# 读取输入

Scanner in = new Scanner(System.in);

输入一行（包含空格）

String str = in.nextLine()

读取一个单词（以空白符作为分隔符）

String firstName = in.next();

读取一个整数

int age = in.nextInt();

读取一个浮点数 in.nextDouble()

输入是可见的，Scanner类不适合从控制台读取密码，Java 6引入Console实现

Console cons = System.console();

char[] passwd = cons.readPassword(“Password:”);

为了安全起见，返回的密码存放在一维字符数组中，而不是字符串中。在对密码进行处理之后，应马上用一个填充值覆盖数组元素。

采用console输入一次只能读取一行。

## Java.util.Scanner 5.0

Scanner(InputStream in)

用给定的输入流创建一个Scanner对象

String nextLine()

读取输入的下一行内容

String next()

读取输入的下一个单词(以空格作为分隔符)

int nextInt()

double nextDouble()

读取并转换下一个表示整数或浮点数的字符序列

boolean hasNext()

检测输入中是否还有其他单词

boolean hasNextInt()

boolean hasNextDouble()

检测是否还有表示整数或浮点数的下一个字符序列

## java.lang.System 1.0

static Console console() 6

如果有可能进行交互操作，就通过控制台窗口为交互的用户返回一个Console对象。否则返回null。对于任何一个通过控制台窗口启动的程序，都可以使用Console对象。否则，其可用性将与使用的系统有关。

## java.io.Console 6

static char[] readPassword(String prompt,Object…args)

static String readLine(String prompt,Object…args)

显示字符串primpt并且读取用户输入，知道输入行结束。args参数可以用来提供输入格式。

# 格式化输出

System.out.printf()

每个以%字符开始的格式说明符都用相应的参数替换。格式说明符尾部的转换符将指示被格式化的数值类型：f表示浮点数，d表示是十进制整数。

用于printf的转换符：



用于printg的标志



+ 打印正数和负数的符号 +3333.33

空格 在正数之前添加空格 | 3333.33|

System.out.printf(“%,(.2f”) 使用分组的分隔符并将负数括在括号内

可以使用s转换符格式化任意的对象。对于任意实现了Formattable 接口的对象都将调用formatTo方法；否则将调用toString方法，它可以将对象转换为字符串。

可以使用静态的String.format方法创建一个格式化的字符串，而不打印输出：

String message = String.format(“Hello,%s. Next Year，you’ll be %d”,name,age);

时间与日期的转换符





可以采用一个格式化的字符串支出要被格式化的参数索引，索引必须紧跟在%后面，并以$终止，例如：

System.out.printf(“%1$s %2$tB %2$te,%2$tY,”date:”,new Date());

date:February 9, 2004

还可以选择使用<标志，指示前面格式说明中的参数将被再次使用。

System.out.printf(“%s %tB %<te %<tY”,”date:”,new Date());

参数索引从1开始



# 文件输入与输出

读取，用File对象构造一个Scanner对象

Scanner in = new Scanner(Path.get(“myfile.txt”));

如果文件名中包含反斜杠符号，就要记住在每个反斜杠之前再加一个额外的反斜杠：“c:\\mydirectory\\myfile.txt”

然后就可以使用任何一个Scanner方法对文件进行读取

写入 需要构造PrintWriter对象。在构造器中，只需要提供文件名

PrintWriter out = new PrintWriter(“myfile.txt”);

如果文件不存在，创建该文件。可以像System.out一样使用print，println，printf

如果用一个不存在的文件构造一个Scanner，或者用一个不能被创建的文件名构造一个PrintWriter，那么会发生异常。

Public static void main(String[] args) **throw FileNotFoundException**

{

Scanner in = new Scanner(Path.get(“myfile.txt”));

…

}

java.util.Scanner 5.0

Scanner(File f)

构造一个从给定文件读取数据的Scanner

Scanner(String data)

构造一个从给定字符串读取数据的Scanner

java.io.PointWriter 1.1

PrintWriter(String filename)

构造一个将数据写入文件的PrintWriter.

Java.nio.file.Paths 7

static Path get(String pathname)

根据给定的路径名构造一个Path