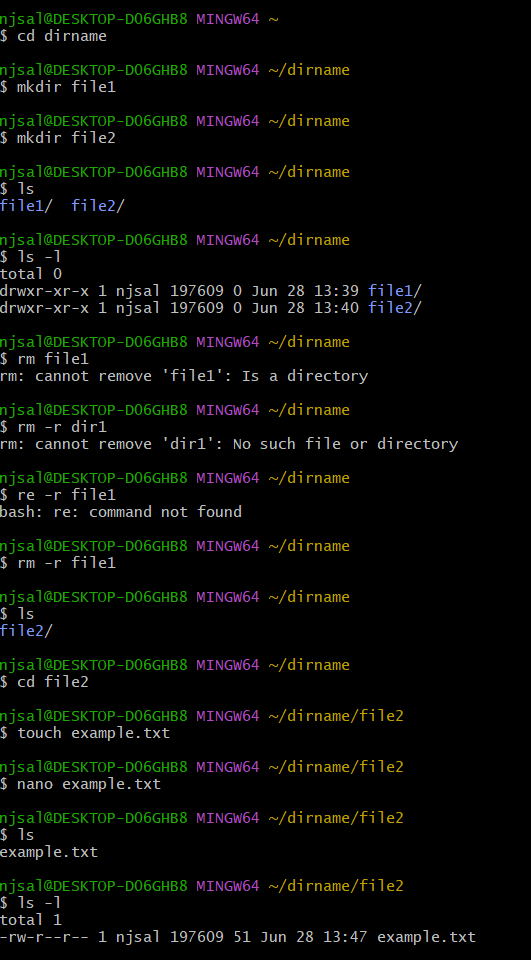
**Lab 1.1 Setting up your computer**

(1)



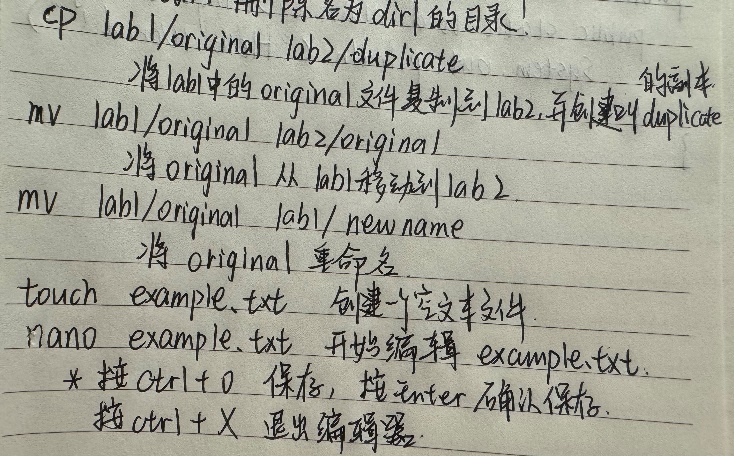
(2)



All finished in Git Bash

另：

pwd 显示当前目录、路径



**1.1 notes**



通过 javac HelloWorld.java 命令，Java 编译器将你的源代码 HelloWorld.java 编译成了这个 .class 文件(它很复杂)。当你运行 java HelloWorld 时，JVM 将加载并执行 HelloWorld.class 文件中的字节码指令，最终输出 "Hello world!"。

编译 Java 程序时我们需要包括 .java 扩展名，但在解释（运行）时却不需要包括 .class 扩展名。

类名不能以数字开头。

Exercise 1

Modify HelloNumbers so that it prints out the cumulative sum of the integers from 0 to 9. For example, your output should start with 0 1 3 6 10... and should end with 45.

public class HelloNumbers{

public static void main(String[] args){

int x=0;

int s=0;

while (x<10){

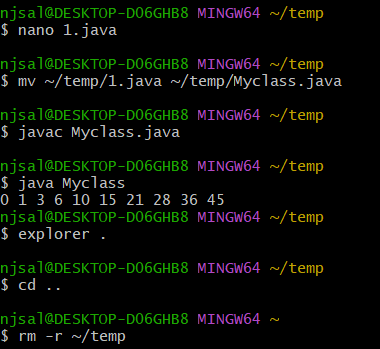
s=s+x;

System.out.print(s+” “);

x=x+1;}

}

}



**HW0**Creative Exercise 1a: Drawing a Triangle

Your goal is to create a program that prints the following figure. Your code should use loops (i.e. shouldn’t just be five print statements, that’s no fun).

