CSCI 1130 Introduction to Computing Using Java

Tutorial 9

Problem Solving Technique

- Enumeration

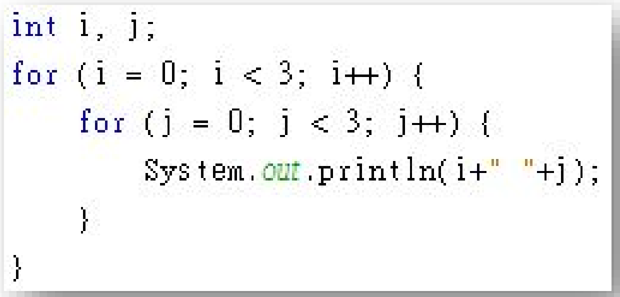
Topics

- Problem solving technique : Enumeration
  - Basic idea
  - Three sample problems
  - Classwork
- Assignment 5

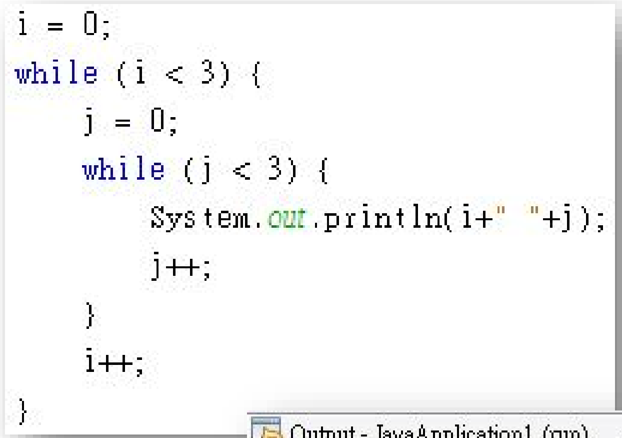
This technique is also known as “brute-force search” or “exhaustive search”.

Preliminaries

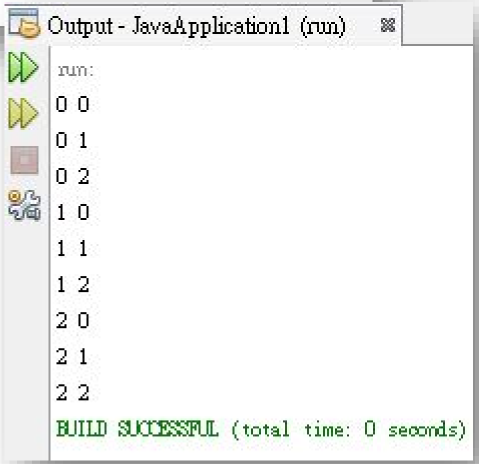
- Nested loops
  - For / while loops



```
int i, j;
for (i = 0; i < 3; i++) {
    for (j = 0; j < 3; j++) {
        System.out.println(i+" "+j);
    }
}
```



```
i = 0;
while (i < 3) {
    j = 0;
    while (j < 3) {
        System.out.println(i+" "+j);
        j++;
    }
    i++;
}
```



```
run:
0 0
0 1
0 2
1 0
1 1
1 2
2 0
2 1
2 2
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Conditional statements
  - If-(else) statements
  - Boolean operators

Example 1

- There are two positive integers, A and B
- You know that:
  - $A - B \geq 5$
  - $A \times B \leq 6$
- Question: A = ? B = ?

- One possible solution: Try all $1 \leq A, B \leq 6$
  - A = 1, B = 1 : First condition violated; try next
  - A = 1, B = 2 : …
- This may not be very clever, but is easy to code

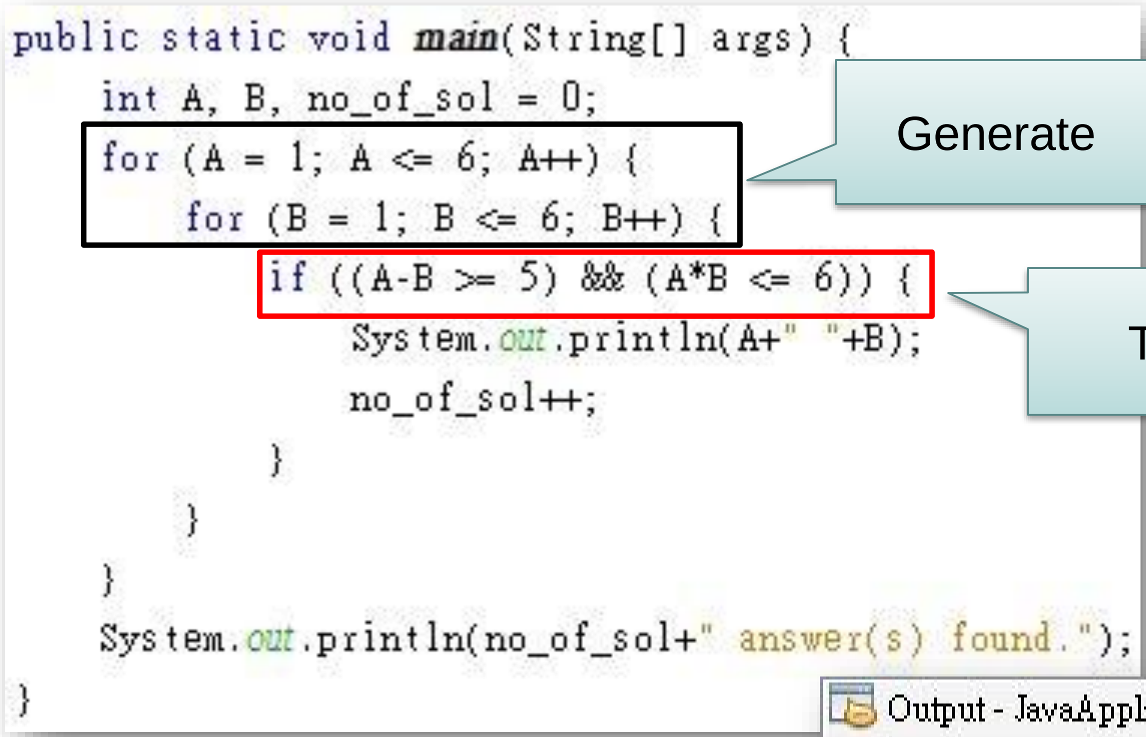
Enumeration

- "Generate and test"
  - Generate all possible solutions
  - Test them one by one
  - If the conditions are satisfied, we have found our answer
  - If not, try the next one

How to generate all possibilities?

How to formulate the conditions?

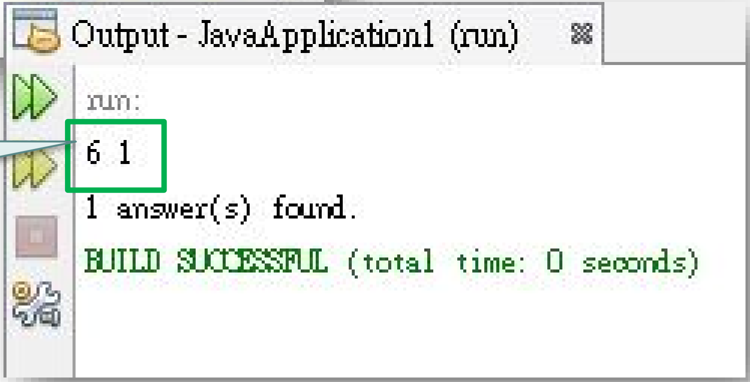
Example 1



