue).

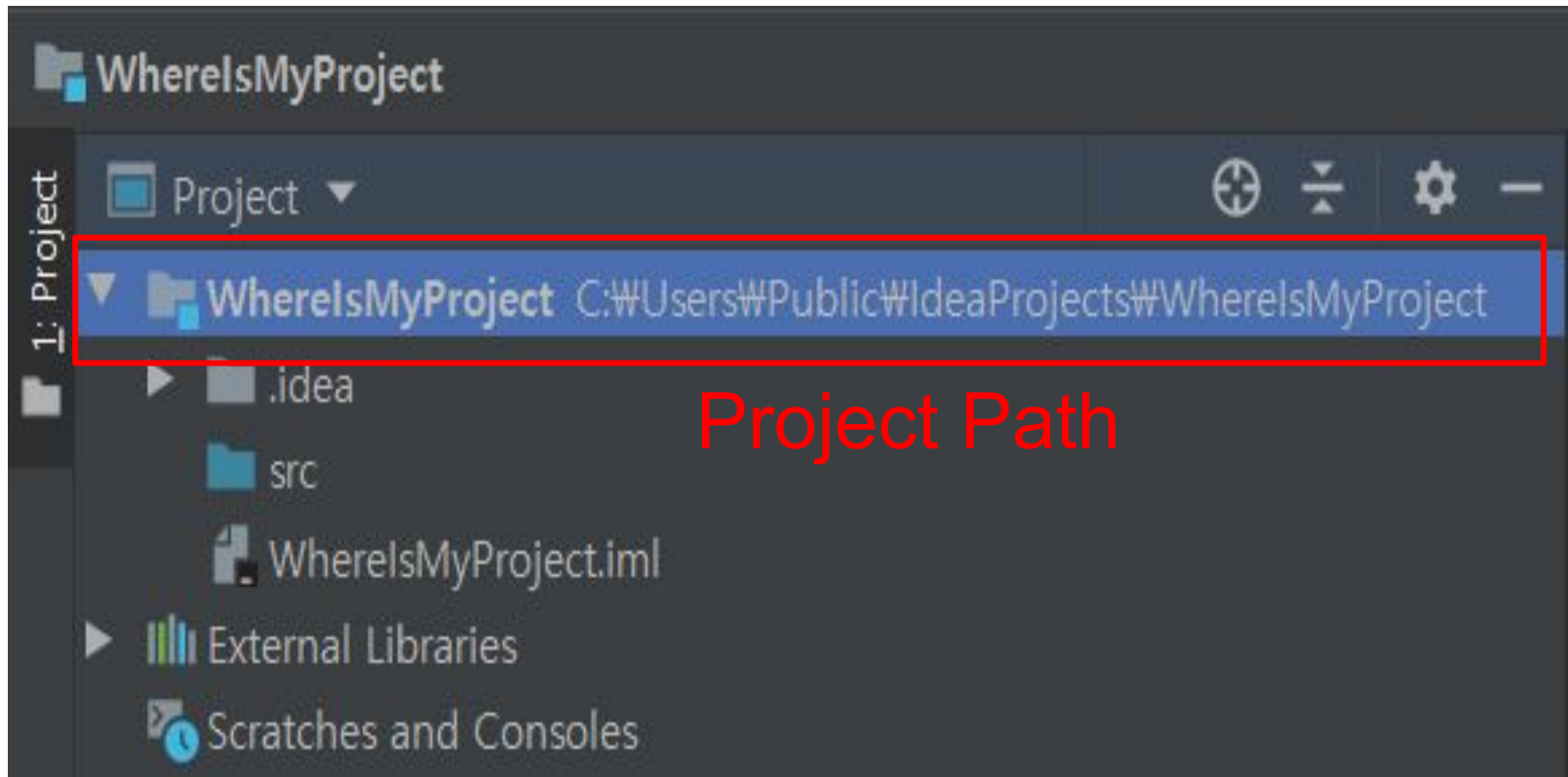
# Project Path

- When we make the project, we can set the path of the projects.