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

在Java中，类的名称通常应该与文件名保持一致，尤其是对于公共类（public class）来说。这是Java编译器的要求之一，如果类的名称与其所在的文件名不一致（特别是对于公共类），编译器会发出警告或错误。

public class HelloNumbers{

    public static void main(String[] args){

        int row=1;

        int col=1;

        System.out.print("\n");

        while(row<=5){

            while (col<=row){

                System.out.print("\*"+" ");

                col=col+1;

                if(col>row){

                    col=1;

                    System.out.print("\n");

                    break;

                }

            }

            row=row+1;

        }

    }

}

优化：

public class HelloNumbers {

    public static void main(String[] args) {

        int rows = 5;

        for (int i = 1; i <= rows; i++) {

            for (int j = 1; j <= i; j++) {

                System.out.print("\*"+" ");

            }

            System.out.println(); // 每行结束后换行

        }

    }

}

Each defines a function that returns the maximum of two values and then prints the maximum of 5 and 15.

public static int max(int x, int y) {

if (x > y) {

return x;

}

return y;

}

public static void main(String[] args) {

System.out.println(max(10, 15));

}

## Creative Exercise 1b: DrawTriangle

Name this new method drawTriangle and give it a return type of void (this means that it doesn’t return anything at all).

The drawTriangle method should take one parameter named N, and it should print out a triangle exactly like your triangle from exercise 1a, but N asterisks tall instead of 5.

After writing DrawTriangle, modify the main function so that it calls DrawTriangle with N = 10.

public class HelloNumbers {

    public static void main(String[] args) {

        drawTriangle(10);

    }

    public static void drawTriangle(int N) {

        for (int i = 1; i <= N; i++) {

            for (int j = 1; j <= i; j++) {

                System.out.print("\*"+" ");

            }

            System.out.println();

        }

    }

}

\*数组：  
**Java**

int[] numbers = new int[3];

numbers[0] = 4;

numbers[1] = 7;

numbers[2] = 10;

System.out.println(numbers[1]);

Or in an alternate (but less general) shorthand:  
**Alternate Java**

int[] numbers = new int[]{4, 7, 10};

System.out.println(numbers[1]);

You can get the length of an array by using .length, for example, the following code would print 3:

int[] numbers = new int[]{4, 7, 10};

System.out.println(numbers.length);

**Exercise 2**

Using everything you’ve learned so far on this homework, you’ll now create a function with the signature public static int max(int[] m) that returns the maximum value of an int array. You may assume that all of the numbers are greater than or equal to zero.

public class HelloNumbers {

    public static int max(int[] m) {

        int max=m[0];

        int i=1;

        for (i=1;i<m.length;i++){

            if (max<=m[i]){

                max = m[i];

            }

        }

        return max;

    }

    public static void main(String[] args) {

       int[] numbers = new int[]{9, 2, 15, 2, 22, 10, 6};

       System.out.println(max(numbers));

    }

}

\*对比（别的都一样）：

String[] a = {"cat", "dog", "laser horse", "ketchup", "horse"};

Optional: Exercise 4

Write a function windowPosSum(int[] a, int n) that replaces each element a[i] with the sum of a[i] through a[i + n], but only if a[i] is positive valued. If there are not enough values because we reach the end of the array, we sum only as many values as we have.

For example, suppose we call windowPosSum with the array a = {1, 2, -3, 4, 5, 4}, and n = 3. In this case, we’d:

* Replace a[0] with a[0] + a[1] + a[2] + a[3].
* Replace a[1] with a[1] + a[2] + a[3] + a[4].
* Not do anything to a[2] because it’s negative.
* Replace a[3] with a[3] + a[4] + a[5].
* Replace a[4] with a[4] + a[5].
* Not do anything with a[5] because there are no values after a[5].

