



JOBSHEET 11

Recursive Function

1. Learning Outcomes

- Student must be able to master recursive function concept
- Student must be proficient to implement recursive function as a problem solving option

2. Labs Activity

2.1 Experiment 1

Times: 60 minutes

In this experiment, we will implement a program to calculate factorial number by using **recursive** function. An **iterative**-based factorial program will be developed also, for comparative reason.

1. Create a new Java class and save it as **RecursiveStudentIDNumber.java**
2. Create a new **static** function `factorialRecursive()` that has **int** datatype and 1 **int** parameter. This parameter will be used to pass a value to calculate factorial.

```
static int factorialRecursive(int n){  
    if(n==1)  
        return 1;  
    else  
        return n*factorialRecursive(n-1);  
}
```

3. Create a new **static** function to calculate factorial using iterative approach. Name the function as `factorialRecursive()`, with **int** datatype and also has 1 **int** parameter.

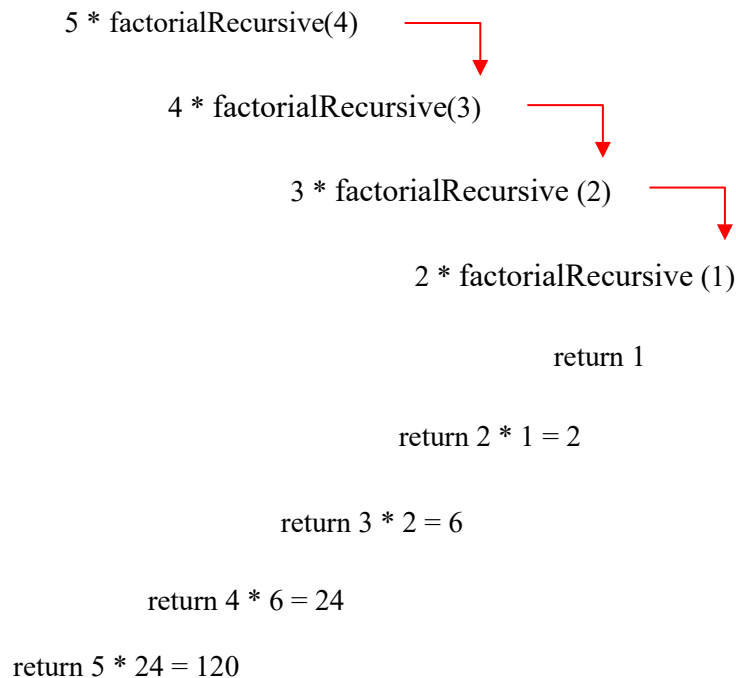
```
static int factorialIterative(int n){  
    int factorial = 1;  
    for(int i=n; i>=1; i--){  
        factorial = factorial*i;  
    }  
    return factorial;  
}
```

