CS 162FZ: Introduction to Computer Science II

Lecture 1

Introduction & Revision

Dr. Chun-Yang Zhang

What have we learned in CS161FZ?

- Computer architectures, software and hardware
- communication protocol
- numbering systems
- Basic programing skills in Java.

Module Overview

- Theoretical Computer Science; Languages: regular expressions, context-free grammars, recursive definitions; Automata: finite automata (deterministic and non-deterministic), conversion from grammars to automata, transition graphs, push-down automata;
- problem-solving: understanding and developing algorithms; Implementing algorithms to apply language definitions and to simulate automaton:
- iteration, nested iteration, arrays, multidimensional arrays, scope of variables, static methods, defining classes, creating objects, defining and calling instance methods and constructors, parament passing recursion, simple searching and sorting algorithms.

Learning Outcomes - On successful completion of the module, you will be able to:

- acknowledge the links between mathematics, science and computer science & appreciate the historical context in which the theory of computer science was developed;
- understand and describe the concept of finite state machines and regular languages;
- write simple programmes in java using conditional statements, loops, arrays and methods;
- apply problem solving techniques to programming problems;
- choose appropriate representations for problems;
- combine programming constructs to implement simple algorithms;
- compare good and bad implementations and algorithms.

CS162FZ Labs

- There are labs for practice, students can practice through Java Virtual Machine (JVM)
- Eclipse is an integrated development environment (IDE) that can be used to run java programs. To install it you will need to:
- Download the Java Software Development Kit (sdk) and Eclipse from CS161FZ.

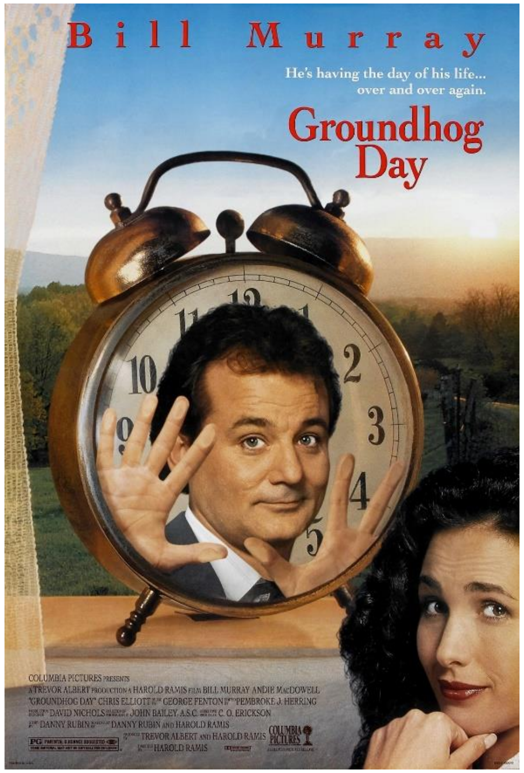
Assessment and Marking

- Your CS162 module mark will be calculated based on:
  - **Written exam (June exam) 70%**
  - **Continuous Assessment (CA) 30%**
    - **2 x lab exams worth 20% in total**
    - **Weekly Lab assignments worth 10% in total.**

Course Textbook

- Follow the text book: “Introduction to Computer Science I & II” by Aidan Mooney Susan Bergin
- Reading of the relevent chapter before lectures is **compulsory** since each teaching session functions as *flip* classroom. Se we read the lecture content ***before*** class and we work through engage in activities *in* class to faciliate deeper learning.

