# 基础题（）

## 用最有效的的方法算出2乘以8等于几

|  |
| --- |
| 答案：2<<3 |

## Math.round(11.5)和Math.round(-11.5)的值是多少？

|  |
| --- |
| Math.round(11.5)：12  Math.round(-11.5)：-11 |

## 两个对象a和b，请问a==b和a.equals(b)有什么区别？

|  |
| --- |
| a==b：比较对象地址  a.equals(b)：如果a对象没有重写过equals方法，效果和==相同，如果重写了就按照重写的规则比较。 |

## switch是否能作用在byte上，是否能作用在long上，是否能作用在String上？

|  |
| --- |
| 答案一：switch可以作用在byte上，不能作用在long上，JDK1.7之后可以作用在String上。 |
| 答案二：switch支持的类型byte,short,int,char，JDK1.5之后支持枚举，JDK1.7之后支持String类型。 |

## char型变量中是否可以存储一个汉字？

|  |
| --- |
| 能，因为Java一个字符是2个字节，每一个字符使用Unicode编码表示 |

## float f=3.4;是否正确，表达式15/2\*2的值是多少

|  |
| --- |
| 答案：不正确，float f = 3.4F; |
| 答案：14 |

## 编写代码实现两个变量值交换，int m = 3, n =5;

|  |
| --- |
| 答案一：  int temp = m;  m = n;  n = temp; |
| 答案二：  m = m + n;  n = m - n;  m = m - n; |
| 答案三：  m = m ^ n;  n = m ^ n;  m = m ^ n; |

## Java的基本数据类型有哪些？String是基本数据类型吗？

|  |
| --- |
| 基本数据类型有：byte,short,int,long,float,double,char,boolean  String是引用数据类型，不是基本数据类型 |

## 数组有没有length()方法？String有没有length()方法？File有没有length()方法？ArrayList有没有length()方法？

|  |
| --- |
| 数组没有length()方法，但是有length属性。  String和File有length()方法。  ArrayList没有length()方法，有size()方法获取有效元素个数。 |

## String str = new String(“hello”);创建了哪些对象？

|  |
| --- |
| 字符串常量池中有一个对象，堆中有一个字符串对象。 |

## 如何将String类型转化Number类型？举例说明String str = “123”;

|  |
| --- |
| 答任意一个都对：  Integer num1 = **new** Integer(str);  或  **int** num2 = Integer.*parseInt*(str);  或  Integer num3 = Integer.*valueOf*(str); |

## 以下代码的运行结果：

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **char** x = 'x';  **int** i = 10;  System.***out***.println(**true**? x : i);  System.***out***.println(**true**? 'x' : 10);  } |
| 答案：  120  x |
| /\*  \* 如果其中有一个是变量，按照自动类型转换规则处理成一致的类型；  \* 如果都是常量，如果一个是char，如果另一个是[0~65535]之间的整数按char处理；  \* 如果一个是char，另一个是其他，按照自动类型转换规则处理成一致的类型；  \*/ |

## 以下代码的执行结果

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int** a = 8, b = 3;  System.***out***.println(a>>>b);  System.***out***.println(a>>>b | 2);  } |
| 答案：  1  3 |

## 下面程序片段的输出结果是？

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int** a = 3;  **int** b = 1;  **if**(a = b){  System.***out***.println("Equal");  }**else**{  System.***out***.println("Not Equal");  }  } |
| 答案：编译不通过 |

## 执行如下代码后，c的值是多少？

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int** a = 0;  **int** c = 0;  **do** {  --c;  a = a - 1;  } **while** (a >= 0);  System.***out***.println("c = " + c);  } |
| 答案：c = -1 |

## 以下代码的运行结果？

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int** i=10;  **while**(i>0){  i = i +1;  **if**(i==10){  **break**;  }  }  System.***out***.println("i=" + i);  } |
| 答案一：是一个负数，因为i一直累加会超过int的存储范围 |
| 答案二：死循环 |