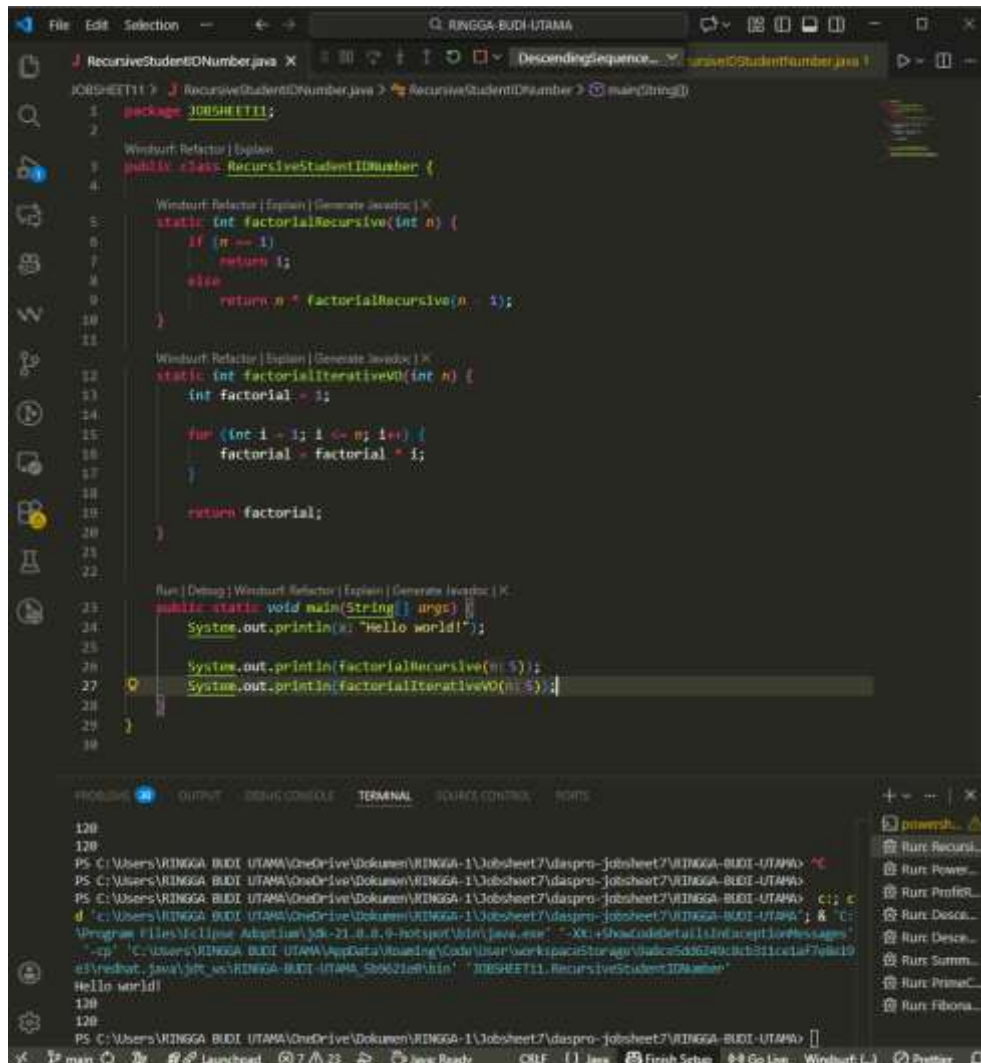
4. Create a **main** function to call both of recursive function by passing a value to be calculate the factorial number. Also display the result in this main function.



```
public static void main(String[] args) {
    System.out.println(x:"Hello world!!!");
    System.out.println(factorialRecursive(n:5));
    System.out.println(factorialIterative(n:5));
}
```

5. Run the code and observe the result!
6. When we call function `factorialRecursive(5)`, the following step illustrates how the recursive function called as well as how the result computed (in reverse of recursive calls):





```

1: package JOBSHEET11;
2:
3: Window: Refactor | Explain | Generate |avadoc | K...
4: public class RecursiveStudentIDNumber {
5:
6:     Window: Refactor | Explain | Generate |avadoc | K...
7:     static int factorialRecursive(int n) {
8:         if (n == 1)
9:             return 1;
10:        else
11:            return n * factorialRecursive(n - 1);
12:    }
13:
14:     Window: Refactor | Explain | Generate |avadoc | K...
15:     static int factorialIterative(int n) {
16:         int factorial = 1;
17:
18:         for (int i = 1; i <= n; i++) {
19:             factorial = factorial * i;
20:         }
21:
22:         return factorial;
23:     }
24:
25:     Run | Debug | Window: Refactor | Explain | Generate |avadoc | K...
26:     public static void main(String[] args) {
27:         System.out.println("Hello world!");
28:
29:         System.out.println(factorialRecursive(5));
30:         System.out.println(factorialIterative(5));
31:     }
32: }

```

```

120
PS C:\Users\RINGGA_BUDI_UTAMA\OneDrive\Documents\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>
PS C:\Users\RINGGA_BUDI_UTAMA\OneDrive\Documents\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>
PS C:\Users\RINGGA_BUDI_UTAMA\OneDrive\Documents\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA> C:\c
d 'C:\Users\RINGGA_BUDI_UTAMA\OneDrive\Documents\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA'; & C:\
Program Files\Eclipse Adoptium\jdk-21.0.8-9-hotspot\bin\java.exe' -XX:+ShowCodeDetails -Djava.awt.headless -cp 'C:\Users\RINGGA_BUDI_UTAMA\AppData\Local\Temp\workspace\storage\8d6c5d6249c8b311c01af7d8c19
e3f9ed8at.java\src\RINGGA_BUDI_UTAMA_S060218\bin' JOBSHEET11.RecursiveStudentIDNumber
Hello world!
120
PS C:\Users\RINGGA_BUDI_UTAMA\OneDrive\Documents\RINGGA-1\Jobsheet7\daspro-jobsheet7\RINGGA-BUDI-UTAMA>

```

Question

1. After observing the experiment of recursive function above, what is the definition of recursive function?
2. How the recursive function works?
3. From the experiment above, do factorialRecursive() and factorialIterative() have the similar result? Then, what are the differences between recursive and iterative if both are having the same result?