```
public static void main(String[] args) {
    int A, B, no_of_sol = 0;
    for (A = 1; A <= 6; A++) {
        for (B = 1; B <= 6; B++) {
            if ((A-B >= 5) && (A*B <= 6)) {
                System.out.println(A+" "+B);
                no_of_sol++;
            }
        }
    }
    System.out.println(no_of_sol+" answer(s) found.");
}
```

Generate

Test



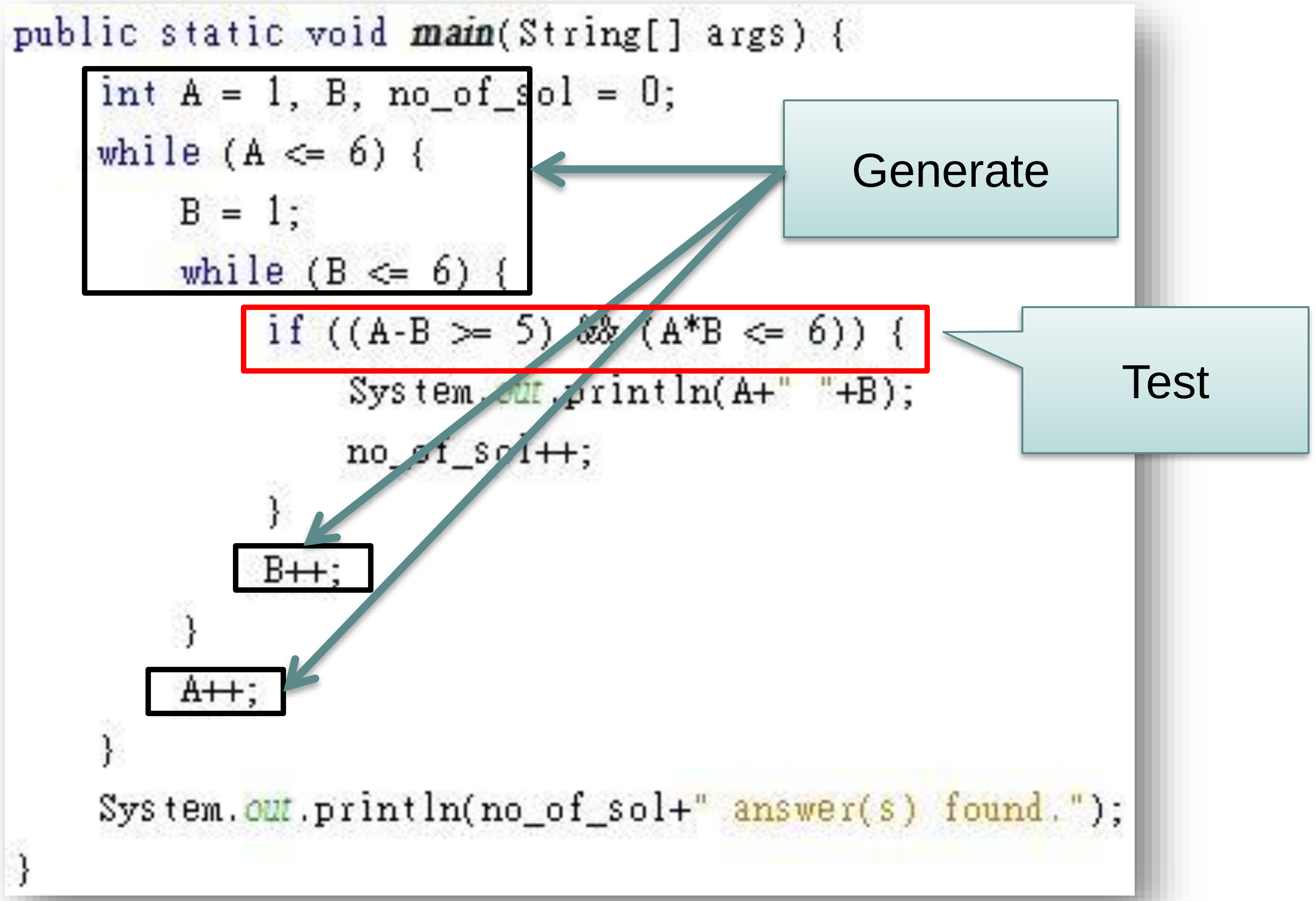
Output - JavaApplication1 (run)

```
run:
6 1
1 answer(s) found.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Answer: A = 6, B = 1

Example 1

- You may also use while loop



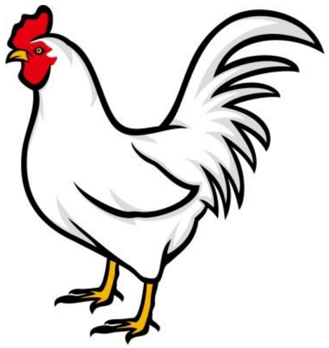
```
public static void main(String[] args) {
    int A = 1, B, no_of_sol = 0;
    while (A <= 6) {
        B = 1;
        while (B <= 6) {
            if ((A-B >= 5) && (A*B <= 6)) {
                System.out.println(A+" "+B);
                no_of_sol++;
            }
            B++;
        }
        A++;
    }
    System.out.println(no_of_sol+" answer(s) found.");
}
```

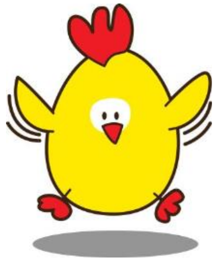
Example 2

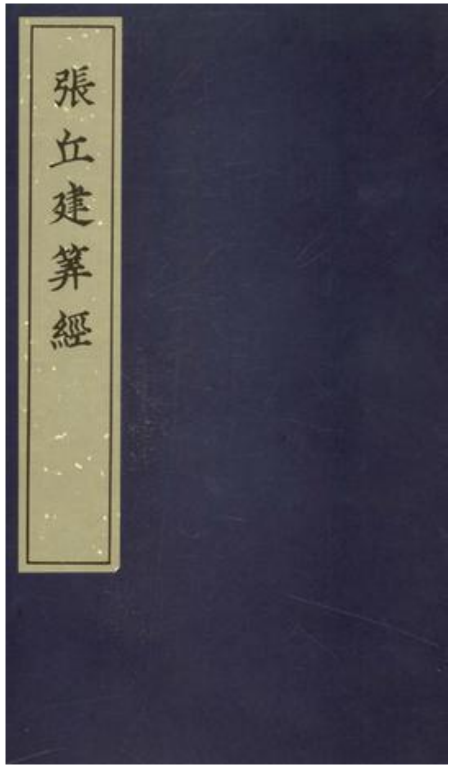
「百錢買百雞問題」：

雞翁一，值錢五，雞母一，值錢三，雞雛三，值錢一，百錢買百雞，問翁、母、雛各幾何？

Given that $5 for 1 rooster, $3 for 1 hen, $1 for 3 chicks and some one has bought 100 chicken with $100, how many roosters, hens and chicks did he buy?