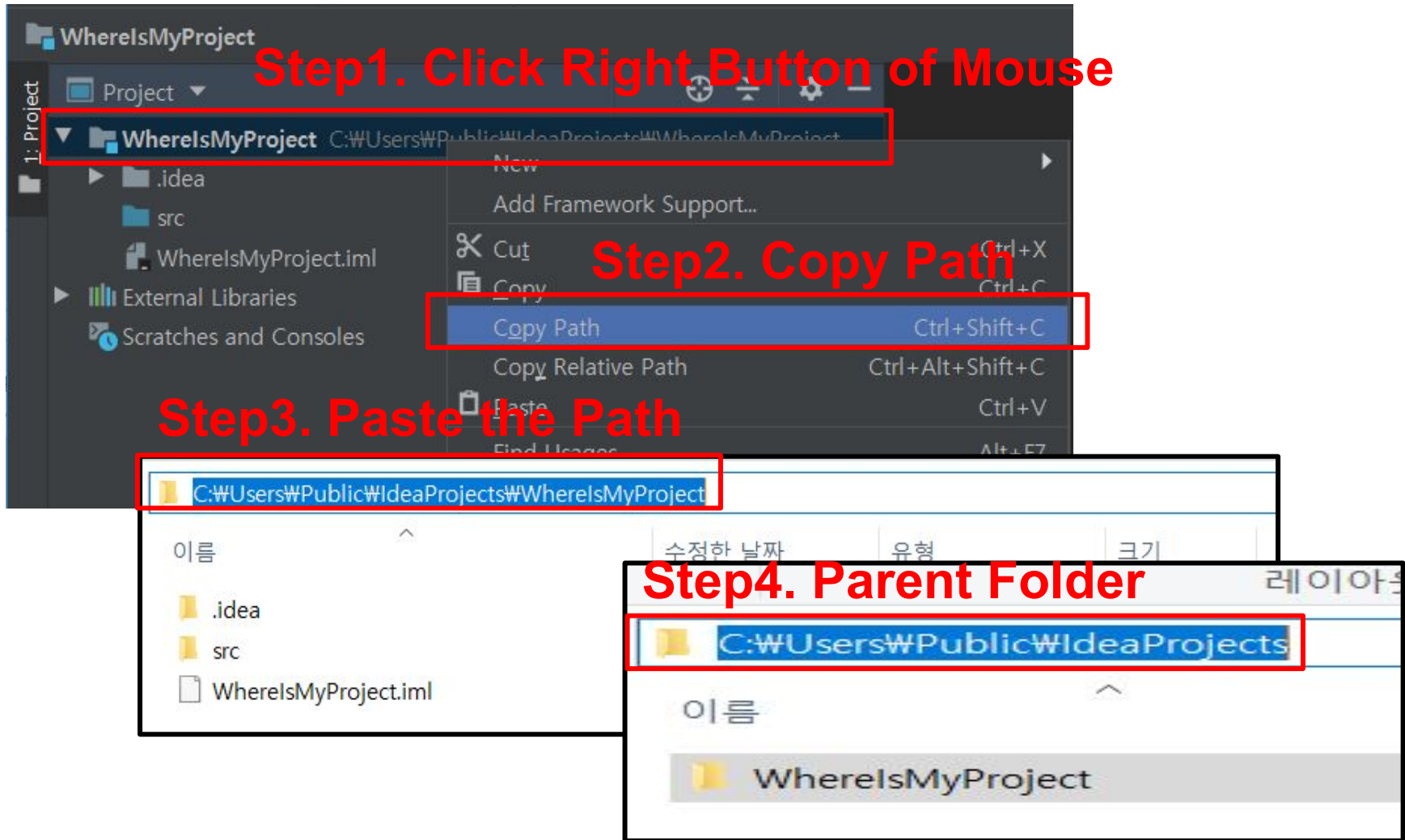


# Project Path

- We can get the project path from the Project Tree.



# Project Path



**Step1. Click Right Button of Mouse**

**Step2. Copy Path**

**Step3. Paste the Path**

**Step4. Parent Folder**

The image illustrates a four-step process to determine the parent folder of a project in IntelliJ IDEA. Step 1 shows a right-click on the 'WhereIsMyProject' folder in the Project tool window. Step 2 shows the 'Copy Path' option selected in the context menu. Step 3 shows the full path 'C:\Users\Public\IdeaProjects\WhereIsMyProject' pasted into a text field. Step 4 shows the parent folder 'C:\Users\Public\IdeaProjects' highlighted in a file explorer view.

# Import Built-In Packages

- Package in Java is a mechanism to encapsulate a group of classes and there are various and useful built-in Packages.
- `java.lang*` is automatically imported.
  - `import java.util.Scanner;` **Recall Scanner Class**
  - `import java.time.*;`



# Java Api : java.time.\*;

- The main API for dates, times, instants, and durations.
- <https://docs.oracle.com/javase/8/docs/api/java/time/>

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP	
PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES ALL CLASSES	
<b>Package java.time</b>	
The main API for dates, times, instants, and durations.	
See: Description	
<b>Class Summary</b>	
Class	Description
Clock	A clock providing access to the current instant, date and time using a time-zone.
Duration	A time-based amount of time, such as '31.5 seconds'.
Instant	An instantaneous point on the time-line.
LocalDate	A date without a time-zone in the ISO-8601 calendar system, such as 2007-12-03.
LocalDateTime	A date-time without a time-zone in the ISO-8601 calendar system, such as 2007-12-03T10:15:30.
LocalTime	A time without a time-zone in the ISO-8601 calendar system, such as 10:15:30.
MonthDay	A month-day in the ISO-8601 calendar system, such as --12-03.
OffsetDateTime	A date-time with an offset from UTC/Greenwich in the ISO-8601 calendar system, such as 2007-12-03T10:15:30+01:00.

After Example, these  
classes will be used

# Java API Documents

- The list and usage of Java APIs are organized systematically in this site.  
(<https://docs.oracle.com/javase/8/docs/api/>)
- Let's find a list of candidates for the package that we expect to have the logic we need.