Answer :

1. **A recursive function is a function that calls itself during its execution until it reaches a stopping condition (base case).**
2. **How a recursive function works:**
 - o **The function keeps calling itself repeatedly with a smaller value.**



- *The process stops when it reaches the base case.*
- *After the base case is reached, the function starts returning back step-by-step.*

3. Yes, *factorialRecursive()* and *factorialIterative()* produce the same result.

The differences are:

- *Recursive uses self-calling functions.*
- *Iterative uses loops (for/while) without self-calling.*
- *Recursive is more elegant/shorter, while iterative is faster and uses less memory.*

2.2 Experiment 2

Times : 60 minutes

In the following experiment, a program to calculate power of a number using recursive function.

1. Create a new Java class, and name it as **PowerRecursiveIDStudentNumber.java**
2. Create a new **static** function **calculatePower()** that has **int** datatype with 2 int parameters (base and power number).

```
static int calculatePower(int base, int pow){
    if(pow==0)
        return 1;
    else
        return base*calculatePower(base, pow-1);
}
```

3. Create a new **main()** function that will get user input for base number and power number, will call the recursive function to calculate the power result and will display the final result.

```
Scanner input = new Scanner(System.in);
System.out.print(s:"Input Base Number: ");
int base = input.nextInt();
System.out.print(s:"Input Power Number: ");
int power = input.nextInt();
```

4. Call function **calculatePower()** and do not forget to pass 2 values into parameter, otherwise the error will arises.

```
System.out.println("Result of "+base+" power "+power+" = "+
    calculatePower(base, power));
```

5. Run the program and observe the result!

```
File Edit Selection View Help
Q: BRAGGUEUBI04M4
JOSHGETT1 > J: Powershell.exe .\PowerRecursive.java > Powershell.exe .\StudentNumber.java > calculatePower.ps1 40
New, 1 second ago | 1 author (You)
1 // calculatePower.ps1
2
3
4 Powershell> java -jar .\Scanner.jar
New, 1 second ago | 1 author (You) | Windows PowerShell | Explorer
powershell> java -jar .\PowerRecursive\StudentNumber.jar
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1
```

Question!

1. In Experiment 2, there is a recursive function call **calculatePower()** in the main function, then the function **calculatePower()** is called repeatedly. Explain how long the function calling process will run!

The recursive function `calculatePower()` will keep calling itself as long as the power value is greater than 0. Each recursive call decreases the power by 1, and the process stops when `power == 0` (the base case). So the function runs until the power reaches 0, then it returns the result back step-by-step.



2. Add program code to print the power calculation series. Example: `calculatePower(2,5)` will print $2 \times 2 \times 2 \times 2 \times 2 = 32!$

2.3 Experiment 3

Time : 60 minutes

We will continue this experiment by creating a new program to calculate the amount of investor money that will be used to invest after earning profits for several years. This program must implement recursive function to calculate the profits that would be got by investor.

1. Create a new class with name **ProvitRecursiveStudentIDNumber.java**
2. Create a new **static** function **calculateProfit()** that has **double** datatype and 2 **int** parameters. The **balance** would be passed through first parameter and **investment period** would be passed through parameter 2. The investor will get 11% interest annually.

Since to calculate profit is **interest * balance**, so the total amount of investor money will be **balance+interest*balance**. In this case, the amount of **interest** is **0.11 * balance**, and the final balance is considered as **1*balance**, so that **1 * balance + 0.11 *** or equivalent formula to calculate the final balance will be **1.11 * balance** in a year.

```
static double calculateProfit(double balance, int period){
    if(period==0)
        return balance;
    else
        return 1.11*calculateProfit(balance, period-1);
}
```

3. Create a new **main()** function, then get the user input for initial **balance** and **investment period**.

```
System.out.print(s:"Input Balance: ");
double initialBalance = input.nextInt();
System.out.print(s:"Input Investment Period: ");
int investPeriod = input.nextInt();
```

4. Then call function **calculateProfit()** from main function.

```
System.out.println("Balance after "+base+" year = "+
    calculateProfit(initialBalance, investPeriod));
```