## 修正如下代码

下面是一段程序，目的是输出10个=，但是不小心代码写错了，现在需要修改代码，使得程序完成功能，但是只能“增加”或“修改”其中“一个”字符，很明显，将i--改为i++，可以完成功能，但是需要修改“两个”字符，所以并不是一个正确的答案？

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int** n=10;  **for** (**int** i = 0; i < n; i--) {  System.***out***.println("=");  }  } |
| i<n修改为-i<n |

## 以下代码的运行结果是什么？

|  |
| --- |
| **public** **class** Test {  **public** **static** **boolean** foo(**char** c) {  System.***out***.print(c);  **return** **true**;  }  **public** **static** **void** main(String[] args) {  **int** i = 0;  **for** (*foo*('A'); *foo*('B') && (i < 2); *foo*('C')) {  i++;// 1 2  *foo*('D');  }  }  } |
| 答案：ABDCBDCB |

## 以下代码的执行结果是什么

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int** i = 0;  *change*(i);  i = i++;  System.***out***.println("i = " + i);  }  **public** **static** **void** change(**int** i){  i++;  } |
| 答案：i = 0 |

## 以下程序的运行结果：

|  |
| --- |
| **public** **static** **void** main(String[] args) {  String str = **new** String("world");  **char**[] ch = **new** **char**[]{'h','e','l','l','o'};  *change*(str,ch);  System.***out***.println(str);  System.***out***.println(String.*valueOf*(ch));  }  **public** **static** **void** change(String str, **char**[] arr){  str = "change";  arr[0] = 'a';  arr[1] = 'b';  arr[2] = 'c';  arr[3] = 'd';  arr[4] = 'e';  } |
| 答案：  world  abcde |

## 以下代码的运行结果是：

|  |
| --- |
| **public** **static** **void** main(String[] args) {  Integer i1 = 128;  Integer i2 = 128;  **int** i3 = 128;  **int** i4 = 128;  System.***out***.println(i1 == i2);  System.***out***.println(i3 == i4);  System.***out***.println(i1 == i3);  } |
| 答案：  false  true  true |

## 以下代码的运行结果

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **double** a = 2.0;  **double** b = 2.0;  Double c = 2.0;  Double d = 2.0;  System.***out***.println(a == b);  System.***out***.println(c == d);  System.***out***.println(a == d);  } |
| 答案：  true  false  true |

## 以下代码的运行结果是？

|  |
| --- |
| **public** **class** Test {  **int** a;  **int** b;  **public** **void** f(){  a = 0;  b = 0;  **int**[] c = {0};  g(b,c);  System.***out***.println(a + " " + b + " " + c[0]);  }  **public** **void** g(**int** b, **int**[] c){  a = 1;  b = 1;  c[0] = 1;  }  **public** **static** **void** main(String[] args) {  Test t = **new** Test();  t.f();  }  } |
| 答案：1 0 1 |

## 以下代码的运行结果是？

|  |
| --- |
| **public** **class** Test {  **static** **int** *x*, *y*, *z*;  **static** {  **int** x = 5;  x--;  }  **static** {  *x*--;  }  **public** **static** **void** main(String[] args) {  System.***out***.println("x=" + *x*);  *z*--;  *method*();  System.***out***.println("result:" + (*z* + *y* + ++*z*));  }  **public** **static** **void** method() {  *y* = *z*++ + ++*z*;  }  } |
| 答案：  x=-1  result:3 |

## 以下程序的运行结果是：

|  |
| --- |
| **public** **class** Test {  **public** **static** **void** main(String[] args) {  **new** A(**new** B());  }  }  **class** A{  **public** A(){  System.***out***.println("A");  }  **public** A(B b){  **this**();  System.***out***.println("AB");  }  }  **class** B{  **public** B(){  System.***out***.println("B");  }  } |
| 答案：  B  A  AB |

