Create a class Player with below attributes:

playerId - int
playerName - String
runs - int
playerType - String
matchType - String

The above attributes should be private, write getters, setters and parameterized constructor as required.

Create class Solution with main method.

Implement two static methods - findPlayerWithLowestRuns and findPlayerByMatchType in Solution class.

findPlayerWithLowestRuns method:
This method will take array of Player objects and a String value as input parameters.
The method will return the least runs of the Player from array of Player objects for the given player type.
(String parameter passed). If no Player with the above condition are present in array of Player objects,
then the method should return 0.

findPlayerByMatchType method:
This method will take array of Player objects and String value as input parameters and return the array of Player objects
belonging to the match type passed as input parameter in Descending order of playerId.
If no Player with the above condition are present in the array of Player objects, then the method should return null.

Note : No two Players will have the same playerId and runs.
All the searches should be case insensitive.

The above mentioned static methods should be called from the main method.

For findPlayerWithLowestRuns method - The main method should print the returned runs as it is
if the returned value is greater than 0 or it should print "No such player".

Eg: 25
where 25 is the lowest runs of the Player.

For findPlayerByMatchType method - The main method should print the playerId from the returned Player array for each
Player if the returned value is not null.
If the returned value is null then it should print "No Player with given matchType".

Eg:

13
11
where 13 and 11 are the playerId's.

Before calling these static methods in main, use Scanner object to read the values of four Player objects referring attributes in the above mentioned attribute sequence.
Next, read the value of two String parameter for capturing player type and match Type.

Consider below sample input and output:

```
Input1:
11
Sachin
100
International
One day
12
Shewag
133
International
Test
13
Varun
78
State
Test
14
Ashwin
67
State
One day
State
One day

Output:
67
14
11


Input2:
11
Sachin
100
```

```
International
One day
12
Shewag
133
International
Test
13
Varun
78
State
Test
14
Ashwin
67
State
One day
District
T20
```

Output:

```
No such player
No Player with given matchType
```

Sample code snippet for reference:

Please use below code to build your Solution.

```
import java.util.Scanner;
public class Solution
{
public static void main(String[] args)
{
//code to read values
//code to call required method
//code to display the result
}

//code the first method


//code the second method
}

//code the class
```

Note on using Scanner object:
Sometimes scanner does not read the new line character while invoking methods like nextInt(), nextDouble() etc. Usually, this is not an issue, but this may be visible while calling nextLine() immediately after those methods.

Consider below input values:

```
1001
Savings
```

Referring below code:

```
Scanner sc = new Scanner(System.in);
int x = sc.nextInt();
String str = sc.nextLine(); -> here we expect str to have value Savings.Instead it may be "".
```

If above issue is observed, then it is suggested to add one more explicit call to nextLine() after reading numeric value.

Uploading Files

Instructions: Kindly mention class name as **MyClass** for Java, C# and Scala.

Start Date/Time	End Date/Time	Attempts	Marks Obtained	Status	Actions
14/02/23 09:37 AM	NA NA	1	NA	Started	• LAUNCH • •

My Attempts

1

- Attempt 1

Attempt Date/Time

14/02/2023

09:37 AM

Submission Date/Time

NA

NA

Marks Obtained

NA

Status

NOT STARTED

Download

Result

Next

Java Assessment - Test 2