5. Run the program and observe the result!



```

package JOESHEET11;

import java.util.Scanner;

public class ProfitRecursiveIDStudentNumber {

    static double calculateProfit(double Balance,
                                int period) {
        if (period == 0)
            return Balance;
        else
            return 1.11 * calculateProfit(Balance,
                                          period - 1);
    }

    static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.print("Input Balance: ");
        double initialBalance = input.nextDouble();

        System.out.print("Input Investment Period: ");
        int InvestPeriod = input.nextInt();

        System.out.println("Balance after " + InvestPeriod + " years = " + calculateProfit(initialBalance,
                                                                                          InvestPeriod));
    }
}
    
```

```

C:\Users\UTAMA\AppData\Local\Microsoft\Windows\CurrentVersion\Explorer\RecentItems\Program Files\Java\jdk-21.0.2\bin\java.exe -DXX-UseCodeCacheInExceptionHandler -cp "C:\Users\UTAMA\Documents\KIRKGA-1\Johsheet7\Uaspro-Johsheet7\KIRKGA-BUDI-UTAMA\src\java\JOESHEET11\ProfitRecursiveIDStudentNumber" JOESHEET11.ProfitRecursiveIDStudentNumber
Input Balance: 60
Input Investment Period: 88
Balance after 88 year = 253586.7658282293
    
```

Question!

- From the above experiment, which statements that is classified as “base case” and “recursion call”!

Base case : if (period == 0)

return balance;

recursion call : return 1.11 * calculateProfit(balance, period - 1);

- Explain using simulation or trace the expansion phase and substitution phase of calculateProfit(100000,3) function!

Expansion phase :

calculateProfit(100000,3)

= 1.11 * calculateProfit(100000,2)

$= 1.11 * \text{calculateProfit}(100000,1)$

$= 1.11 * \text{calculateProfit}(100000,0)$

Substituion phase :