## 如下代码是否可以编译通过，如果可以，运行结果是什么？

|  |
| --- |
| **interface** A{  **int** ***x*** = 0;  }  **class** B{  **int** x = 1;  }  **class** C **extends** B **implements** A{  **public** **void** printX(){  System.***out***.println(x);  }  **public** **static** **void** main(String[] args) {  **new** C().printX();  }  } |
| 答案：编译错误  System.***out***.println(x);报错，x有歧义 |

## 以下代码的运行结果：

|  |
| --- |
| **public** **class** Test {  **public** **static** **void** main(String[] args) {  Base b1 = **new** Base();  Base b2 = **new** Sub();  }  }  **class** Base{  Base(){  method(100);  }  **public** **void** method(**int** i){  System.***out***.println("base : " + i);  }  }  **class** Sub **extends** Base{  Sub(){  **super**.method(70);  }  **public** **void** method(**int** j){  System.***out***.println("sub : " + j);  }  } |
| 答案：  base : 100  sub : 100  base : 70 |

## 以下代码的执行过程？

|  |
| --- |
| **public** **static** **void** main(String[] args) {  **int** test = *test*(3,5);  System.***out***.println(test);  }  **public** **static** **int** test(**int** x, **int** y){  **int** result = x;  **try**{  **if**(x<0 || y<0){  **return** 0;  }  result = x + y;  **return** result;  }**finally**{  result = x - y;  }  } |
| 答案：8 |

## 以下代码的运行结果？

|  |
| --- |
| **public** **static** **void** main(String[] args) {  Integer[] datas = {1,2,3,4,5};  List<Integer> list = Arrays.*asList*(datas);  list.add(5);  System.***out***.println(list.size());  } |
| 运行异常，不允许添加元素 |

## 在{1}添加什么代码，可以保证如下代码输出100

提示：t.wait() 或 t.jion() 或 t.yield() 或 t.interrupt()？

|  |
| --- |
| **public** **class** Test {  **public** **static** **void** main(String[] args) {  MyThread m = **new** MyThread();  Thread t = **new** Thread(m);  t.start();    {1}    **int** j = m.i;  System.***out***.println(j);  }  }  **class** MyThread **implements** Runnable{  **int** i;  **public** **void** run(){  **try** {  Thread.*sleep*(1000);  } **catch** (InterruptedException e) {  e.printStackTrace();  }  i=100;  }  } |
| 答案：t.join() |

## 以下代码如何优化

|  |
| --- |
| if(username.equals(“admin”){  ....  } |
| 答案：  if(“admin”.equals(username)){  } |

# 基础编程题（）

## 用循环控制语句打印输出：1+3+5+...+99=?的结果

|  |
| --- |
|  |
|  |

## 请写一个冒泡排序，实现{5,7,3,9,2}从小到大排序

|  |
| --- |
|  |
|  |

## 编写方法实现：求某年某月某日是这一年的第几天

提示：闰年（1）能被4整除不能被100整除（2）能被400整除

|  |
| --- |
| **public** **static** **int** daysOfYear(**int** year, **int** month, **int** day){  **int**[] daysOfMonth = {31,28,31,30,31,30,31,31,30,31,30,31};  补充代码  } |
|  |
|  |

## 通项公式如下：f(n)=n + (n-1) + (n-2) + .... + 1，其中n是大于等于5并且小于10000的整数，例如：f(5) = 5 + 4 + 3 + 2 + 1，f(10) = 10 + 9 + 8 + 7+ 6 + 5 + 4 + 3 + 2 + 1，请用非递归的方式完成方法long f( int n)的方法体。

|  |
| --- |
|  |

## 求1+2！+3！+...+20！的和

|  |
| --- |
|  |

## 输出一个如下图形，一共有n行，第n行有2n-1个\*，完成方法public void printStar(int n)的方法体

|  |
| --- |
| \*  \*\*\*  \*\*\*\*\*  \*\*\*\*\*\*\*  \*\*\*\*\*\*\*\*\* |
|  |

## 请编写代码使用把一个字符串反转，例如：hello1234，反转后：4321olleh。

|  |
| --- |
|  |
|  |

## 编写代码实现，从一个标准url里取出文件的扩展名，尽可能高效。

|  |
| --- |
| **public** **static** **void** main(String[] args) {  String str = *fileExtNameFromUrl*("http://localhost:8080/testweb/index.html");  System.***out***.println(str);  }    **public** **static** String fileExtNameFromUrl(String url){  补充代码  } |
|  |