# Java API Documents

## 1. Java API Package List (Built-In)

java.awt.print  
java.beans  
java.beans.beancontext  
java.io  
java.lang  
java.lang.annotation  
java.lang.instrument  
java.lang.invoke  
java.lang.management  
java.lang.ref  
java.lang.reflect  
java.math  
java.net

## 2. Classes List of the selected Package

Boolean  
Byte  
Character  
Character.Subset  
Character.UnicodeBlock  
Class  
ClassLoader  
ClassValue  
Compiler  
Double  
Enum  
Float  
InheritableThreadLocal  
Integer  
Long  
Math  
Number  
Object  
Package  
Process  
ProcessBuilder  
ProcessBuilder.Redirect  
Runtime  
RuntimePermission  
SecurityManager  
Short  
StackTraceElement  
StrictMath  
String  
StringBuffer  
StringBuilder

### Fields

Modifier and Type	Field and Description
static double	<b>E</b> The double value that is closer than any other to <i>e</i> , the base of the natural logarithms.
static double	<b>PI</b> The double value that is closer than any other to <i>pi</i> , the ratio of the circumference of a circle to its diameter.

### Method Summary

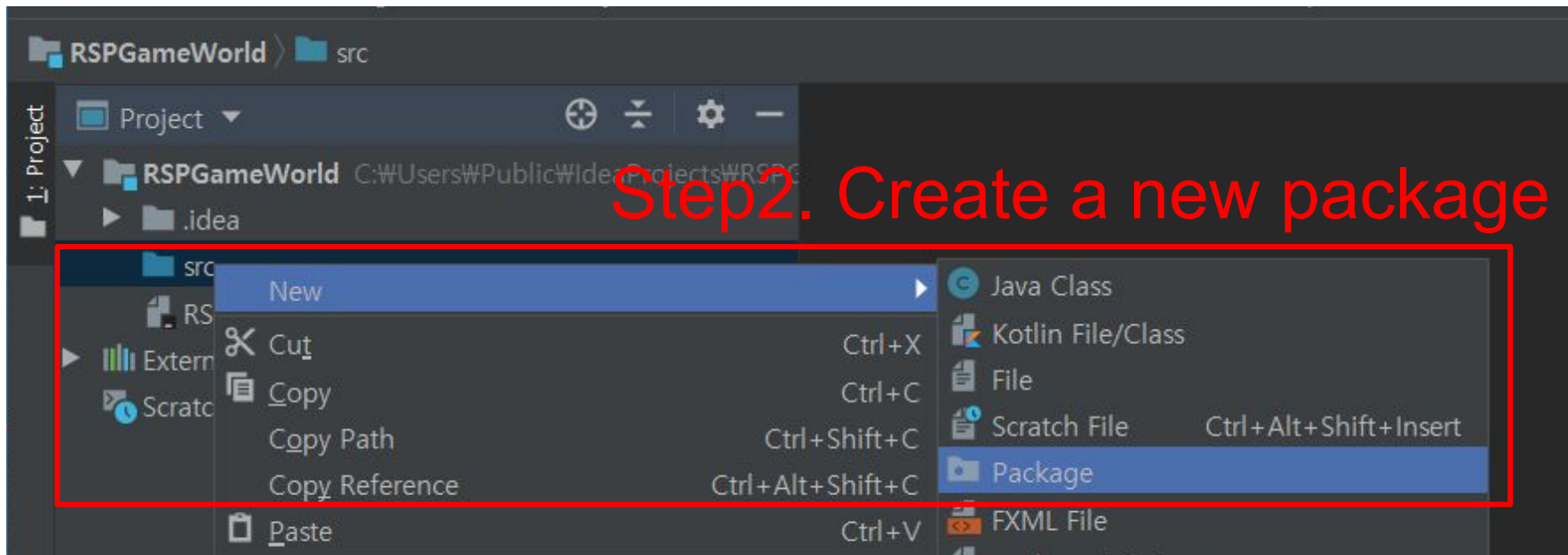
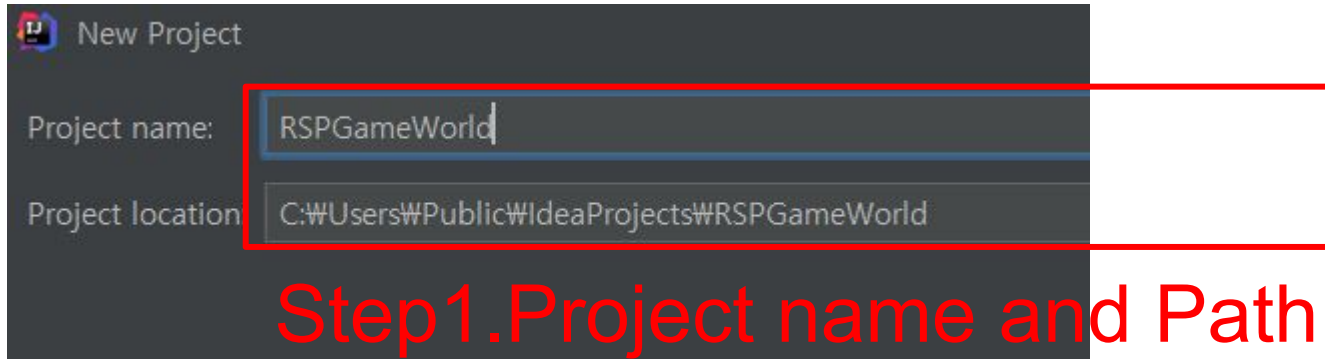
#### All Methods Static Methods Concrete Methods