Thus, the result after calling windowPosSum would be {4, 8, -3, 13, 9, 4}.

As another example, if we called windowPosSum with the array a = {1, -1, -1, 10, 5, -1}, and n = 2, we’d get {-1, -1, -1, 14, 4, -1}.

Hint 1: Use two for loops.

Hint 2: Use continue to skip negative values.

Hint 3: Use break to avoid going over the end of the array.

public class HelloNumbers {

    public static void windowPosSum(int[] a, int n) {

        int i=0;

        int j=1;

        for(i=0;i<a.length;i++){

            if(a[i]<0){

                continue;

            }

            if(a[i]>=0){

                if(n<(a.length-i)){

                    for(j=1;j<=n;j++){

                        a[i]=a[i]+a[i+j];

                    }

                }

                else{

                    for(j=1;j<(a.length-i);j++){

                        a[i]=a[i]+a[i+j];

                    }

                }

            }

        }

    }

      public static void main(String[] args) {

        int[] a = {1, 2, -3, 4, 5, 4};

        int n = 3;

        windowPosSum(a, n);

        // Should print 4, 8, -3, 13, 9, 4

        System.out.println(java.util.Arrays.toString(a));

//将数组 a 转换为字符串

      }

}

\*另一种写法：

However, in this case, we do not create an index i. Instead, the String s takes on the identity of each String in a exactly once, starting from a[0], all the way up to a[a.length - 1].

public class EnhancedForBreakDemo {

public static void main(String[] args) {

String[] a = {"cat", "dog", "laser horse", "ketchup", "horse", "horbse"};

for (String s : a) {

for (int j = 0; j < 3; j += 1) {

System.out.println(s);

if (s.contains("horse")) {

break;

}

}

}

}

}

**1.2 notes**

public class Dog {

public static void makeNoise() {

System.out.println("Bark!");

}

}

public class DogLauncher {

public static void main(String[] args) {

Dog.makeNoise();

}

}

以上为调用方法：

$ java DogLauncher

Bark!

A class that uses another class is sometimes called a "client" of that class, i.e. DogLauncher is a client of Dog.

public class DogLauncher {

public static void main(String[] args) {

Dog d;

d = new Dog();

d.weightInPounds = 20;

d.makeNoise();

}

}

也可以写成：

Dog d = new Dog(20);

\*

我们也可以将 maxDog 实现为一个非静态方法，例如：

public Dog maxDog(Dog d2) {

if (this.weightInPounds > d2.weightInPounds) {

return this;

}

return d2;

}

在上面的例子中，我们使用关键字 this 来引用当前对象。这个方法可以像下面这样调用：

Dog d = new Dog(15);

Dog d2 = new Dog(100);

d.maxDog(d2);

\*👆静态方法与非静态方法的区分

public class ArgsDemo {

public static void main(String[] args) {

System.out.println(args[0]);

}

}

This program prints out the 0th command line argument, e.g.

$ java ArgsDemo these are command line arguments

these

In the example above, args will be an array of Strings, where the entries are {"these", "are", "command", "line", "arguments"}.

**Exercise 1.2.3**: Try to write a program that sums up the command line arguments, assuming they are numbers. For a solution, see the webcast or the code provided on GitHub.

public class HelloNumbers {

    public static void main(String[] args) {

        if (args.length == 0) {

            System.out.println("Usage: java SumCommandLineArguments <number1> <number2> ... <numberN>");

            return;

        }

        double sum = 0.0;

        try {

            for (String arg : args) {

                double num = Double.parseDouble(arg);

                sum += num;

            }

            System.out.println("Sum of command line arguments: " + sum);

        } catch (NumberFormatException e) {

            System.err.println("Error: One of the command line arguments is not a valid number.");