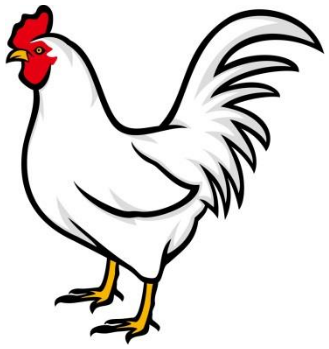






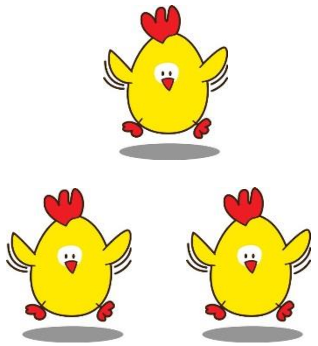
https://img3.doubanio.com/lpic/s8539723.jpg

Example 2



$5

$3



$1

$100

100 in total:

Example 2

- Hints:
  - X: the number of roosters.
  - Y: the number of hens.
  - Z: the number of chicks.
  - The following equations hold:

$$X + Y + Z = 100$$

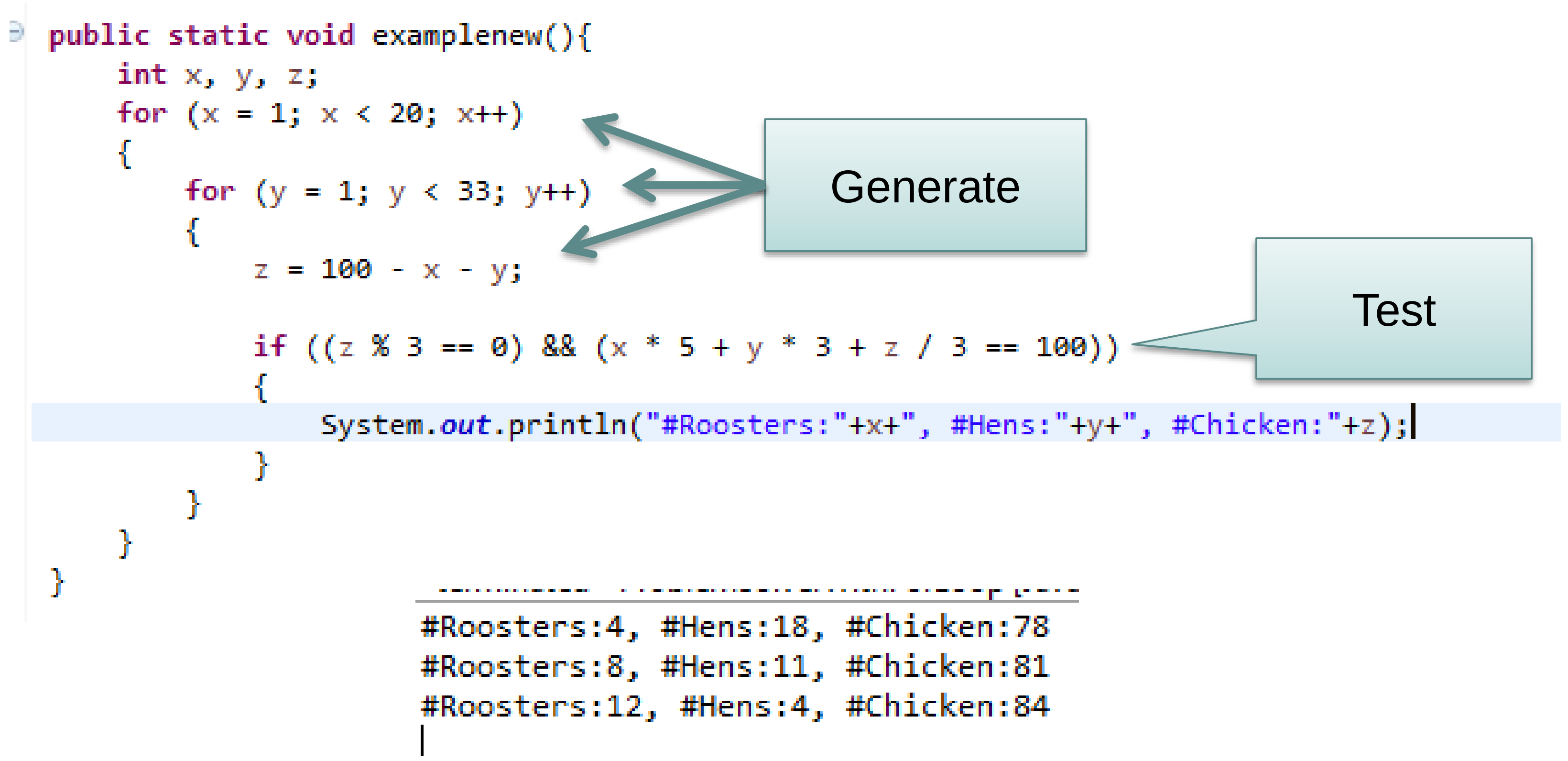
$$5X + 3Y + Z/3 = 100$$

  - X, Y and Z must be non-negative integers:

$$X < 20, \mathrm{Y} < 33 \text{ and } \mathrm{Z} = 100 - \mathrm{X} - \mathrm{Y}$$

Example 2

- Using For Loop:

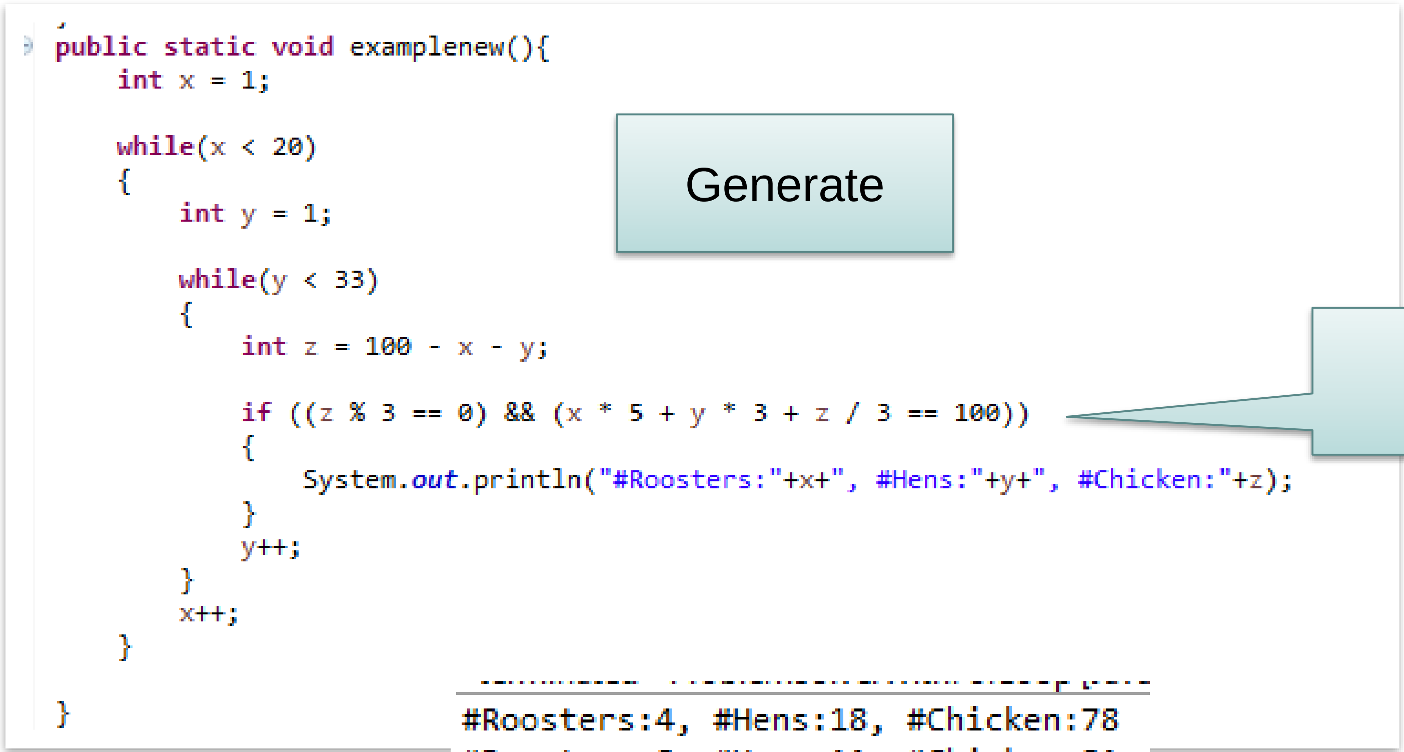


```
public static void examplenew(){
    int x, y, z;
    for (x = 1; x < 20; x++)
    {
        for (y = 1; y < 33; y++)
        {
            z = 100 - x - y;

            if ((z % 3 == 0) && (x * 5 + y * 3 + z / 3 == 100))
            {
                System.out.println("#Roosters:"+x+", #Hens:"+y+", #Chicken:"+z);
            }
        }
    }
}
```