Modifier and Type	Method and Description
static double	<b>abs(double a)</b> Returns the absolute value of a double value.
static float	<b>abs(float a)</b> Returns the absolute value of a float value.
static int	<b>abs(int a)</b> Returns the absolute value of an int value.
static long	<b>abs(long a)</b> Returns the absolute value of a long value.
static double	<b>acos(double a)</b> Returns the arc cosine of a value; the returned angle is in the range 0.0 through <i>pi</i> .
static int	<b>addExact(int x, int y)</b> Returns the sum of its arguments, throwing an exception if the result overflows an int.
static long	<b>addExact(long x, long y)</b> Returns the sum of its arguments, throwing an exception if the result overflows a long.
static double	<b>asin(double a)</b> Returns the arc sine of a value; the returned angle is in the range $-\pi/2$ through $\pi/2$ .

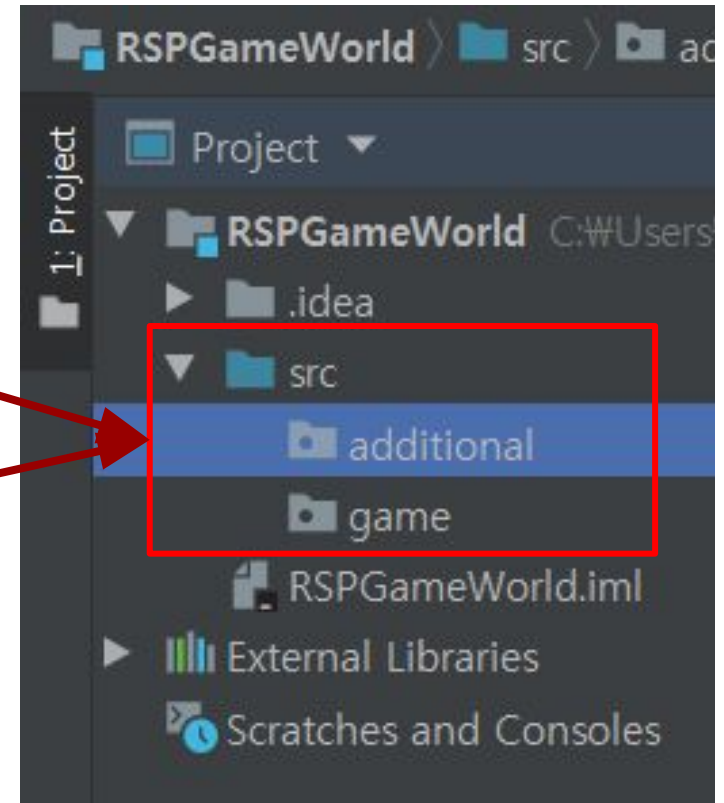
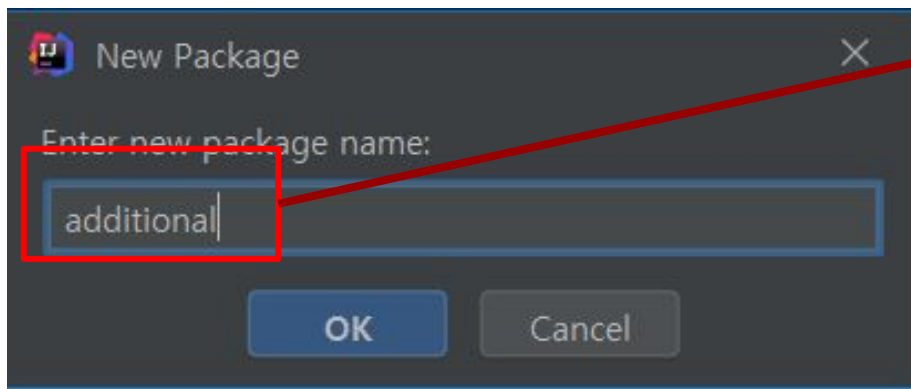
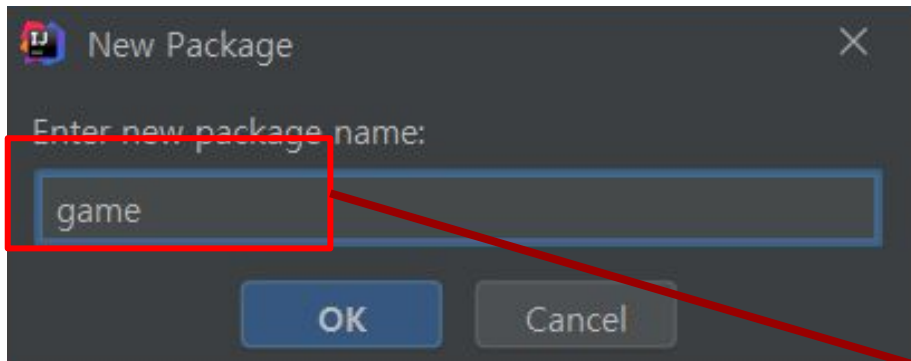
# Encapsulation Example - overview

- “Rock Scissors Paper Game”
  
- This Project is consisted of 2 Modules
  - game package :  
Provides components and UI for the game.
    - RSPGame.java, GameSystem.java
  - additional package :  
Provides components for additional services
    - Analyzer.java, Summary.java