Attendance

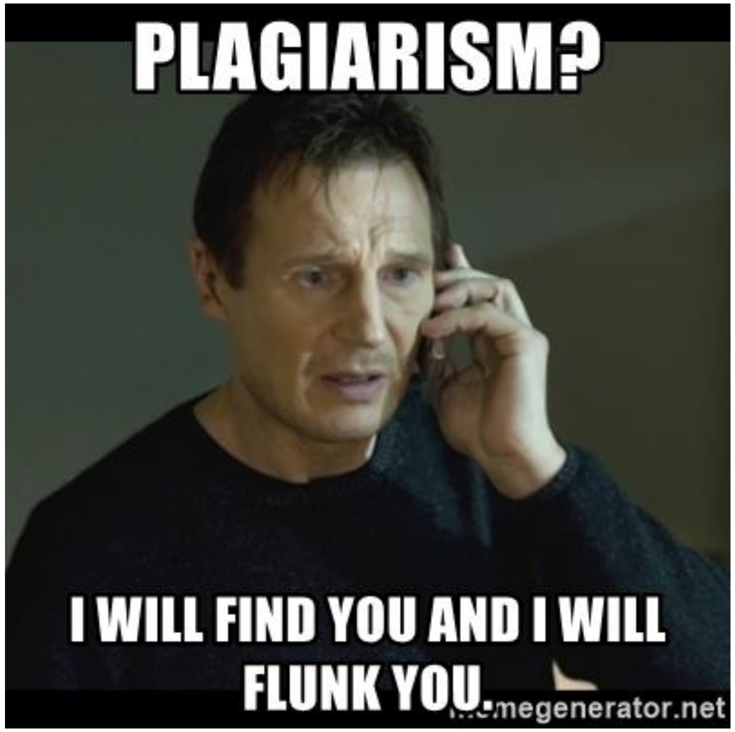


- High exam failure is directly correlated with low attendance.
- Do not drop anchor here!!!
- Same goes for Repeat students
- Take reponsibility for your own Learning!

Plagiarism – Zero Tolerance

[Avoiding Plagiarism- Library Video Guide]

[Plagiarism Policy]



Revision- CS161FZ Final Exam

We will review the CS161 briefly by looking at the final exam paper and answers:

- Basic knowledge of computer architecture, computer networks and number systems.

- Basic skills of Java programming to solve some problems.

Q1-Q2

1. In Java, the invalid statemen is (   ) **[2 marks]**

   A. String s= “join”+ “was”+ “here”;

   B. String s= “join”+3;

   C. int[] Arr = { 1, 1, 1, 1, 1 };

   D. Int[] Arr = new Arr[5]; Arr = { 1, 1, 1, 1, 1 };

   answer: D

2. When the following statements are implemented, the value of

   str is (   ) . **[2 marks]**

   String str ="123456789"; str = str.substring(1, 3);

   A. "23 " ;   B. "123" ;   C. "12" ;   D.

   "234" ;

   answer: A

Initialising an array – option 1

```
int[] myFirstArray = { 1, 1, 1, 1, 1 };
// declare and initialise together



int[] myFirstArray;
 myFirstArray = { 1, 1, 1, 1, 1 };
// Wrong case.



Int[] myFirstArray = new myFirstArray[5];
 myFirstArray = { 1, 1, 1, 1, 1 };
// Wrong case
```

String Methods

- You can concatenate two Strings using the + sign e.g

```
String s = new String(“hello”);
String s1 = new String(s + “ world”);
String s2 = new String(s + s1);
```

- Or you can use <String_1>.concat(<String_2>)

String Methods

- <String_1>.charAt(<int>) //see what char is at position <int>
- <String_1>.replace(<char>, <char>) // Replaces the first char specified with the second character specified
- <String_1>.indexOf(<char>) //Returns the index within this string of the first occurrence of the specified substring
- `<String_1>`.equals`(<String_2>)` // returns a boolean to check whether two strings are equal

String Methods

- To generate a substring of a String we specify the start and end position as ints - ends at end index –1.

  String_1>.substring(<int>, <int>)

- To generate a substring of a String (specify start position and always ends at end of String)

  <String_1>.substring(<int>)

Q3-Q4

3. The main() is defined as

   ```
   public static void main  (String[]   args ) {
    System.out.print( “Hello”+args[1] ) ;}
   ```

   When the arguments from command line are given by "people world nation", the result of the program is (   ) . **[2 marks]**

   A. Hello people

   B. Hello world

   C. Hello people world nation

   D. An exception occurred at runtime

   answer: B

4. In the following declarations, (   ) is invalid. **[2 marks]**

   A. float f=1/3; B. int i=1/3;

   C. float f=1*3.0; D. double f=1.0/3;

   answer: C

Command Line Parameters

- Sometimes we don’t know the value of a variable until the program runs, e.g. until a user provides input.
- Parameters can be passed to the main method and are stored in args[ ]
  - E.g. java test Hello World
- Within your program you can access the parameters in the args array e.g.
  - args[0] will reference Hello
  - args[1] will reference World

Data conversion

- Fractional number is given in default double type.
- Narrowing or explicit conversion cannot be automatically conducted.

double ➡ float ➡ long ➡ int ➡ short ➡ byte

- Type mismatch: cannot convert from double to float
- 1/3 could be assigned to int, float and double variables, because the numerator and denominator are both int.