## 有一个字符串String abc = “342567891”，请写程序将字符串abc进行升序，可以使用JDK API中的现有的功能方法。

|  |
| --- |
|  |

## 编写一个懒汉式单例设计模式

|  |
| --- |
|  |

## 请编写一个饿汉式单例设计模式

|  |
| --- |
|  |
|  |
|  |

## 补充如下枚举类型的代码，使得如下代码达到运行效果

单词提示：monday,tuesday,wednesday,thursday,friday,saturday,sunday

|  |
| --- |
| **import** java.util.Scanner;  **public** **class** TestWeek {  **public** **static** **void** main(String[] args) {  Scanner input = **new** Scanner(System.***in***);  System.***out***.print("今天是星期几(1-7)：");  **int** number = input.nextInt();//假设输入的是2  Week w = Week.*getByNumber*(number);  System.***out***.println("今天是：" + w);//今天是：TUESDAY(2,星期二)  }  }  **enum** Week{  **（1）**    **private** **int** number;  **private** String decription;    **private** Week(**int** number, String decription) {  **this**.number = number;  **this**.decription = decription;  }    **public** **static** Week getByNumber(**int** number){  （2）  }  @Override  **public** String toString() {  **return** **super**.toString()+"(" + number + ","+ decription + ")";  }  } |
|  |

## 写一段代码实现在遍历ArrayList时移除一个元素，例如：”java”？

|  |
| --- |
| **import** java.util.ArrayList;  **import** java.util.Iterator;  **public** **class** Test {  **public** **static** **void** main(String[] args) {  ArrayList<String> list = **new** ArrayList<String>();  list.add("hello");  list.add("java");  list.add("world");  补充代码  }  } |
|  |

## 把如下信息添加到Map中，并遍历显示，请正确指定泛型

|  |
| --- |
| 浙江省  绍兴市  温州市  湖州市  嘉兴市  台州市  金华市  舟山市  衢州市  丽水市  海南省  海口市  三亚市  北京市  北京市 |
|  |
|  |
|  |

## 完成在如下Map中查询城市信息

已知有省份Provice类型，有属性省份编号id和名称name，有城市City类型，有属性城市编号id和名称name,所属省份编号pid，以及所有信息现保存在一个Map中，现在要在map中，根据省份编号，查找这个省份下所有的城市。

|  |  |
| --- | --- |
| **import** java.util.HashSet;  **import** java.util.HashMap;  **import** java.util.Set;  **public** **class** AreaManager {  **private** HashMap<Province,HashSet<City>> map;    **public** AreaManager(){  map = **new** HashMap<Province,HashSet<City>>();    HashSet<City> bj = **new** HashSet<City>();  bj.add(**new** City(1,"北京市",1));  map.put(**new** Province(1,"北京市"), bj);    HashSet<City> hn = **new** HashSet<City>();  hn.add(**new** City(1,"海口市",2));  hn.add(**new** City(2,"三亚市",2));  map.put(**new** Province(2,"海南省"), hn);    HashSet<City> zj = **new** HashSet<City>();  zj.add(**new** City(1,"绍兴市",3));  zj.add(**new** City(2,"温州市",3));  zj.add(**new** City(3,"湖州市",3));  zj.add(**new** City(4,"嘉兴市",3));  zj.add(**new** City(5,"台州市",3));  zj.add(**new** City(6,"金华市",3));  zj.add(**new** City(7,"舟山市",3));  zj.add(**new** City(8,"衢州市",3));  zj.add(**new** City(9,"丽水市",3));  map.put(**new** Province(3,"浙江省"), zj);  }    **public** HashSet<City> findCity(**int** pid){  补充代码  }  } | 1:北京市  1:北京市  2:海南省  1:海口市  2:三亚市  3:浙江省  1:绍兴市  2:温州市  7:舟山市  8:衢州市  9:丽水市  5:台州市  4:嘉兴市  6:金华市  3:湖州市 |
|  | |