$\text{calculateProfit}(100000,0) = 100000$

$= 1.11 * 100000 = 111000$

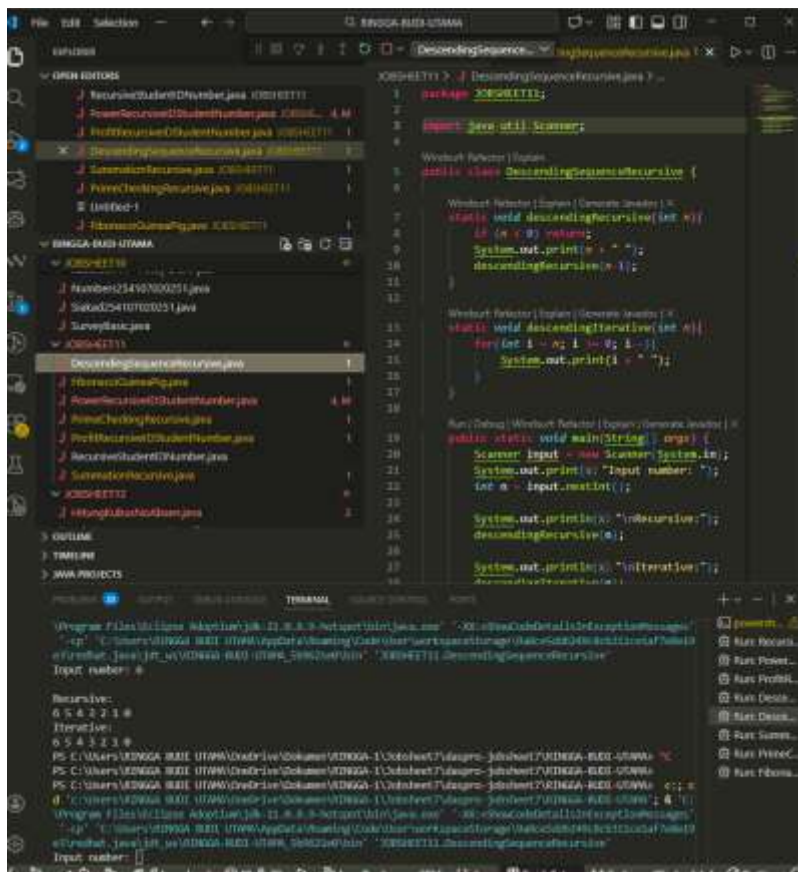
$= 1.11 * 111000 = 123210$

$= 1.11 * 123210 = 136761$

3. Assignment

Times: 120 minutes

1. Write a program to display numbers n to 0 using recursive functions and iterative functions. (DescendingSequenceRecursive).



```

package X00042711;

import java.util.Scanner;

public class DescendingSequenceRecursive {

    // Window: Refactor | Display | Generate | Invoke | ...
    // static void descendingRecursive(int n) {
    //     if (n <= 0) return;
    //     System.out.print(n + " ");
    //     descendingRecursive(n-1);
    // }

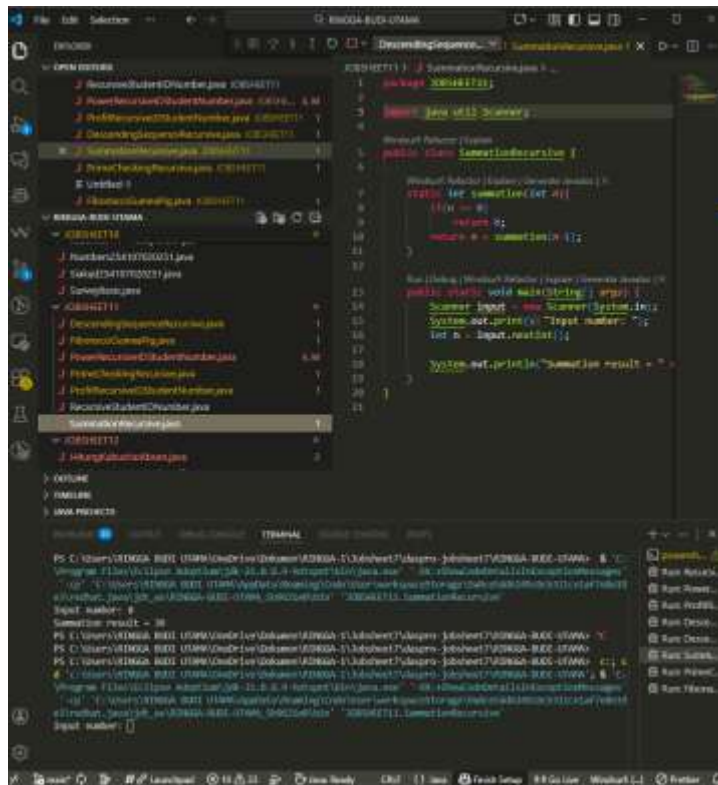
    // Window: Refactor | Display | Generate | Invoke | ...
    // static void descendingIterative(int n) {
    //     for (int i = n; i >= 0; i--) {
    //         System.out.print(i + " ");
    //     }
    // }

    // Run: Debug | Window: Refactor | Display | Generate | Invoke | ...
    // public static void main(String[] args) {
    //     Scanner input = new Scanner(System.in);
    //     System.out.print("Input number: ");
    //     int n = input.nextInt();
    //     System.out.println();
    //     System.out.println(x); "Recursive";
    //     descendingRecursive(n);
    //     System.out.println(x); "Iterative";
    //     descendingIterative(n);
    // }

    Input number: 6

    Recursive:
    6 5 4 3 2 1 0
    Iterative:
    6 5 4 3 2 1 0
  
```


2. Create a program to sum the numbers using recursive function. For example $n = 8$, then it will result $1+2+3+4+5+6+7+8 = 36$ (**SummationRecursive**).



```
package com.kodinga.recursion;

import java.util.Scanner;

public class SummationRecursive {

    public static int sum(int n) {
        if (n == 0) {
            return 0;
        }
        return n + sum(n-1);
    }

    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        System.out.print("Input number: ");
        int n = input.nextInt();

        System.out.println("Summation result = " + sum(n));
    }
}
```

3. Create a program that contains a recursive function to check whether a number n is a prime number or not. A number will not be classified as a prime number if it is divisible by a number less than n . (**PrimeCheckingRecursive**).