# Create Packages

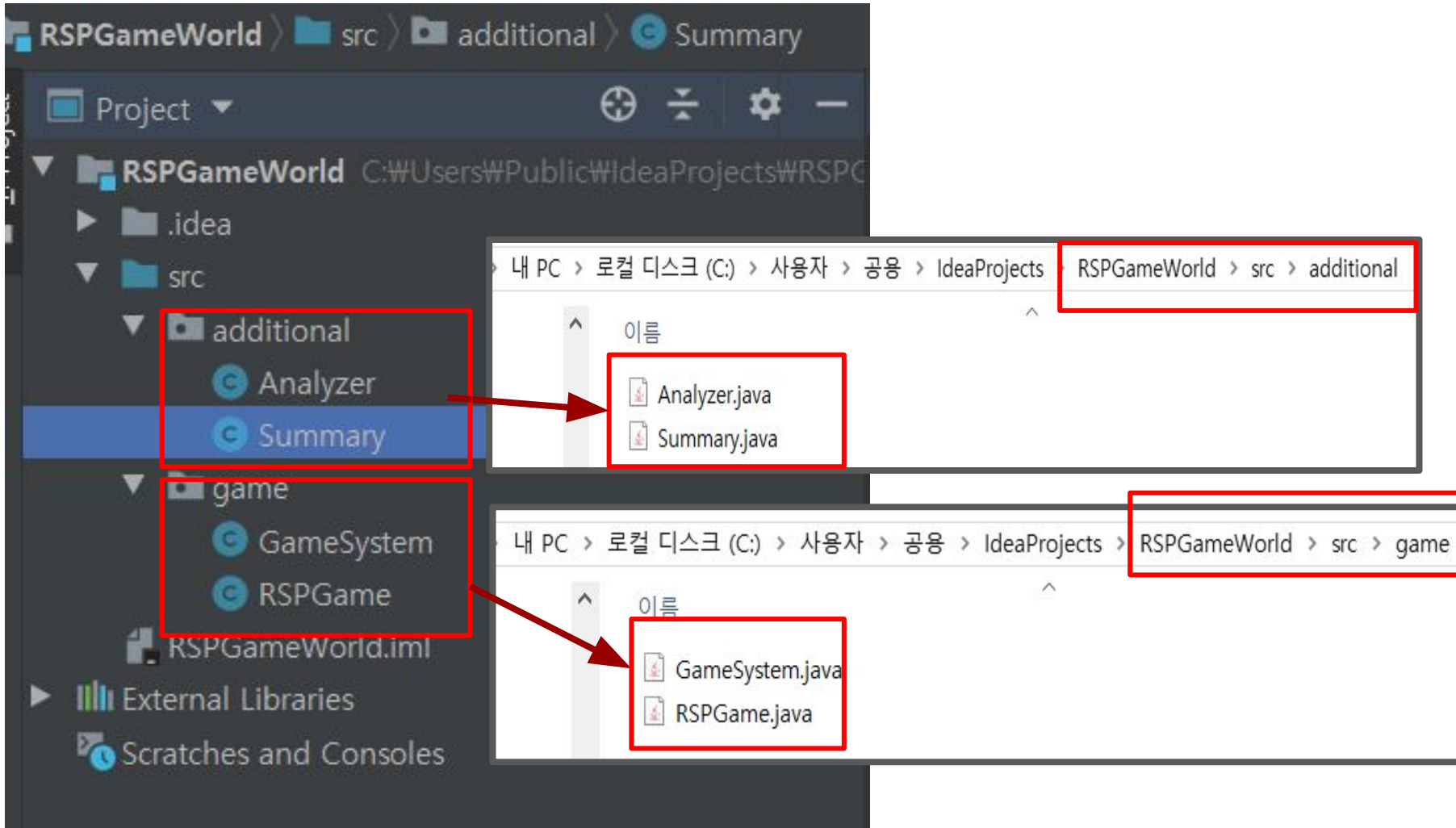


# Create Packages



Step4. Create 2 packages Under the src folder.  
(‘game’ and ‘additional’)

# Create Class files



The screenshot illustrates the process of creating class files in the `RSPGameWorld` project. The `Project` tool window on the left shows the project structure:

- `RSPGameWorld` (C:\Users\WPublic\IdeaProjects\WRSPC)
  - `.idea`
  - `src`
    - `additional`
      - `Analyzer`
      - `Summary`
    - `game`
      - `GameSystem`
      - `RSPGame`

Two file explorer windows are overlaid, showing the file creation process:

- Top Explorer:** Shows the path `RSPGameWorld > src > additional`. The files `Analyzer.java` and `Summary.java` are listed under the heading `이름` (Name).
- Bottom Explorer:** Shows the path `RSPGameWorld > src > game`. The files `GameSystem.java` and `RSPGame.java` are listed under the heading `이름` (Name).

Red boxes and arrows highlight the mapping between the project structure and the file explorer windows.