Q5-Q7

5. If the follow program segment is implemented, (   ) will occur. **[2 marks]**

   ```
   String s = null; s.concat (" abc");
   ```

   A. ArithmeticException B. NullPointerException
   C. IOException D. EOFException

   answer: B

6. What is Von Neumann architecture? Briefly describe the functionalities of its four main components. **[5 marks]**

   Answer: Input and output devices, memory, CPU (ALU, CU, PC and registers).

7. What is operation system? Give three examples of operation systems. **[5 marks]**

   Answer: Operation system is a powerful, and usually large, program that controls and manages the hardware and other software on a computer. Eg. Linux, Windows, MacOS.

Exceptions

- Exceptions are runtime errors.
- An exception is an object that represents a type of error.

  ArithmeticException， NullPointerException， IOException，EOFException
- Unintended exceptions can cause program to terminate abnormally.
- Handling exceptions can make program more robust, but, on the other hand, it can potentially degrade the performance of theprogram.

Q8-Q9

8. Discuss the importance of using standard communication protocols for the Internet. **[5 marks]**

   Answer: A network protocol includes all the rules and conventions for communication between network devices, including ways devices can identify and make connections with each other. There are also formatting rules that specify how data is packaged into sent and received messages.

9. What is object-oriented programming, and explain the concepts: class, method and object. **[5 marks]**

   Answer: Older programming languages like COBOL and C followed the Procedural Programming approach. The program written using these languages used to be a series of step by step instructions. They made use of procedures/subroutines for making the program modular. This programming paradigm focused on logic more than data and the program used to combine both of them together. Modern programming languages like Java, C# etc. follow the Object Oriented approach. In object oriented programming, importance is given to data rather than just writing instructions to complete a task. A class defines attributes and behavior (methods). An object is a thing or idea that you want to model in your program.

Q10-Q12

10. Convert $(861.125)_{10}$ to binary, and convert $(11001001101)_2$ to hexadecimal. **[6 marks]**

    Answer:$(1101011101.001)_2$; $(64D)_{16}$

11. If declare and initialize four integer variables as a = 4, b= 8, c = 11, compute the expression: result = ++c%a/2 ==0 | b>a*2 & c <3*a; step by step. **[5 marks]**

    Answer: result = (c++)%a/2 ==0 | (b>a*2 & c <3*a)= ture.

12. What is the value in the variable *result* and *f* after the following code segment runs: **[5 marks]**

```
int result = 4;
boolean f = (++result < 4) && (result-- < 10);
```

    Answer: result = 4; f = false

A10

Decimal to binary

- Divide the integer part repeatedly by 2 until the quotient equals 0.
- Multiply the fractional part repeated by 2 until the remainder equals 0.

Binary to hexadecimal

- Group bits in fours, starting on right (you may need to pad out the left most group with 0’s)

A11: Operator Precedence

Precedence	Operator				Associativity
Highest	( )	[ ]	.		left to right
	++	--	!		right to left
	*	/	%		left to right
	+	-			left to right
	>	>=	<	<=	left to right
	==	!=			left to right
	&				left to right
	^				left to right
	|				left to right
	&&				left to right
	||				left to right
Lowest	=				right to left

When there are two operators with the same precedence the expression is evaluated according to its *associativity*.

A12: i++ and ++i

```
int i = 5;
int a = i++; // a = 5, i = 6
System.out.println(a);

int b = ++a; // b = 6, a = 6
System.out.println(b);
```

Increment goes first if ++ is in the left side.

Q13

13. What is the expected output following the execution of the code fragment given below? Please also identify the values for *result1*, *result2*, *result3* and *result4*. **[5 marks]**

```
public class StringComparision {

	public static void main(String[] args) {
		String str1 = "CS161FU";
		String str2 = new String("CS161FU");
		String str3 = "cs161FU";
		String str4 = "CS161FU";
		Boolean result1 = str1 ==str2;
		Boolean result2 = str1.equals(str2);
		Boolean result3 = str1.equals(str3);
		Boolean result4 = str1 ==str4;
		Boolean result = (result1 & result2)^(result3 |
result4);
		System.out.println(result);
	}
}
```

Answer: false;true;false;true;true

String methods (continued)

- Test if 2 Strings are lexicographically equal

```
<String_1>.equals(<String_2>)   // returns a boolean
<String_1>.compareTo(<String_2>)   // returns an int
```