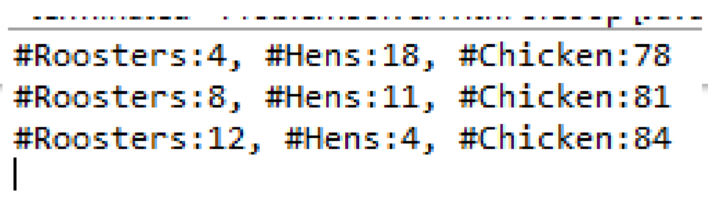
```
#Roosters:4, #Hens:18, #Chicken:78
#Roosters:8, #Hens:11, #Chicken:81
#Roosters:12, #Hens:4, #Chicken:84
```

Example 2

- Using While Loop:



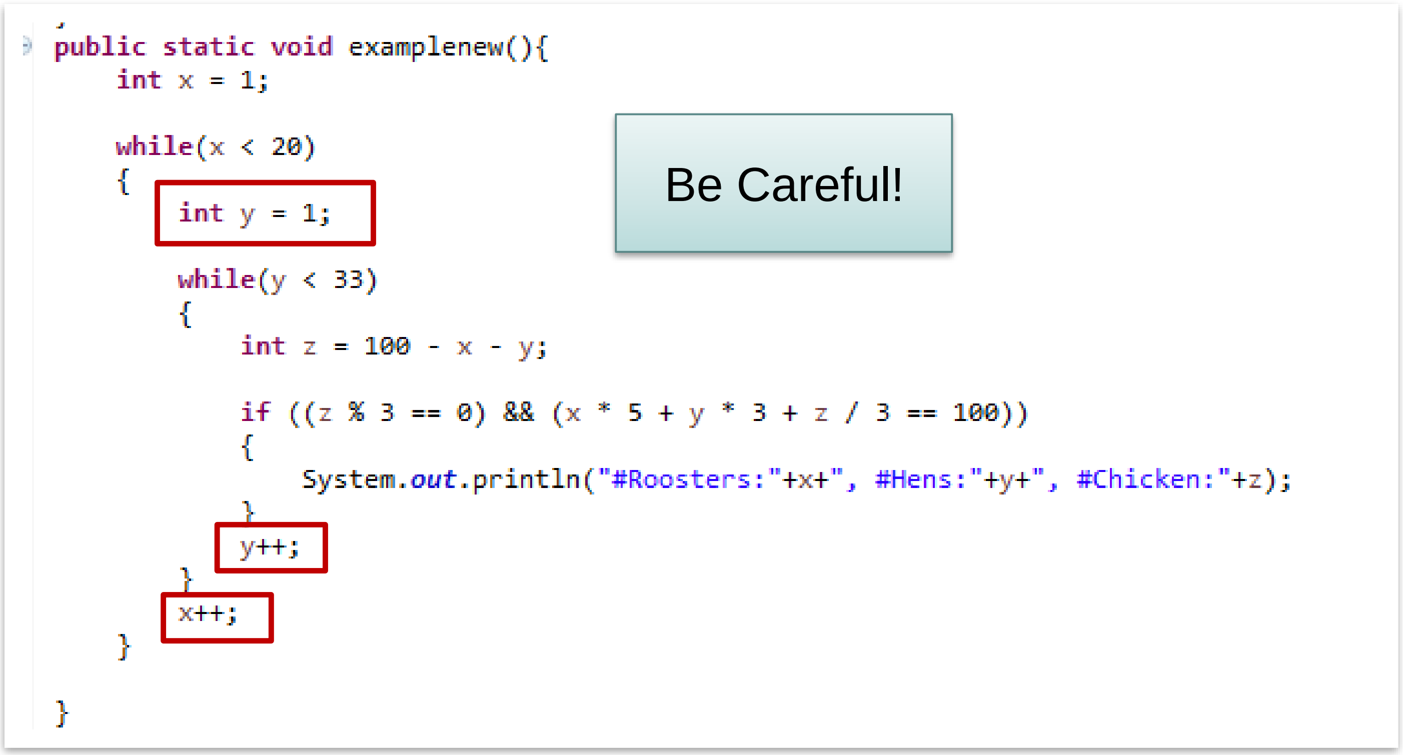
```
public static void examplenew(){
    int x = 1;

    while(x < 20)
    {
        int y = 1;

        while(y < 33)
        {
            int z = 100 - x - y;

            if ((z % 3 == 0) && (x * 5 + y * 3 + z / 3 == 100))
            {
                System.out.println("#Roosters:"+x+", #Hens:"+y+", #Chicken:"+z);
            }
            y++;
        }
        x++;
    }

}
```



```
#Roosters:4, #Hens:18, #Chicken:78
#Roosters:8, #Hens:11, #Chicken:81
#Roosters:12, #Hens:4, #Chicken:84
```

Example 2

- Using While Loop:

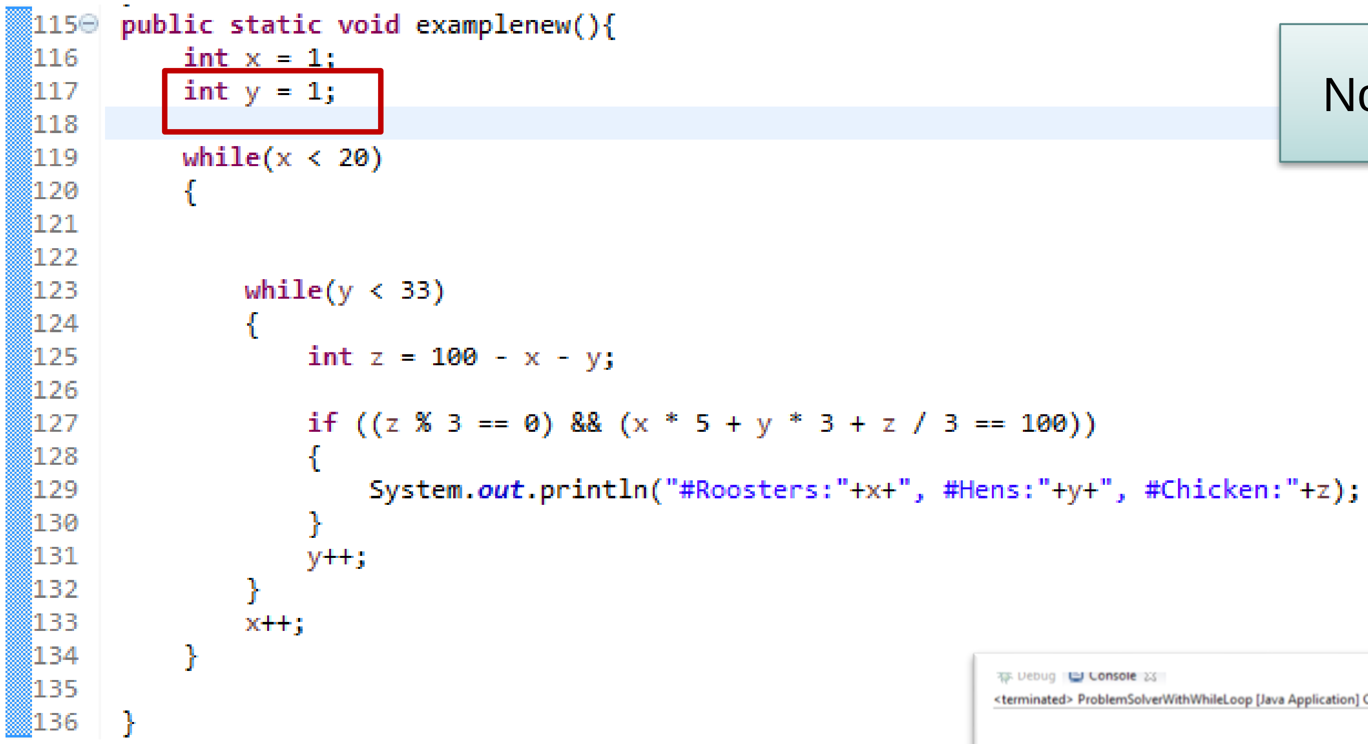


```
public static void examplenew(){
    int x = 1;

    while(x < 20)
    {
        int y = 1;

        while(y < 33)
        {
            int z = 100 - x - y;

            if ((z % 3 == 0) && (x * 5 + y * 3 + z / 3 == 100))
            {
                System.out.println("#Roosters:"+x+", #Hens:"+y+", #Chicken:"+z);
            }
            y++;
        }
        x++;
    }

}
```

Be Careful!

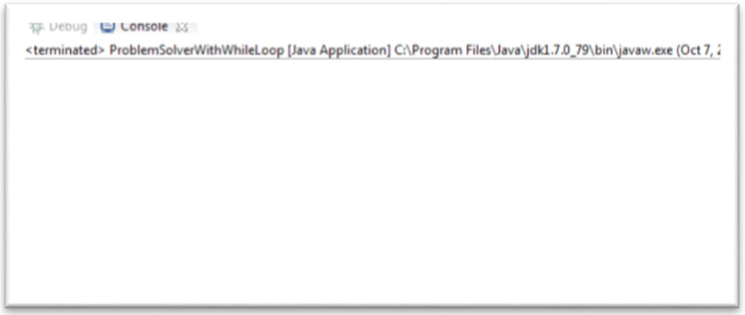
Example 2

- What if?



```java
public static void examplenew(){
    int x = 1;
    int y = 1;

    while(x < 20)
    {


        while(y < 33)
        {
            int z = 100 - x - y;

            if ((z % 3 == 0) && (x * 5 + y * 3 + z / 3 == 100))
            {
                System.out.println("#Roosters:"+x+", #Hens:"+y+", #Chicken:"+z);
            }
            y++;
        }
        x++;
    }

}
```

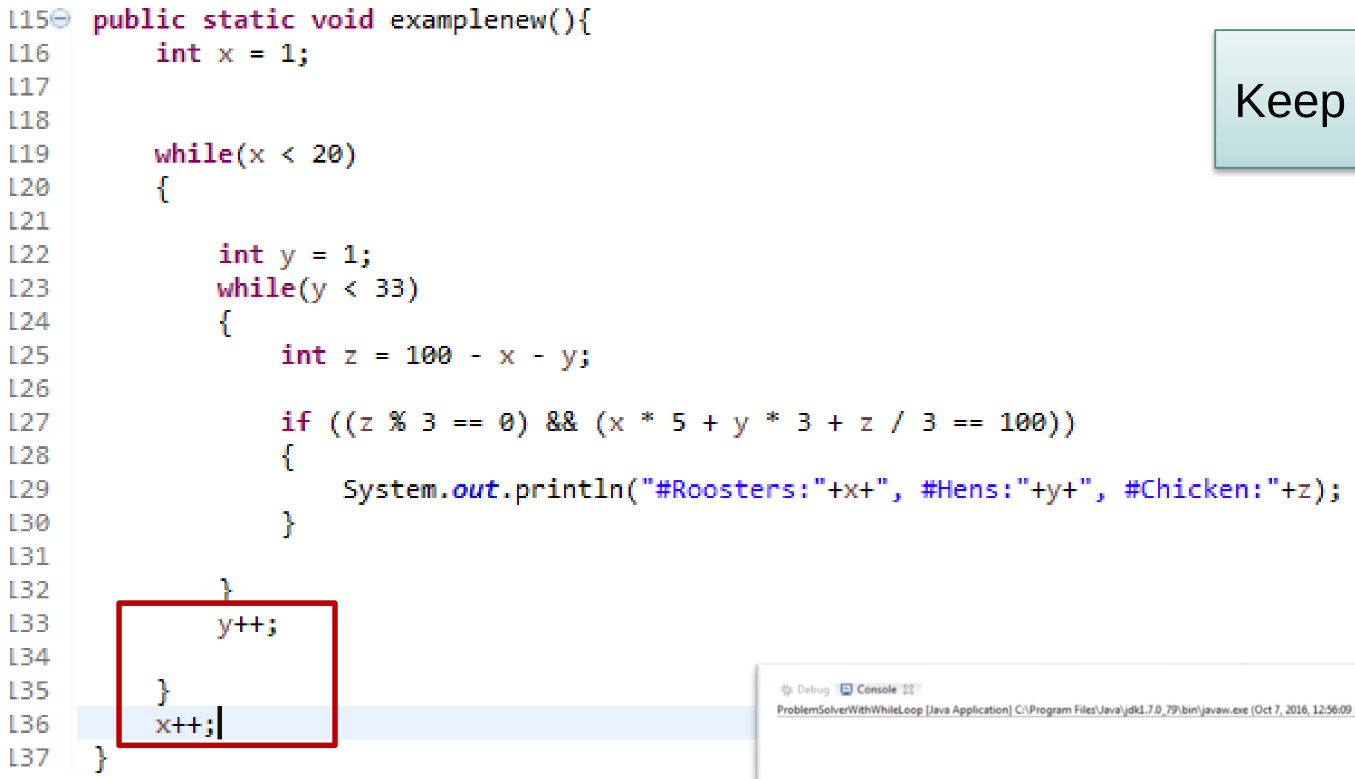
No solution!



```
<terminated> ProblemSolverWithWhileLoop [Java Application] C:\Program Files\Java\jdk1.7.0_79\bin\javaw.exe (Oct 7,
```