# Project Description

- `GameSystem.java` : offer the UI and overall systems between game program and User.
- `Game.java` : offer the “Rock Scissors Paper Game” with computer system  
( 2 mode : i) Draw Allow ii) Draw Not Allow)
- `Analyzer.java` : Analyze module the result of game
- `Summary.java` : offer summarized informations of results



# GameSystem.java Structure

```
package game;
```

GameSystem.java

```
// Import Sentences
```

```
public class GameSystem {
```

```
    // Field members will be filled
```

```
    public void main(String[] arg) {
```

```
        // Overall System flow will be filled
```

```
    }
```

```
    // Method members will be filled
```


```
}
```

# GameSystem.java

- In our system, we don't need a string of GameSystem instances.  
=> We will manage the system with **static** variables.

## Field Members

```
private static int rounds;  
private static boolean mode;  
private static RSPGame.History[] histories;  
private static Summary summary;  
static RSPGame rspGame;  
import additional.Summary;
```



Related fields with game settings

# GameSystem.java

Method members

```
private static void setEnvironment() {  
    // set static fields of GameSystem Class  
}  
  
private static boolean setRounds(int num) {  
    // check the validation of setting Rounds values  
}  
  
private static boolean setMode(int num) {  
    // check the validation of mode value and determine  
    draw's valid  
}
```

# GameSystem.java

- Overall flow of system and communication with User will be offered by main() Method in GameSystem.java

main() Method

```
// Overall System flow will be filled

// Step1. Get the values about game setting from
user input. Then create the environment for game.
// Step2. Proceed games
// Step3. Report the end of games
// Step4. Show the summarized result of the games
```

# GameSystem.java

main() Method

```
setEnvironment(); // Step1
```

```
for (int i = 0; i < rounds; i++) {  
    histories[i] = rspGame.playGame(); // Step2  
}
```

```
System.out.println("Game is over!\n"); // Step3
```

```
summary = new Summary(histories); // Step4  
summary.printSummary();
```

# GameSystem.java

```
private static void setEnvironment() -(1)
```

```
System.out.println("Welcome to RSP GAME World!\n");  
System.out.println("How many times? ( with natural  
number)");
```

```
Scanner scanner = new Scanner(System.in);
```

```
import java.util.Scanner;
```

```
int input;
```

```
do {  
    input = scanner.nextInt();  
} while (!setRounds(input));
```

Until pass the validation

# GameSystem.java

```
private static void setEnvironment() - (2)
```

```
System.out.println("Draw Allow? (Yes : 1 /No : 0)");
```

```
do {  
    input = scanner.nextInt();  
} while (!setMode(input));
```

```
rspGame = new RSPGame(mode);
```

```
String modeName;
```

# GameSystem.java

```
private static void setEnvironment() - (3)
```

```
if (mode) {  
    modeName = "Draw Valid Mode";  
} else {  
    modeName = "Draw not Valid Mode";  
}  
  
System.out.println(rounds + " game(s) will be started( "  
    + modeName + " )\n");
```

```
histories = new RSPGame.History[rounds];
```

Access inner Class from External



# GameSystem.java

```
private static boolean setRounds(int num)

if (num > 0) {
    rounds = num;
    return true;
}

return false;
```

Validation and Setting

# GameSystem.java

```
private static boolean setMode(int num)

if (num == 0) {
    mode = false;
    return true;
} else if (num == 1) {
    mode = true;
    return true;
} else {
    System.out.println("Not a valid mode");
}

return false;
```

# RSPGame.java

```
package game;
```

```
// import sentences
```

```
public class RSPGame {
```

```
    // field members and constructor
```

```
    History playGame() { }
```

```
    private int checkUserInput(String userInput) { }
```

```
    private int generateSystemPick() { }
```

```
    private void recordHistory(int userPick,  
                               int systemPick) { }
```

```
    public class History{ /* inner class */ }
```

```
}
```

RSPGame.java  
Overall Structure

# History Class

## History Class Definition

```
public class History{  
    // field members  
    // Constructor  
    public boolean printHistory() {  
        // validaiton check,print the histories  
    }  
    private void setPresentDT() {  
        // set the present Data and Time  
    }  
    public int getResult() {  
        // get the result of the game  
    }  
}
```

# History Class

- History Class's field must be written when history is recorded (in RSPGame Class : outer class)

## Field Members

```
private int round;  
private String userPick;  
private String systemPick;  
private String result;  
private LocalDate date;  
private LocalTime time;  
private boolean update = false;
```

Few accesses  
from external.

# History Class

- A History instance will be created by round value
- To use LocalDate & LocalTime Class, import java.time.\* package

## Constructor

```
public History(int round) {  
    this.round = round;  
}
```

```
import java.time.*;
```

```
private void setPresentDT()
```

```
date = LocalDate.now();  
time = LocalTime.now();
```

# History Class

```
public boolean printHistory()
```

```
if (!update) {  
    System.out.println("History is not valid.");  
    return false;  
}
```

```
System.out.println("\n< Result of Round " + round + " >");  
System.out.println("Date : " + date);  
System.out.println("Time : " + time);  
System.out.println("User Pick : " + userPick);  
System.out.println("System Pick : " + systemPick);  
System.out.println("Game result : " + result);  
return true;
```

# History Class

- find the index of game result for external accesses  
(if not valid => return -1 )

```
public int getResult()
```

```
for (int i = 0; i < resultCases.length; i++) {  
    if (resultCases[i].equals(result)) {  
        return i;  
    }  
}  
  
return -1;
```

filed from RSPGame

```
private static String[] resultCases =  
    {"Draw!\n", "User Win\n", "User Lose!\n"};
```