        }

    }

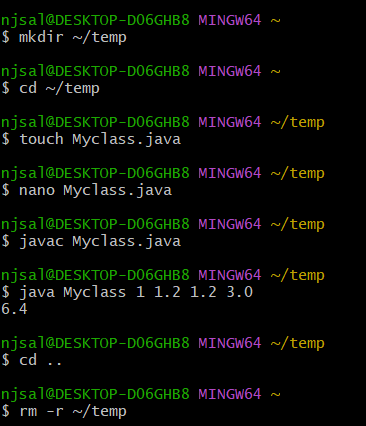
}

编译这个Java程序：javac SumCommandLineArguments.java

运行程序并提供数字作为命令行参数：java SumCommandLineArguments 1.5 2 3.7

例如，如果运行 java SumCommandLineArguments 1.5 2 3.7，程序将计算并输出这些数字的总和：

Sum of command line arguments: 7.2



在命令行（cmd）中，当你执行一个Java程序并传递参数时，这些参数会被作为字符串数组传递给Java程序的 main 方法。这个字符串数组被称为 args，它是程序的命令行参数数组。

1、将字符串转换为数值类型：使用适当的方法（如 Double.parseDouble() 或 Integer.parseInt()）将字符串参数转换为 double 或 int。这样可以进行数值计算。

2、进行数值计算：将转换后的数值相加、减去或进行其他数学操作，具体取决于你的程序逻辑。

3、处理异常：由于用户可以输入无效的字符串，如非数值字符串，因此在转换过程中可能会抛出 NumberFormatException 异常。需要使用 try-catch 块来捕获并处理这些异常。

**Discussion**

递归函数、函数的嵌套

Implement fib which takes in an integer n and returns the nth Fibonacci number.The Fibonacci sequence is 0,1,1,2,3,5,8,13,21,....

法1：

public static int fib(int n) {

if (n <= 1) {

return n;}

else {

return fib(n - 1) + fib(n - 2); } }

法2：

public static int fib2(int n, int k, int f0, int f1) {

if (n == k) {

return f0;

}

else{

return fib2(n, k+1, f1, f0+f1);

}

**Lab 1: javac, java, git**

我的仓库（本地）：C:\Users\njsal\CS61B

使用以下命令将其添加到Git跟踪列表中（添加到暂存区）：

git add 61b.txt

然后，创建一个新的提交，并包含这个文件：

git commit -m "Add 61b.txt"

**创建一个新的提交**，包含升级后的 61b.txt 和 61c.txt：

git commit -m "Update 61b.txt and add 61c.txt"

\*查看当前Git状态：

git status

这个命令将显示当前的Git仓库状态，包括已修改但未暂存和已暂存但未提交的文件。

\*查看提交记录：

git log

这个命令将显示所有的提交记录，包括每个提交的作者、提交日期和提交信息。

git push origin master 全部提交

In the lab1 folder, you should see a file called LeapYear.java. This program is supposed to test whether or not a given year is a Leap Year. The user will give a year as a command line parameter (examples given below), and then print out whether or not that year is a leap year, e.g.

$ java LeapYear 2000

2000 is a leap year.

$ java LeapYear 1999

1999 is not a leap year.

$ java LeapYear 2004

2004 is a leap year.

$ java LeapYear 2100

2100 is not a leap year.

A leap year is either:

divisible by 400 or

divisible by 4 and not by 100.

public class LeapYear {

    public static boolean isLeapYear(int year){

        if (year % 400 ==0 || ((year%4==0) && (year%100 != 0))){

            return true;}

        else {

            return false;}

    }

    /\*\* Calls isLeapYear to print correct statement.

     \*  @param  year to be analyzed

     \*/

    private static void checkLeapYear(int year) {

        if (isLeapYear(year)) {

            System.out.printf("%d is a leap year.\n", year);

        } else {

            System.out.printf("%d is not a leap year.\n", year);

        }

    }

    /\*\* Must be provided an integer as a command line argument ARGS. \*/

    public static void main(String[] args) {

        if (args.length < 1) {

            System.out.println("Please enter command line arguments.");

            System.out.println("e.g. java Year 2000");

        }

        for (int i = 0; i < args.length; i++) {

            try {

                int year = Integer.parseInt(args[i]);

                checkLeapYear(year);

            } catch (NumberFormatException e) {

                System.out.printf("%s is not a valid number.\n",args[i]);

            }

        }

    }

}