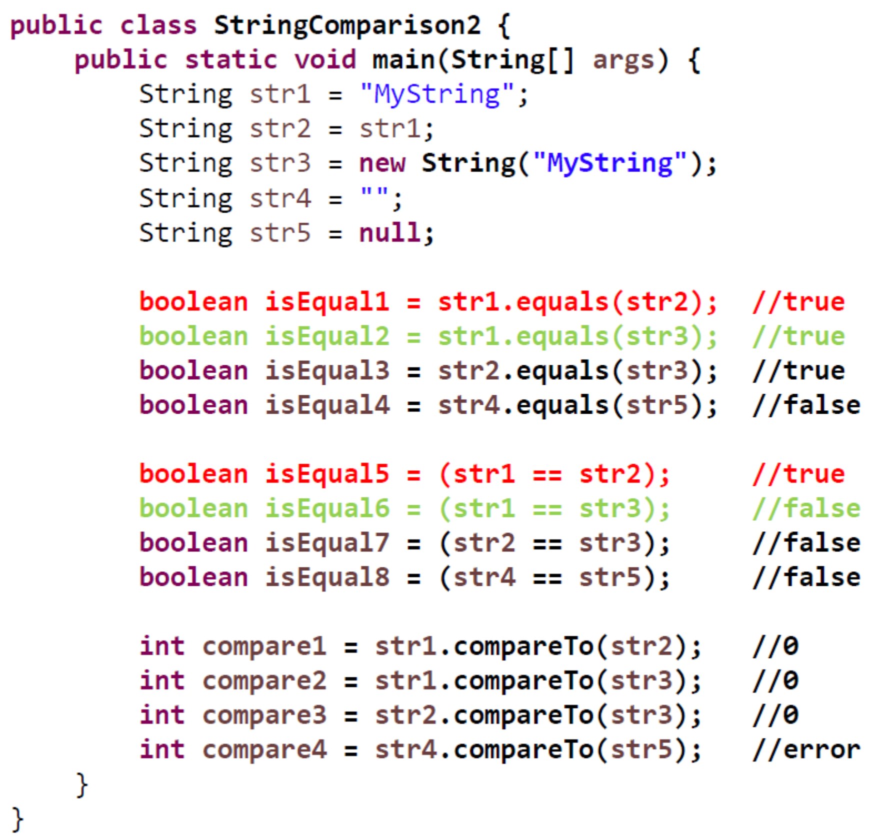
```
if(s1.equals(s2)==true)
   { …
   }

if(s1.compareTo(s2)==0)
   { …
   }
```

Returns a value of 0 if the two Strings are lexicographically equal Returns a value < 0 if String_2 comes lexicographically after String_1 OR a value > 0 if String_1 comes lexicographically after String_2

- Can also use `.equalsIgnoreCase()` and `.compareToIgnoreCase()`

String Comparison

- Remember Strings are objects and not primitive types.
- **You can’t compare Strings using <, >, <=, >=.**
- **When we compare Strings using ==, we are not comparing contents but references.**



```java
public class StringComparison2 {
    public static void main(String[] args) {
        String str1 = "MyString";
        String str2 = str1;
        String str3 = new String("MyString");
        String str4 = "";
        String str5 = null;

        boolean isEqual1 = str1.equals(str2);   //true
        boolean isEqual2 = str1.equals(str3);   //true
        boolean isEqual3 = str2.equals(str3);   //true
        boolean isEqual4 = str4.equals(str5);   //false

        boolean isEqual5 = (str1 == str2);      //true
        boolean isEqual6 = (str1 == str3);      //false
        boolean isEqual7 = (str2 == str3);      //false
        boolean isEqual8 = (str4 == str5);      //false

        int compare1 = str1.compareTo(str2);    //0
        int compare2 = str1.compareTo(str3);    //0
        int compare3 = str2.compareTo(str3);    //0
        int compare4 = str4.compareTo(str5);    //error
    }
}
```

Q14

14. Please find all errors in following program. **[6 marks]**

```
public class test {
public static void main(String[] args) {

    for ( count = 0; count >= 0; count++);
    {
    System.out.println("Count is: " + count );
    }

}
```

Answer: (1) there is no semicolon in the end of condition; (2) a } is missing for the class; (3) int count = 0; (4) infinite loop

Q15

15. Write a complete program to print all the Narcissistic numbers between 100 and 1000 exclusive. A three-digits number is call ed as Narcissi number if it equals the sum of the cubes of its digits. For example, $370 = 3^3 + 7^3 + 0^3$, and 370 is a Narcissi number.