# RSPGame.java

Field Members

```
private int round = 0;
private static String[] rsp =
    {"scissors", "rock", "paper"};
private static String[] resultCases =
    {"Draw!\n", "User Win!\n", "User Lose!\n"};
private History history ;
private boolean drawValid;
private Scanner scanner;
```

```
RSPGame(boolean drawValid) {
    this.drawValid = drawValid;
    this.scanner = new Scanner(System.in);
}
```

Constructor  
Method

# RSPGame.java

History playGame() -(1)

```
this.round++;  
history = new History(this.round);  
System.out.println("Round " + round);  
String input;  
int userPick;  
int systemPick;  
  
do {  
    systemPick = generateSystemPick();  
    System.out.println("Please enter one of the 'scissors',  
'rock', or 'paper'");  
    input = scanner.next();  
    userPick = checkUserInput(input);  
  
    ...  
}
```

# RSPGame.java

History playGame() -(2)

...

**Pick Again!**

```
if(!drawValid && (systemPick == userPick)) {
```

```
    System.out.println("Draw case! pick one more time for  
this round !");
```

```
    userPick = -1;
```

```
}
```

```
} while (userPick < 0);
```

**Not Valid pick => Pick Again!**

```
recordHistory(userPick, systemPick);
```

```
history.printHistory();
```

```
return history;
```

# RSPGame.java

```
private int checkUserInput(String userInput)

for (int i = 0; i < rsp.length; i++) {
    if (rsp[i].equals(userInput)) {
        return i;
    }
}

System.out.println("Not valid input!\n");
return -1;
```

```
private int generateSystemPick()
```

```
return (int) (3 * Math.random());
```

Static Method

# RSPGame.java

```
private void recordHistory(int userPick, int systemPick)
int result = (userPick - systemPick + 3) % 3;
history.userPick = rsp[userPick];
history.systemPick = rsp[systemPick];
history.result = resultCases[result];
history.setPresentDT();
history.update = true;
```

```
private static String[] rsp =
    {"scissors", "rock", "paper"}; // idx = 0, 1, 2
```

Lose : My index is smaller 1(Larger 2) than opponent's

# RSPGame.java(package & import)

```
package game;  
  
import java.util.Scanner;  
import java.time.*;
```

# Analyzer.java Structure

```
package additional;

import game.RSPGame;

public class Analyzer {

    // Field Members

    public Analyzer(RSPGame.History[] histories) { }
    private void updateRecords(RSPGame.History history) { }
    private float computWiningRate() { }
    public float getWinningRate() { }

}
```

# Analyzer.java

```
private float winningRate;  
int TotalGames = 0;  
int TotalWins = 0;  
int TotalLosses = 0;  
int TotalDraws = 0;  
boolean IsComputed = false;
```

Field Members

```
public Analyzer(RSPGame.History[] histories)
```

```
TotalGames = histories.length;  
for (RSPGame.History history : histories) {  
    updateRecords(history);  
}
```



# Analyzer.java

```
private void updateRecords(RSPGame.History history)

int result = history.getResult();
switch (result) {
    case 0:
        TotalDraws++;
        break;
    case 1:
        TotalWins++;
        break;
    case 2:
        TotalLoses++;
        break;
    default:
        System.out.println("Not valid history");
}
```

# Analyzer.java

```
private float computWiningRate()  
  
winningRate = (float) TotalWins / TotalGames;  
isComputed = true;  
return winningRate;
```

Check the  
Setting of  
winningRate  
Value

```
private float computWiningRate()  
  
if (!isComputed) {  
    computWiningRate();  
}  
  
return winningRate;
```

# Summary.java Structure

```
package additional;

import game.RSPGame;

public class Summary {

    Analyzer analyzer;

    public Summary(RSPGame.History[] histories) { }
    public void printSummary() { }

}
```

# Summary.java

```
public Summary(RSPGame.History[] histories)
```

```
    analyzer = new Analyzer(histories);
```

```
    public void printSummary()
```

```
    {
        System.out.println("< Summary >\n" );
```

```
        System.out.println("TotalGames: " + analyzer.TotalGames);
```

```
        System.out.println("TotalWins: " + analyzer.TotalWins);
```