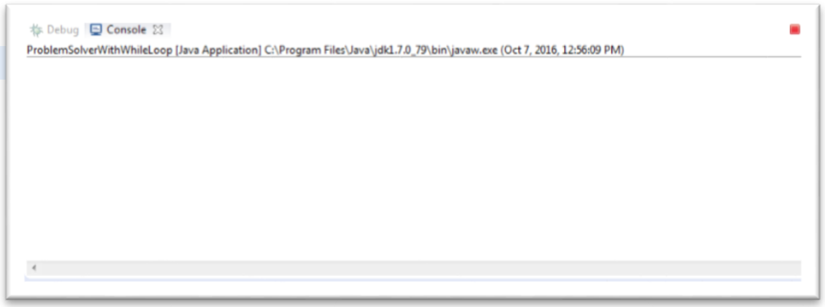
Example 2

- What if?



```java
public static void examplenew(){
    int x = 1;


    while(x < 20)
    {

        int y = 1;
        while(y < 33)
        {
            int z = 100 - x - y;

            if ((z % 3 == 0) && (x * 5 + y * 3 + z / 3 == 100))
            {
                System.out.println("#Roosters:"+x+", #Hens:"+y+", #Chicken:"+z);
            }

        }
        y++;

    }
    x++;
}
```

Keep running!



```
ProblemSolverWithWhileLoop [Java Application] C:\Program Files\Java\jdk1.7.0_79\bin\javaw.exe (Oct 7, 2016, 12:56:09 PM)
```

Classwork

$100

How many possible combinations?

Reminder: the total number of coins is not specified.

Classwork

$100

Surprisingly, 541

possible combinations!

Example 3

- Five divers, A, B, C, D and E, join a diving competition and are asked to predict the results
  - A: I will be the 3rd and B will be the 2nd
  - B: I will be the 2nd and E will be the 4th
  - C: I will be the 1st and D will be the 2nd
  - D: I will be the 3rd and C will be the 5th
  - E: I will be the 4th and A will be the 1st

- It turns out that the everyone's prediction is only half right (so one guess is correct and the other is wrong)
- Question: What is the real ranking?

Problem taken from *Fundamentals of Programming*, Wenhu Wu

Example 3

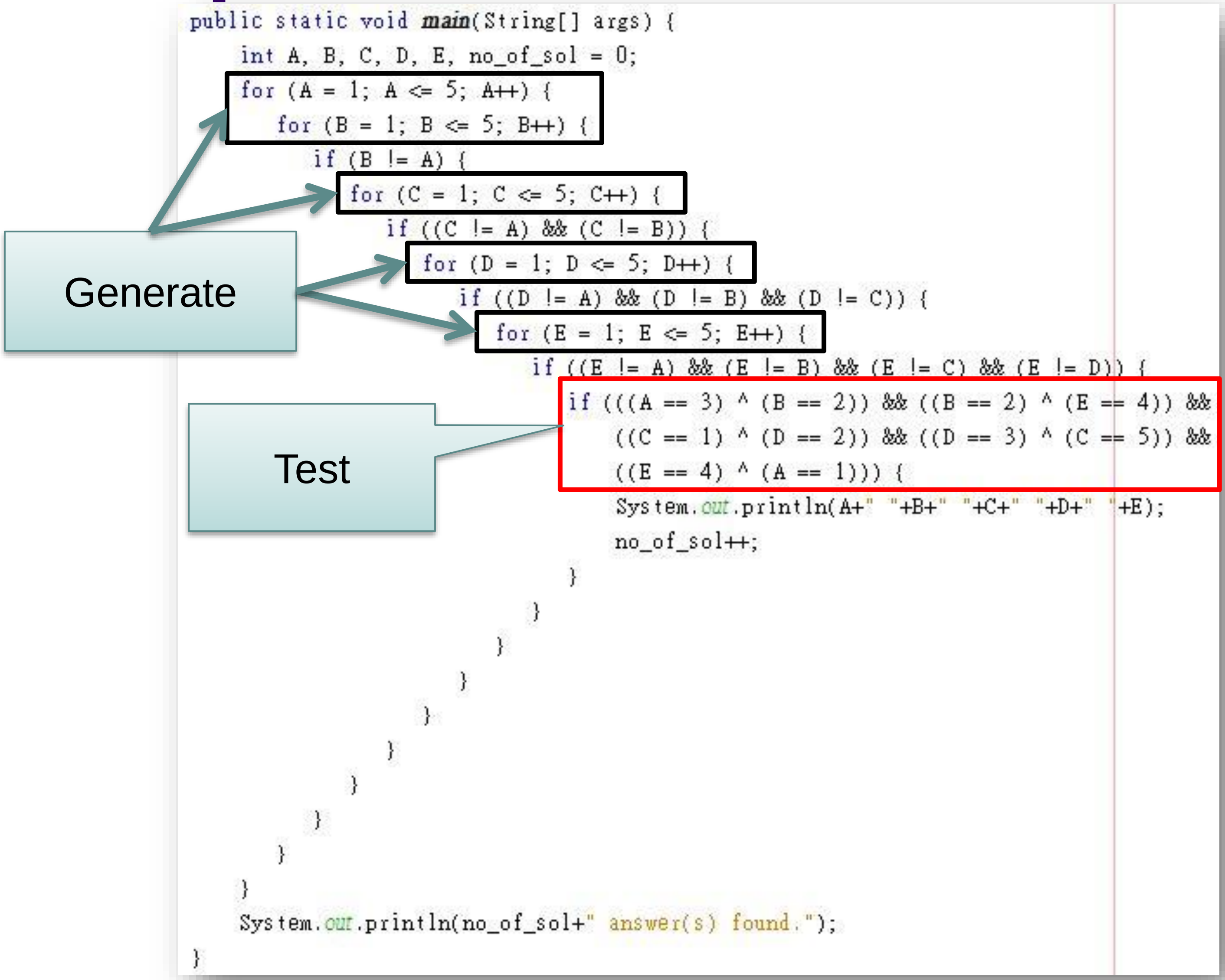
- Generate all possibilities
  - Five for loops
- Test for the conditions
  - Take A's prediction as an example
    - ((A == 3) && (B != 2)) || ((A != 3) && (B == 2))
  - Alternatively, use exclusive-OR:
    - (A == 3) ^ (B == 2)

Not all parentheses are necessary, but with them we can see the order more easily.

XOR returns true when exactly one side holds.