Step (1): split each number properly; **[3 marks]**

Step (2): check if it is a Narcissi number; **[5 marks]**

Step (3) : print all the Narcissi numbers in single line with four blank spacing. **[3 marks]**

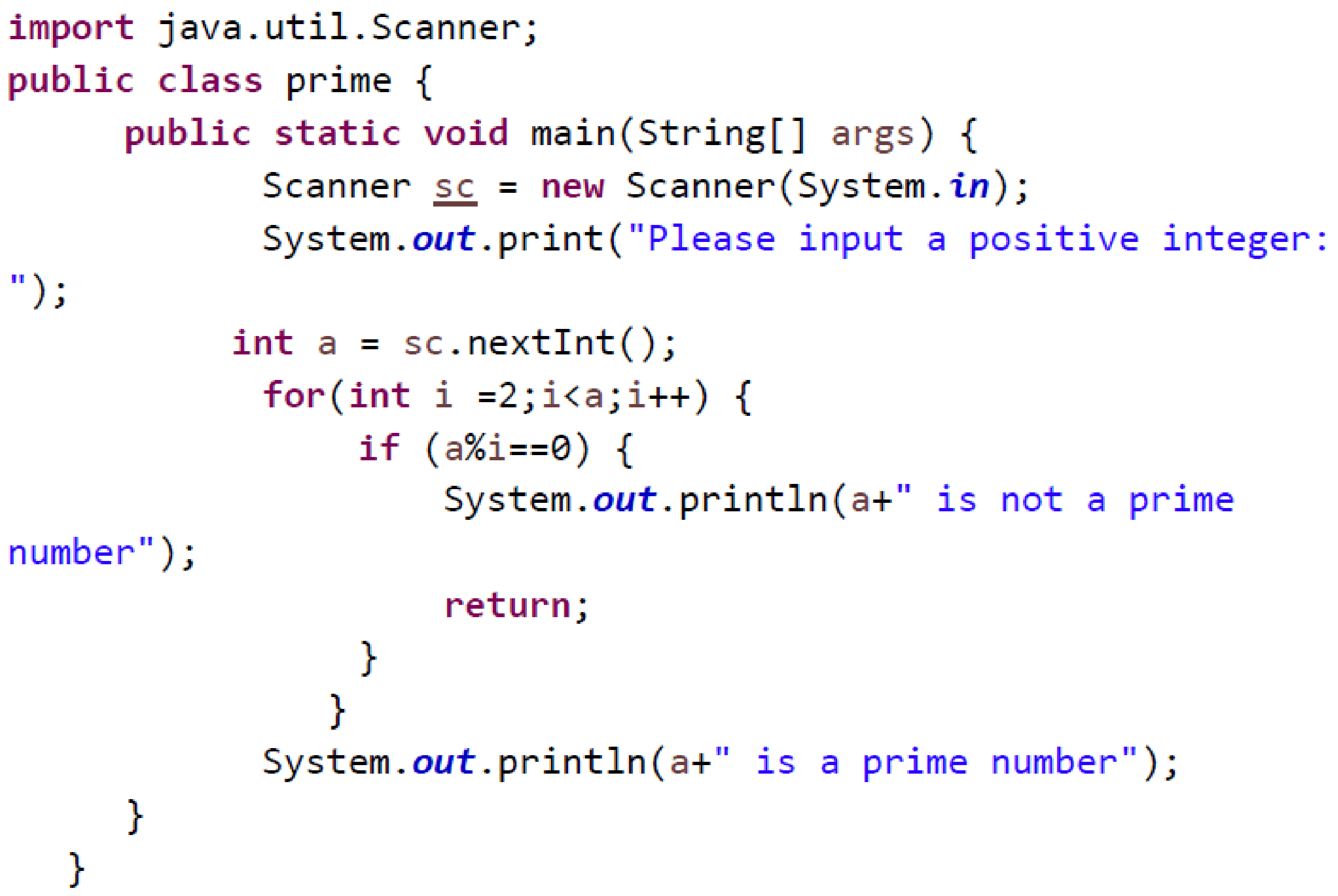
Answer:

```
public class Narcissistic {
     public static void main(String[] args) {
          int firstdigit = 0;
          int seconddigit = 0;
          int lastdigit = 0;
          for(int i = 100; i < 1000; i++){
            lastdigit = i/100;
            seconddigit = i/10%10;
            firstdigit = i%10;
            if(Math.pow(firstdigit, 3) + Math.pow(seconddigit,
3) + Math.pow(lastdigit, 3) == i ){
              System.out.print(i+"\t");
            }
          }
     }
   }
```