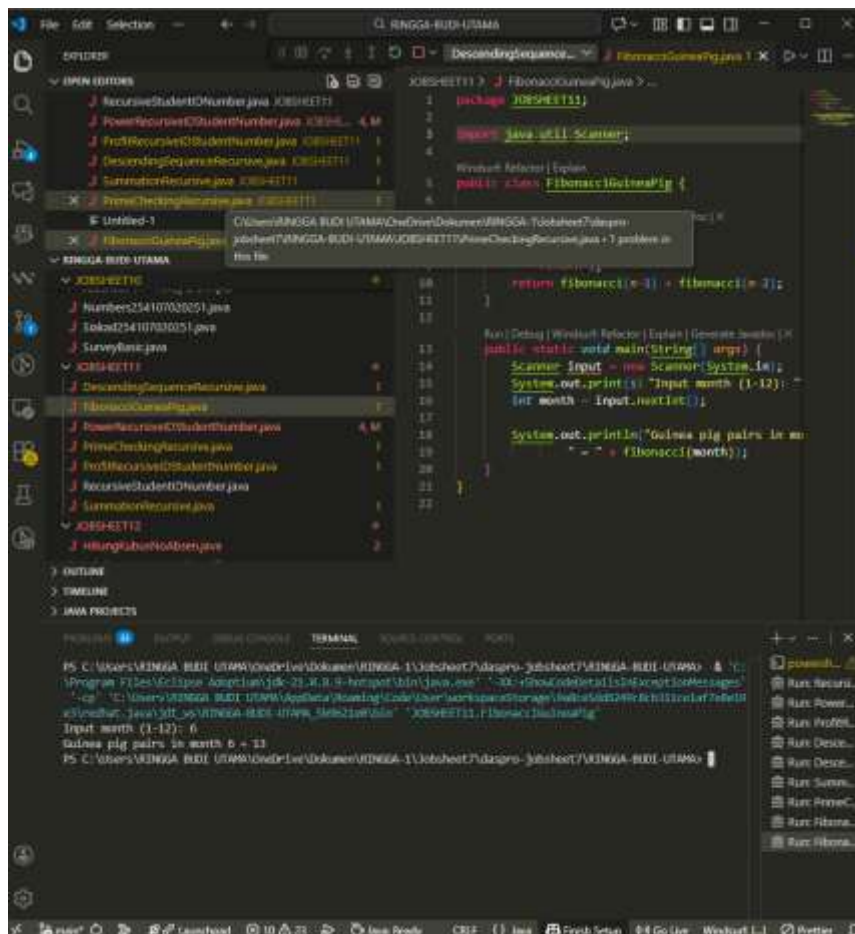


```

1 package 2023041111;
2
3 import java.util.Scanner;
4
5 public class PrimeCheckingRecursive {
6
7     // Method to check if a number is prime (Recursive function)
8     static boolean isPrime(int n, int i) {
9         if (i == n)
10             return false;
11         if (i == 1)
12             return true;
13         if (n % i == 0)
14             return false;
15         return isPrime(n, i + 1);
16     }
17
18     // Main method to test the isPrime function
19     public static void main(String[] args) {
20         Scanner input = new Scanner(System.in);
21         System.out.print("Input number: ");
22         int n = input.nextInt();
23
24         if (isPrime(n, 1))
25             System.out.println(n + " is prime");
26         else
27             System.out.println(n + " is NOT prime");
28     }
29 }
    
```

4. A pair of newborn guinea pigs (male and female) are placed in a nursery. After two months the guinea pig pair gave birth to a pair of twin guinea pigs (male and female). Every pair of guinea pigs that is born will also give birth to a pair of guinea pigs every 2 months. How many pairs of guinea pigs were there at the end of the 12th month? Write a program using a recursive function! (**Fibonacci**). And the following table illustrates the calculation.

Month	Pair Number		Pair Total
	Productive	Non-Productive	
1	0	1	1
2	0	1	1
3	1	1	2
4	1	2	3
5	2	3	5
6	3	5	8
7	5	8	13
8	8	13	21
9	13	21	34
10	21	34	55
11	34	55	89
12	55	89	144



```
1 package X089EET11;
2
3 import java.util.Scanner;
4
5 /**
6  * Fibonacci Guinea Pig
7  */
8
9 public class FibonacciGuineaPig {
10
11     // Fibonacci sequence
12     public static int fibonacci(int n) {
13         if (n == 1 || n == 2) {
14             return 1;
15         }
16         return fibonacci(n - 1) + fibonacci(n - 2);
17     }
18
19     public static void main(String[] args) {
20         Scanner input = new Scanner(System.in);
21         System.out.print("Input month (1-12): ");
22         int month = input.nextInt();
23
24         System.out.println("Guinea pig pairs in month " +
25             month + " = " + fibonacci(month));
26     }
27 }
```