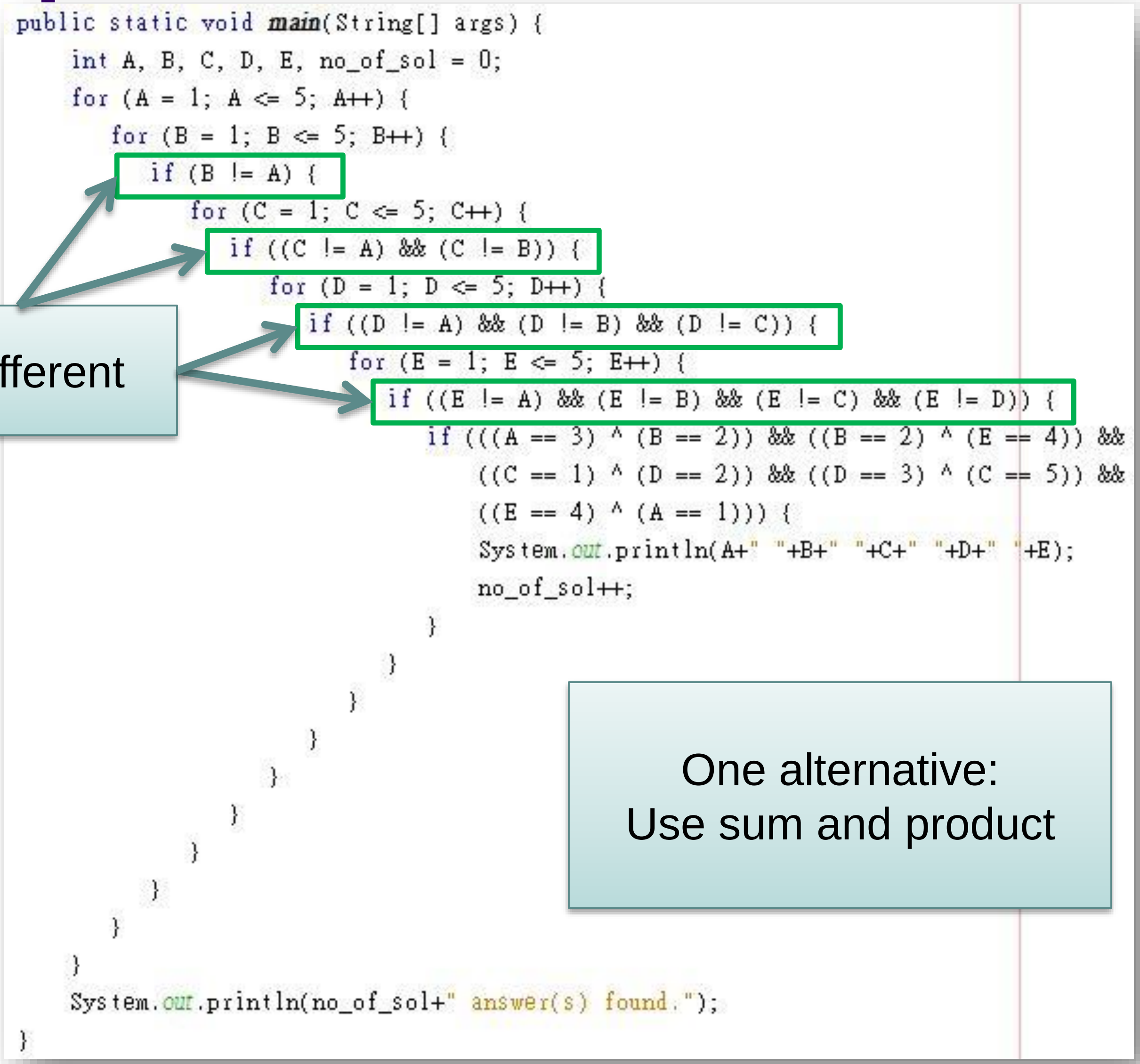
- *One more constraint:* two divers cannot share the same rank (Alldifferent)
  - This is a bit harder… well, at least for now

Example 3



```
public static void main(String[] args) {
    int A, B, C, D, E, no_of_sol = 0;
    for (A = 1; A <= 5; A++) {
        for (B = 1; B <= 5; B++) {
            if (B != A) {
                for (C = 1; C <= 5; C++) {
                    if ((C != A) && (C != B)) {
                        for (D = 1; D <= 5; D++) {
                            if ((D != A) && (D != B) && (D != C)) {
                                for (E = 1; E <= 5; E++) {
                                    if ((E != A) && (E != B) && (E != C) && (E != D)) {
                                        if (((A == 3) ^ (B == 2)) && ((B == 2) ^ (E == 4)) &&
                                            ((C == 1) ^ (D == 2)) && ((D == 3) ^ (C == 5)) &&
                                            ((E == 4) ^ (A == 1))) {
                                            System.out.println(A+" "+B+" "+C+" "+D+" "+E);
                                            no_of_sol++;
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
    System.out.println(no_of_sol+" answer(s) found.");
}
```

Example 3



```
public static void main(String[] args) {
    int A, B, C, D, E, no_of_sol = 0;
    for (A = 1; A <= 5; A++) {
      for (B = 1; B <= 5; B++) {
        if (B != A) {
          for (C = 1; C <= 5; C++) {
            if ((C != A) && (C != B)) {
              for (D = 1; D <= 5; D++) {
                if ((D != A) && (D != B) && (D != C)) {
                  for (E = 1; E <= 5; E++) {
                    if ((E != A) && (E != B) && (E != C) && (E != D)) {
                      if (((A == 3) ^ (B == 2)) && ((B == 2) ^ (E == 4)) &&
                          ((C == 1) ^ (D == 2)) && ((D == 3) ^ (C == 5)) &&
                          ((E == 4) ^ (A == 1))) {
                          System.out.println(A+" "+B+" "+C+" "+D+" "+E);
                          no_of_sol++;
                      }
                    }
                  }
                }
              }
            }
          }
        }
      }
    }
    System.out.println(no_of_sol+" answer(s) found.");
}
```

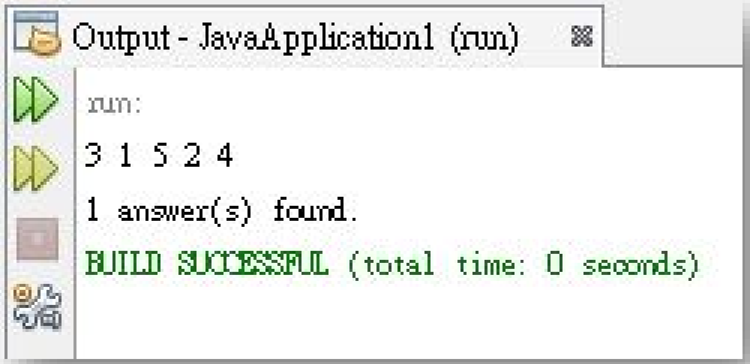
Alldifferent

One alternative:
Use sum and product

Example 3

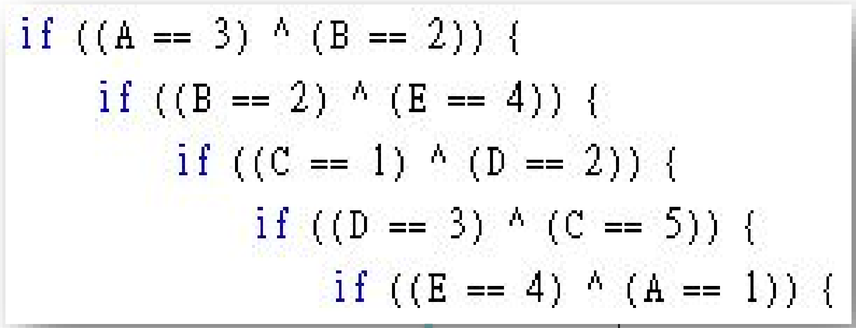
- Answer:
  - A: 3rd
  - B: 1st
  - C: 5th
  - D: 2nd
  - E: 4th

A: **I will be the 3rd** and B will be the 2nd
B: I will be the 2nd and **E will be the 4th**
C: I will be the 1st and **D will be the 2nd**
D: I will be the 3rd and **C will be the 5th**
E: **I will be the 4th** and A will be the 1st