Q16

16. For any positive integer, write a complete program to check whether it is a prime number. Prime numbers are the positive integers having only two factors, 1 and the integer itself.
    1. Step 1: declare an integer variable, assign its value from keyboard with user prompt message “Please input a positive integer:”; **[4 marks]**
    2. Step 2: identify whether the integer is a prime number. **[5 marks]**
    3. Step 3: print appreciate messages. If the number is 5, print “5 is a prime number”. If the number is 4, print “4 is a prime number”. **[2 marks]**

A16



```
import java.util.Scanner;
public class prime {
	public static void main(String[] args) {
		Scanner sc = new Scanner(System.in);
		System.out.print("Please input a positive integer: 
");
	      int a = sc.nextInt();
	       for(int i =2;i<a;i++) {
	           if (a%i==0) {
	               System.out.println(a+" is not a prime 
number");
	               return;
	           }
	         }
		System.out.println(a+" is a prime number");
	}
    }
```

Q17

1. Write a complete program that creates two strings *s1* and *s2* reference “Hello World” and “hello world” respectively. The program should:
   (1) check if the Strings have the same length. Print an appropriate message; **[2 marks]**
   (2) If they are the same length check if the strings are lexicographically the same; **[2 marks]**
   (3) get the 6rd character in *s1*; **[2 marks]**
   (4) change the case of *s1* and *s2*, one to uppercase and one to lower case; **[2 marks]**
   (5) check if *s1* contains the letter ‘l’. If it does print a message to display the first position of the letter ‘l’ in the String. **[3 marks]**

A17

```
public class similar
{
   public static void main(String args[])
   {
      String s1 = new String("Hello world"), s2 = new
String("Hello World");
      int len1 = s1.length(), len2 = s2.length();
      if(len1 == len2){
         System.out.println("The Strings are both of length "
+ len1);
         if(s1.equals(s2)){
            System.out.println("The Strings are
lexicographically the same");
         }
         else{
            System.out.println("The Strings are not
lexicographically the same");
         }
      }
      else{
         System.out.println("The Strings are not the same
length");
      }
```

A17 continued

```
         char c1 = s1.charAt(2);
        s1 = s1.toUpperCase();
        s2 = s2.toLowerCase();
        System.out.println("S1: " + s1);
        System.out.println("S2: " + s2);

        if(s1.indexOf('l') != -1){
            System.out.println("The character e appears at index: 
" + s1.indexOf('l'));
        }
    }
}
```

Q18

1. Given a matrix $\begin{pmatrix} 8 & 1 & 2 & 2 & 9 \\ 1 & 9 & 4 & 0 & 3 \\ 0 & 3 & 0 & 0 & 7 \end{pmatrix}$, write a complete program to calculate the minimal and maximal elements, mean value and standard deviation, denoted by *min, max, mean and std* respectively. For a matrix a= $\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$, *mean* is the average value of all the elements in the matrix, and $std = \frac{1}{N-1}\sqrt[2]{\sum_{i=1}^{3}\sum_{j=1}^{3}(a_{ij} - mean)^2}$, where N is the number of elements.

   (1) Declare and initialize an 2D array with integer type; **[3 marks]**

   (2) Compute the *max, min*, *mean* and *std* with proper types; **[5 marks]**

Print the four values to the screen in different lines; **[2 marks]**

A18

```
public class matrix {
   public static void main(String[] args) {
   int [][] a =
{{8,1,2,2,9},{1,9,4,0,3},{0,3,0,0,7}};
   int r = a.length, c = a[0].length;
   double mean = 0, std = 0;
   int min = a[0][0], max = a[0][0];
   for(int i=0;i<r;i++)
        for(int j=0;j<c;j++) {
             min = min<a[i][j]? min:a[i][j];
             max = max>a[i][j]? max:a[i][j];
             mean+=a[i][j];
            }
   mean/=(r*c);

   for(int i=0;i<r;i++)
        for(int j=0;j<c;j++)
             std += Math.pow((a[i][j]-mean), 2);

          std = Math.pow(std, 0.5)/(r*c-1);
          System.out.println("Min:"+min);
          System.out.println("Max:"+max);
          System.out.println("mean:"+mean);
          System.out.println("std:"+std);
         }
 }
```