## 请编写代码读取一个项目根目录下info.properties文件

里面的内容有user=atguigu等，请获取user的value中，并在控制台打印

|  |
| --- |
|  |

## 请编写代码把一个GBK的文本文件内容读取后存储到一个UTF-8的文本文件中。

|  |
| --- |
|  |

## 用实现Runnable接口的方式，启动一个线程完成在线程中打印1-100的数字

|  |
| --- |
|  |
|  |

# 基础简答题（5分/题）

## break、continue、return的区别？

|  |
| --- |
| break用于switch和循环，用于结束switch，和当前循环  continue用于循环，用于结束本次循环  return用于结束当前方法，还可以用于return 返回值;返回结果 |

## 请列出一些常用的类、接口、包，各至少5个

|  |
| --- |
| 注意答案不固定  常用类：String，Math，,ArrayList，HashMap，System  常用接口：Comparable，Comparator，Runnable，Serializable，Collection  常用包：java.lang， java.util，java.io，java.net，java.text，java.lang.reflect |

## 访问修饰符的作用范围由大到小，及各自的范围是什么？可以修饰什么？

|  |
| --- |
| public->protected->缺省(default)->private    外部类只能使用public或缺省。  如果是修饰类的成员，四种都可以。 |

## 请对public static void main(String[] args)的每一个单词做解释？

|  |
| --- |
| public：公共的，用它修改的类或成员在任意位置可见  static：静态的，用它修改的方法，可以不用创建对象就可以调用  void：表示该方法没有返回值  main：Java的主方法名，JavaSE的程序入口  String[]：字符串数组，这是main方法的形参类型，可以通过命令行参数传值  args：这是main方法的形参名，如果要在main中使用命令行参数，可以遍历该args数组。 |

## 请解释Overload与Override的区别？

|  |
| --- |
| Overload是方法重载，指的是在同一个类中，方法名称相同，形参列表不同的两个或者多个方法，和返回值类型无关。  Override是方法的重写，指的是子类在继承父类时，当父类的方法体不适用于子类时，子类可重写父类的方法。重写必须遵守方法名和形参列表与父类的被重写的方法相同，而返回值类型可以小于等于父类被重写的方法（如果是基本数据类型和void必须相同），权限修饰符可以大于等于父类被重写的方法，抛出的异常列表可以小于等于父类被重写的方法。 |

## final、finalize、finally的区别？

|  |
| --- |
| final是表示最终的，是一个修饰符，修饰类时表示不能被继承，修饰方法时表示不能被子类重写，修饰属性和局部变量时表示值不能被修改，是个常量。  finally是表示最终块，是异常处理的一部分，和try..catch一起使用，不管是否发生异常都要执行的代码放在finally块中。  finalize是表示最终方法，是java.lang.Object类的一个方法，在对象被垃圾回收时调用。 |

## 面向对象的基本特征有哪些？并作出解释

|  |
| --- |
| 答出三个基本特征给3分  面向对象的基本特征有：  （1）封装：封装的好处就是安全，方便。封装隐藏了对象的具体实现，当要操纵对象时，只需调用其中的方法，而不用管方法的具体实现。属性的封装就是属性私有化并提供get/set方法，这样外界只能通过get/set方法来操作属性，行为变得可控。  （2）继承：继承的好处就是代码的复用和扩展。继承可以保留父类的属性和方法，同时子类又可以扩展自己的属性和方法。  （3）多态：目的是实现代码的灵活性，多态体现在重载和重写方法，更多的时候指的是对象的多态性，即当父类的变量指向子类的对象时，那么调用子类重写的方法时，运行的是子类重写过的代码，从而实现同一个父类的变量，因为赋值的子类对象不同而体现出不同的功能。应用主要体现在多态参数和多态数组中。 |