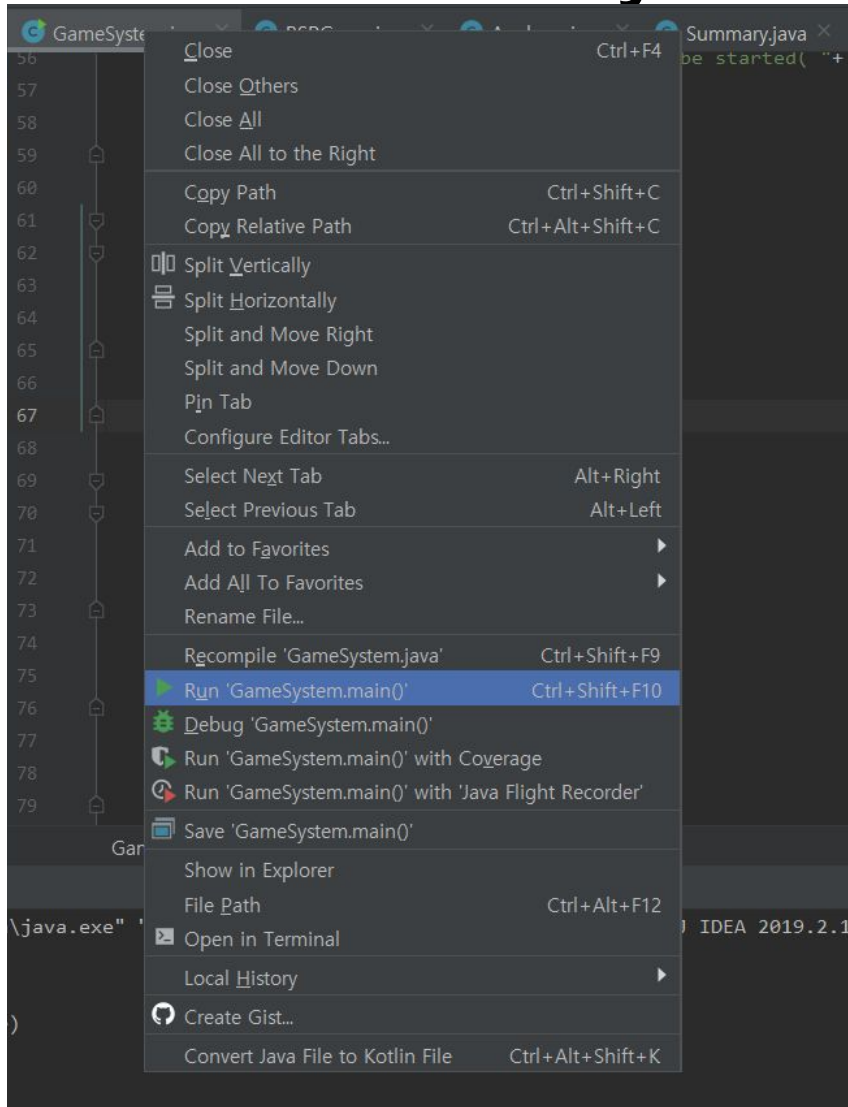
```
        System.out.println("TotalDraws: " + analyzer.TotalDraws);
```

```
        System.out.println("TotalLoses: " + analyzer.TotalLoses);
```

```
        System.out.println("Wining Rate: " +
```

```
            analyzer.getWinningRate());
```

# Run `GameSystem.main()`



# Outputs (Draw Valid Mode)

Welcome to RSP GAME World!

How many times? ( with natural number)

3

Draw Allow? (Yes : 1 /No : 0)

1

3 game(s) will be started( Draw Valid Mode )

Round 1

Please enter one of the 'scissors', 'rock', or 'paper'  
rock

< Result of Round 1 >

Date : 2019-09-25

Time : 10:40:07.642856400

User Pick : rock

System Pick : scissors

Game result : User Win

# Outputs (Draw Valid Mode)

Round 2

Please enter one of the 'scissors', 'rock', or 'paper'  
rock

< Result of Round 2 >

Date : 2019-09-25

Time : 10:40:11.181480100

User Pick : rock

System Pick : paper

Game result : User Lose!

Round 3

Please enter one of the 'scissors',  
'rock', or 'paper'  
paper

< Result of Round 3 >

Date : 2019-09-25

Time : 10:40:13.003058900

User Pick : paper

System Pick : scissors

Game result : User Lose!

Game is over!

# Outputs (Draw Valid Mode)

< Summary >

TotalGames: 3

TotalWins: 1

TotalDraws: 0

TotalLoses: 2

Wining Rate:

0.33333334





# Outputs (Draw Not Valid Mode)

Welcome to RSP GAME World!

How many times? ( with natural number)

4

Draw Allow? (Yes : 1 /No : 0)

0

4 game(s) will be started( Draw not Valid Mode )

Round 1

Please enter one of the 'scissors', 'rock', or 'paper'

rock

< Result of Round 1 >

Date : 2019-09-25

Time : 13:36:52.922228300

User Pick : rock

System Pick : paper

Game result : User Lose!

# Outputs (Draw Not Valid Mode)

Round 2

Please enter one of the 'scissors', 'rock', or 'paper'  
rock

< Result of Round 2 >

Date : 2019-09-25

Time : 13:36:54.477738200

User Pick : rock

System Pick : paper

Game result : User Lose!

# Outputs (Draw Not Valid Mode)

Round 3

Please enter one of the 'scissors', 'rock', or 'paper'  
rock

Draw case! pick one more time for this round !

Please enter one of the 'scissors', 'rock', or 'paper'  
rock

< Result of Round 3 >

Date : 2019-09-25

Time : 13:36:58.711766400

User Pick : rock

System Pick : paper

Game result : User Lose!

(Because, Draw is not  
valid case in this mode)

# Outputs (Draw Not Valid Mode)

Round 4

Please enter one of the 'scissors', 'rock', or 'paper'  
paper

< Result of Round 4 >

Date : 2019-09-25

Time : 13:37:01.430157400

User Pick : paper

System Pick : scissors

Game result : User Lose!

Game is over!

< Summary >

TotalGames: 4

TotalWins: 0

TotalDraws: 0

TotalLoses: 4

Wining Rate: 0.0