JVM, Java Virtual Machine (Java虚拟机)
JDK, Java Development Kit (Java开发工具包)
JRE, Java Runtime Environment (Java运行环境)

Java.exe 解释器|J2SE Java 嵌入式应用开发平台
标识符不能是true, false, null（尽管true, false, null不是关键字）

```
Scanner reader=new Scanner(System.in);
double sum=0;
int m=0;
while(reader.hasNextDouble())
{
    double x=reader.nextDouble();
    m=m+1;
    sum=sum+x;
}
```

Java不支持多重继承，即一个类只能继承一个父类。
为了克服单继承的缺点，Java使用了**接口**，一个类可以**实现**多个接口。逗号隔开

StringBuffer是线程安全的，StringBuilder不是线程安全的

- **线程安全是指多个线程操作同一个对象不会出现问题。**

如果字符串缓冲区被**单个线程**使用（这种情况很普遍），建议优先采用**StringBuilder**，因为效率高（而**线程同步**需要时间开销）

```
String cost = "市话费：176.89元，长途费：187.98
Scanner scanner = new Scanner(cost);
scanner.useDelimiter("[^0123456789.]+");
while(scanner.hasNext())
{
    try{
        double price = scanner.nextDouble();
        System.out.println(price);
```

```
Pattern p;
Matcher m;
String input = "9A00A3";
p = Pattern.compile("\\dA\\d");
m = p.matcher(input);
while(m.find())
{
    String str = m.group();
```

.	.	代表任何一个字符
\d	\\d	代表0~9的任何一个数字
\D	\\D	代表任何一个非数字字符
\s	\\s	代表空格类字符，'\t', '\n', '\x0B', '\f', '\r'
\S	\\S	代表非空格类字符
\w	\\w	代表可用于标识符的字符（不包括美元符号）
\W	\\W	代表不能用于标识符的字符

X?	X出现0次或1次
X*	X出现0次或多次
X+	X出现1次或多次
X{n}	X恰好出现n次
X{n,}	X至少出现n次
X{n, m}	X出现n次至m次

y, yy：2位数字年份，如14
yyyy：4位数字年份，如2014
M, MM：2位数字月份，如08
MMM：汉字月份，如八月
d, dd：2位数字日期，如09, 22
a：上午或下午
H, HH：2位数字小时（00-23）
h, hh：2位数字小时（am/pm，01-12）
m, mm：2位数字分
s, ss：2位数字秒
E, EE：星期

```
String pattern = "yyyy-MM-dd";
SimpleDateFormat sdf = new SimpleDateFormat(pattern);
Date currentTime = new Date();
String currenTime2 = sdf.format(currentTime);
```

```
File file = new File("hello.java");
Scanner scanner = new Scanner(file);
scanner.useDelimiter(正则表达式);
```

在线程的run()方法结束之前，即没有进入死亡状态之前，线程调用isAlive()方法返回**true**。当线程进入死亡状态后（实体内存被释放），线程仍可以调用方法isAlive()，这时返回的值是**false**。线程未调用start()方法之前，调用isAlive()方法返回**false**。

```
class Bank implements Runnable{
    int money = 300;
    String treasurerName, cashierName;
    public Bank(String s1,String s2){ treasurerName=s1; cashierName=s2; }
    public void run(){ saveOrTake(30); }
    public synchronized void saveOrTake(int number){
        if(Thread.currentThread().getName().equals(treasurerName)){
            for(int i=1;i<=3;i++){
                money = money + number;
                try { Thread.sleep(1000); }
                catch(InterruptedException e) {}
                System.out.println(treasurerName + " : " + money);
            }
```

```
try{
    FileReader fr = new FileReader("input.txt");
    BufferedReader input = new BufferedReader(fr);
    FileWriter fw = new FileWriter("output.txt");
    BufferedWriter output = new BufferedWriter(fw);

    String s=null;
    int i=0;
    while((s = input.readLine())!=null){
        i++;
        output.write(i + ": " + s);
        output.newLine();
    }
    output.flush(); output.close(); fw.close();
    input.close(); fr.close();
```

```
public void actionPerformed(ActionEvent e)
{
   JTextField textSource = (JTextField)e.getSource();
   if(textSource==titleText)
   {
      this.setTitle(titleText.getText());
   }
```

发送:

```
byte b[]=outMessage.getText().trim().getBytes();
try
{
    InetAddress address = InetAddress.getByName("127.0.0.1");
    DatagramPacket data = new DatagramPacket(b,b.length,address,5678);
    DatagramSocket mail = new DatagramSocket();
→   mail.send(data);
```

接收:

```
DatagramPacket pack = null;
DatagramSocket mail = null;
byte b[]=new byte[8192];
try
{
   pack = new DatagramPacket(b,b.length);
   mail = new DatagramSocket(1234);
}
catch(Exception e){}

while(true)
{
    try
    {
→       mail.receive(pack);
        String message=new String(pack.getData(),0,pack.getLength());
        inMessage.append("收到数据来自："+pack.getAddress());
        inMessage.append("\n收到数据是："+message+"\n");
        inMessage.setCaretPosition(inMessage.getText().length());
    }
```

```
    try {
            ServerSocket serverSocket = new ServerSocket(8080);
            Socket socket = serverSocket.accept();
            InputStream is = socket.getInputStream();
            BufferedReader br = new BufferedReader(new InputStreamReader(is));
            OutputStream os = socket.getOutputStream();
            PrintWriter pw = new PrintWriter(os);

            String info=null;
while(!((info=br.readLine())==null)){
    System.out.println("收到请求："+info);
}
String reply=getQuestions();
pw.write(reply);
pw.flush();
pw.close();
os.close();
br.close();
is.close();
socket.close();
serverSocket.close();
```

1.super 和 this 关键字：在子类构造方法中使用 super（）显示调用父类的构造方法，super（）必须写在子类构造方法的第一行，否则编译不通过；）this（）和 super（）不可以同时出现在一个构造函数中；this（）和 super（）不可以在 static 环境中使用，无论是 static 方法和 static 语句块。

2.java 继承只允许继承一个父类，可以实现多个接口。

3. hasNext 不是 Object 的方法。

4.多态经过继承体现。

5.抽象类没有构造方法，且必须经过继承才能实例化，子类需要实现父类的所有抽象方法。

6.final 定义的变量只允许赋值一次，且必须在定义时直接赋值。

7.在 try-catch-finally 语句块中，try 可以单独与 finally 一起使用。

9.switch 可以作用在 string byte,long 不可以（实际支持 byte 和 long）

10.一个".java"源文件中可以包含多个类（不是内部类），但主类必须和文件名一样，如果右 public，必须是同名的，且 public 类不能超过一个。

	wait	sleep
同步	必须在同步上下文调用，否则抛出异常	不需要在同步方法或同步块中调用
作用对象	定义在Object类中，作用于对象本身	定义在java.lang.Thread中，作用于当前线程
释放锁资源	同步方法中，想调用wait(),前提是必须获取对象上的锁资源	如果sleep()在同步上下文中调用，则其他线程无法进入当前同步块/方法
唤醒条件	其它线程调用notify()或notifyAll()	超过或者调用interrupt()
方法属性	实例方法	静态方法