## 请解释String、StringBuilder、StringBuffer的区别？

|  |
| --- |
| String是不可变的字符序列，因此字符串常量存储在常量池中，而StringBuilder和StringBuffer是可变的字符序列。  String对象是常量对象，因此一旦拼接和修改就会产生新的String对象。  SringBuffer和StringBuilder可以在原对象上进行append,insert,delete,replace等修改。  StringBuilder和StringBuffer是完全兼容的API，但是StringBuilder是线程不安全的、StringBuffer是线程安全的。 |

## 如下关于String比较的代码的运行结果是什么

|  |
| --- |
| **public** **static** **void** main(String[] args) {  String str1 = "1";  String str2 = "2";  String str3 = **new** String("1");  **final** String str4 = "2";  **final** String str5 = **new** String("2");  String str6 = "12";    String str7 = "1" + "2";  String str8 = str1 + "2";  String str9 = str1 + str2;  String str10 = str3 + str4;  String str11 = "1" + str4;  String str12 = "1" + str5;  String str13 = (str1 + str2).intern();    System.***out***.println("(1)"+ (str1 == str3));  System.***out***.println("(2)"+ (str2 == str4));  System.***out***.println("(3)"+ (str4 == str5));  System.***out***.println("(4)"+ (str6 == str7));  System.***out***.println("(5)"+ (str6 == str8));  System.***out***.println("(6)"+ (str6 == str9));  System.***out***.println("(7)"+ (str6 == str10));  System.***out***.println("(8)"+ (str6 == str11));  System.***out***.println("(9)"+ (str6 == str12));  System.***out***.println("(10)"+ (str6 == str13));  } |
| 答案：一个0.5分  (1)false  (2)true  (3)false  (4)true  (5)false  (6)false  (7)false  (8)true  (9)false  (10)true |

## BigDecimal和float、double有什么区别？BigInteger和int、long有什么区别？

|  |
| --- |
| 在用C或者C++处理大数时感觉非常麻烦，但是在Java中有两个类BigInteger和BigDecimal分别表示大整数类和大浮点数类，至于两个类的对象能表示最大范围不清楚，理论上能够表示无线大的数，只要计算机内存足够大。这两个类都在java.math.\*包中，因此每次必须在开头处引用该包。  BigInteger和BigDecimal是用对象表示数据的，其实底层是用字符串存储数据的，因此无法使用“算术运算符”进行算术运算，只能调用add等方法完成计算。  而float,double,int,long等是基本数据类型，可以直接用算术运算符运算，但是有存储范围有限以及精度的问题。 |

## 请对Java的基本数据类型与包装类做解释？

|  |
| --- |
| Java的八种基本数据类型与包装类：  byte <--> Byte  short <-->Short  int <--> Integer  long <--> Long  float <--> Float  double <--> Double  char <--> Character  boolean <--> Boolean  八种基本数据类型只与自己的包装类之间进行装箱与拆箱。JDK1.5之后支持自动装箱与自动拆箱。 |

## java.lang.Comparable与java.util.Comparator有什么区别？

|  |
| --- |
| java.lang.Comparable<T>被称为自然排序接口。包含一个抽象方法int compareTo(T obj)，如果当前对象比指定对象obj大，则返回正整数，小则返回负整数，相等返回0。  java.util.Comparator<T>被称为定制排序接口。包含一个抽象方法int compare(T t1, T t2)，如果t1大于t2，则返回正整数，t1小于t2，则返回负整数，相等返回0。  如果在使用Arrays.sort(数组)或Collections.sort(Collection集合)方法时，TreeSet和TreeMap时元素默认按照Comparable比较规则排序；也可以单独为Arrays.sort(数组)或Collections.sort(Collection集合)方法时，TreeSet和TreeMap指定Comparator定制比较器对象。 |

## 请解释Collection 和 Collections 的区别?List、Set、Map是否继承Collection？

|  |
| --- |
| Collection是接口，是List和Set系列接口的父接口。是Collection系列接口的根接口。  Collections是工具类，其中提供了很多静态方法来操作各种集合。  List和Set继承Collection，Map不继承Collection。 |