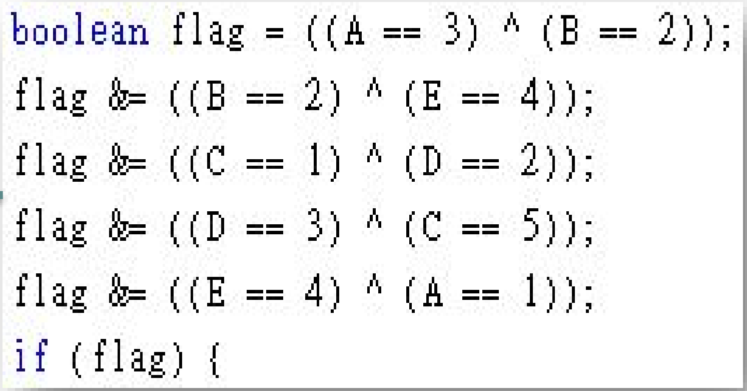
```
run:
3 1 5 2 4
1 answer(s) found.
BUILD SUCCESSFUL (total time: 0 seconds)
```

Example 3



```
if ((A == 3) ^ (B == 2)) {
    if ((B == 2) ^ (E == 4)) {
        if ((C == 1) ^ (D == 2)) {
            if ((D == 3) ^ (C == 5)) {
                if ((E == 4) ^ (A == 1)) {
```

- Don’t want the long conditional statement?
  - Use nested if
  - Use a temporary boolean variable
  - Use a failure count



```
boolean flag = ((A == 3) ^ (B == 2));
flag &= ((B == 2) ^ (E == 4));
flag &= ((C == 1) ^ (D == 2));
flag &= ((D == 3) ^ (C == 5));
flag &= ((E == 4) ^ (A == 1));
if (flag) {
```

```
int failCount = 0;
if (! ((A == 3) ∧ (B == 2))) failCount++;
if (! ((B == 2) ∧ (E == 4))) failCount++;
if (! ((C == 1) ∧ (D == 2))) failCount++;
if (! ((D == 3) ∧ (C == 5))) failCount++;
if (! ((E == 4) ∧ (A == 1))) failCount++;
if (failCount > 0) { … }
```

- Can you rewrite the code using while loop?

Classwork

- After a nuclear power station accident, the authorities received four different accounts / reports of the explosion order:
  - Reporter A: “Plant 3 was the second and Plant 1 was the third.”
  - Witness B: “Plant 1 was the second and Plant 3 was the fourth.”
  - Witness C: “Plant 2 was the second and Plant 4 was the third.”
  - Rescuer D: “Plant 3 was the fourth and Plant 2 was the first.”
- However, everyone had only got one statement correct (and thus the other statement was wrong)

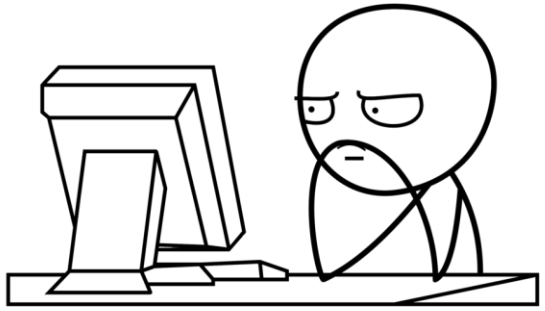
A past-paper problem!

Classwork

- Write a Java program to determine the full picture of the accident
- Your program should exhaustively try all the ordering combinations and output like this:

```
Explosion order of Plant 1: 3
Explosion order of Plant 2: …
…
```

- There is only one correct answer



Assignment 5

- Aims:
  - To compute some technical indicators that are/were commonly used for stock trading.
  - Practise the use of arrays.
  - Practise exception handling in Java.
- Background
  - **Technical Analysis for Stock Trading.**Technical analysis is a method to predict the direction of stock prices by studying the historical market data such as the stock prices and trading volumes. Simple Moving Average (SMA) and Bollinger Bands are some of the popular technical indicators that are/were used for stock trading.

Assignment 5

- Background
  - **Simple Moving Average (SMA).**A simple moving average (SMA) is calculated by averaging the prices of a stock over a specific time period. Given M a positive integer, a M-day SMA is the M-day sum of closing stock prices divided by M. As it is a moving average, data older than M days are dropped once new data become available. The computed average moves along the time axis. Below is an example showing the computation of a 9-day SMA.

Example:

Daily Closing Prices: 100, 102, 104, 106, 108, 110, 112, 114, 116, 118, 120, 122, 124, 126, 128

The first day of 9-day SMA: (100 + 102 + 104 + 106 + 108 + 110 + 112 + 114 + 116) / 9 = 108

The second day of 9-day SMA: (102 + 104 + 106 + 108 + 110 + 112 + 114 + 116 + 118) / 9 = 110

The third day of 9-day SMA: (104 + 106 + 108 + 110 + 112 + 114 + 116 + 118 + 120) / 9 = 112

Assignment 5

- Background
  - **Bollinger Bands**. Bollinger Bands were developed by John Bollinger. They represent the volatility of the prices of a particular stock. The estimation of the volatility is based on the computation of standard deviation. The higher the volatility, the wider is the resulting Bollinger Band. Bollinger Bands are made up of 3 bands:

    Middle Band = M-day simple moving average (SMA)

    Upper Band = [M-day SMA] + ([M-day standard deviation of price] x D)

    Lower Band = [M-day SMA] – ([M-day standard deviation of price] x D)

Assignment 5

- Input

In this assignment, you are required to write some classes to compute the simple moving average (SMA) and Bollinger Bands of stock prices. The stock prices are input to your program in the form of a Comma-Separated Values (**.csv**) file. The file format is the same as that of the historical stock quotes exported from yahoo finance (https://finance.yahoo.com). You need to refer to the sample files provided for the actual input requirements. It is assumed that stock prices are available on each stock trading day.

It is mandatory for you to create a new class named StockData to read the prices from the .csv file. This class should have a data field to store the prices of a particular stock over a period of time. You can safely assume that the maximum number of closing price ticks recorded in the input file is **5000**. The object created from class StockData will be used for the computation of technical indicators afterwards. **Input sample can be seen from Asg5 spec.**

Assignment 5

- Computing SMA and the Bollinger Bands

  To compute SMA and the Bollinger Bands, you are only required to use the **CLOSING PRICES (fifth column)**in the files. You need to define two classes named SMA and BBands to compute the SMA and the Bollinger Bands of a particular stock. **M and D are variables in the calculation of SMA and Bollinger Bands**. You need to read these values from the standard input by users.

  You are required get the stock data from the object created from class **StockData**. You can pass this object to the instances of **SMA** and **BBands** via their constructors or any other methods.

Assignment 5

- Output

  The computed SMA and Bollinger Bands are saved in a new file also in .csv format. The data starts from the first row. Each row records the resulting data of a single day. For example, if the input file contains 100 days of stock closing prices and M equals 20, there should be **82** rows in the output file **with a title in the first row**. See output sample in Asg5 spec.

  The first column of the .csv file stores the values of the SMA. The second and the third column store the upper and lower band of the Bollinger Bands, respectively. You can write the results to the .csv file within the SMA and BBands classes.

Assignment 5

- Exception Handling

  You are required to handle exceptions occurred in this assignment. You need to catch these exceptions in some appropriate places in your program code. If there are errors related to file reading, a message “**File Reading Errors**” should be displayed in the standard output. If errors related to writing the output file occur, a message **“File Writing Errors**” should be printed in the standard output. For both cases, your program should terminate normally.

  You can assume that values of M and D entered by the users are valid.

- More details are available in Asg5 spec.

END