## 请解释Arraylist、Linkedlist和Vector的区别？

|  |
| --- |
| ArrayList：是线程不安全的动态数组，底层是数组结构，JDK1.7后初始化为空数组，在添加第一个元素时初始化为长度为10的数组，如果容量满了，按照1.5倍扩容。支持foreach和Iterator遍历。  Vector：是线程安全的动态数组，底层是数组结构，初始化为长度为10的数组，如果容量满了，按照2.0倍扩容。除了支持foreach和Iterator遍历，还支持Enumeration迭代。  LinkedList：是双向链表，底层是链表结构。当频繁在集合中插入、删除元素时，效率较高，但是查找遍历的效率较低。 |

## Hashtable与HashMap的区别？如何解决那个线程不安全的问题？

|  |
| --- |
| Hashtable是线程安全的哈希表，底层结构是数组+链表。  HashMap是线程不安全的哈希表，底层结构是JDK1.7时数组+链表，JDK1.8时数组+链表/红黑树。  HashMap的线程安全问题可以使用Collections的synchronizedMap(Map<K,V> m) 方法解决。 |

## List、Map、Set 三个接口，存取元素时，各有什么特点？

|  |
| --- |
| List：是有序的，可重复的，添加元素的方法是add，可以根据索引获取元素。  Set：是无序的，不可重复的，添加元素的方式是add，HashSet和LinkedHashSet的元素是依据hashCode和equals区别元素是否相等，而TreeSet是依据compareTo或compare区别元素是否相等。  Map：是存储键值对的，添加的方法是put(key,value)，可以根据key获取value。 |

## ArrayList和LinkedList的底层实现（存储结构、扩容机制）

|  |
| --- |
| 1.ArrayList是实现了基于动态数组的数据结构，LinkedList基于链表的数据结构。  2.对于随机访问get和set，ArrayList觉得优于LinkedList，因为LinkedList要移动指针。  3.对于新增和删除操作add和remove，LinedList比较占优势，因为ArrayList要移动数据。 这一点要看实际情况的。若只对单条数据插入或删除，ArrayList的速度反而优于LinkedList。但若是批量随机的插入删除数据，LinkedList的速度大大优于ArrayList. 因为ArrayList每插入一条数据，要移动插入点及之后的所有数据。 |

## 请列举一些常见的异常或错误类型（至少5个）

|  |
| --- |
| 运行时异常：  数组下标越界异常：ArrayIndexOutOfBoundsException  类型转换异常：ClassCastException  算术异常：ArithmeticException  空指针异常：NullPointerException  编译时异常：  IO操作异常：IOException  文件找不到异常：FileNotFoundException  已到达文件流末尾异常：EOFException  类找不到异常：ClassNotFoundException  没有对应的方法异常：NoSuchMethodException  错误：  堆内存溢出：OutOfMemoryError  栈内存溢出：StackOverflowError |

## 请解释Java异常处理的过程

|  |
| --- |
| Java的异常处理过程如下：  （1）当程序运行到某一句代码，如果发生了异常（可能是JVM判定的异常，也可能是遇到throw的），程序都会停下来，然后把异常信息封装到异常的对象中，并且“抛”出  （2）JVM会检测在这段程序代码的外围，是否有try...catch，如果有try...catch，就判断是否有catch可以捕获它，如果捕获了，程序就进入对应的catch块进行异常处理，处理后程序继续运行try..cath之后的代码。  （3）JVM会检测在这段程序代码的外围，根本就没有try...catch或者是有try...catch但是捕获不住，即类型对不上，JVM都会把这个异常对象抛出“上级，方法的调用者”  （4）上级一旦接到异常对象，处理过程还是1,2,3  （5）如果一直抛，一路上都没有可以捕获它，程序就崩溃了。 |

## 请解释Java异常处理机制相关的5个关键字

|  |
| --- |
| try：尝试执行可能发生异常的代码。  catch：尝试捕获try部分发生的异常。可以存在多个catch，如果多个catch的异常类型有继承关系，那么遵循子上父下。  finally：不管是否发生异常都要执行的代码放在finally块中。  throws：方法声明时显示抛出异常，指定该方法可能抛出的异常类型列表。  throw：手动抛出异常，可以抛出系统预定异常，也可以抛出用户自定异常，而且用户自定义异常必须用throw语句抛出，可以代替return语句结束方法运行。 |

## Java中的IO流的四大基类是什么（2分）,请列出常用的IO流类型（至少5个）（3分）

|  |
| --- |
| 所有的IO流都是从以下四个抽象基类，超级父类中分出来的：  （1）字节输入流：InputStream  （2）字节输出流：OutputStream  （3）字符输入流：Reader  （4）字符输出流：Writer  可以延伸出很多IO流，例如：和文件相关  （1）文件字节输入流：FileInputStream  （2）文件字节输出流：FileOutputStream  （3）文件字符输入流：FileReader  （4）文件字符输出流：FileWriter  例如：缓冲流  （1）字节输入缓冲流：BufferedInputStream  （2）字节输出缓冲流：BufferedOutputStream  （3）字符输入缓冲流：BufferedReader  （4）字符输出缓冲流：BufferedWriter  例如：转换流  （1）InputStreamReader：把字节输入流转为字符输入流，解码  （2）OutputStreamWriter：把字符输出流转为字节输出流，编码  例如：数据流  （1）字节输入数据流：DataInputStream  （2）字节输出数据流：DataOutputStream  例如：对象流  （1）对象输入流：ObjectInputStream，用于对象的序列化  （2）对象输出流：ObjectOutputStream，用于对象的反序列化  例如：打印流  （1）字节打印流：PrintStream  （2）字符打印流：PrintWriter |

## InputStream里的read()返回的是什么值,read(byte[] data)是什么意思,返回的是什么值。Reader里的read()返回的是什么值，read(char[] data)是什么意思，返回的是什么值。如果想要一次读取一行怎么办？

|  |
| --- |
| InputStream：  read()方法，返回的是所读取的字节的int型（范围0-255）  read（byte[] data）将读取的字节储存在这个数组，返回的是实际读取的字节数。  Reader：  read()方法，返回的是所读取的字符的int型（范围0-65535）  read(char[] data)将读取的字符存储在这个数组中，返回的是实际读取的字符数。  如何读取一行：  BufferedReader类中有readLine()方法。Scanner类中也有nextLine()方法。 |

## Java反射机制的作用？

|  |
| --- |
| 反射就是动态加载对象，并对对象进行剖析。Java反射机制的作用：  （1）在运行时创建任意类型的对象  （2）在运行时获取任意类型的信息  （3）在运行时获取和设置任意属性值  （4）在运行时调用任意对象的方法 |

## 如何获取Class的对象？4种方式

|  |
| --- |
| 获取Class对象的四种方式：  （1）类型名.class  （2）对象.getClass()  （3）Class.forName("类型的全名称")  （4）ClassLoader对象.loadClass("类型的全名称") |

## 编写多线程程序有几种实现方式？

|  |
| --- |
| JavaSE阶段考试答出两种即对：  1、继承Thread类，可以直接调用start()启动，有单继承限制，共享数据时需要使用static方式，只能选择当前类.class对象或其他共享对象当锁。  2、实现Runnable接口，必须借助Thread对象的start()启动，实现接口可以解决单继承限制问题，需要共享数据时，共享同一个Runnable对象即可，线程安全锁可以直接选择this对象。 |
| 企业面试阶段答案一如下：  1、继承Thread类，可以直接调用start()启动，有单继承的限制。  2、实现Runnable接口，必须借助Thread对象的start()启动，实现接口可以解决单继承限制问题。  3、使用ExecutorService、Callable、Future实现有返回结果的多线程 |
| 企业面试阶段答案二如下：  Java多线程实现方式主要有四种：继承Thread类、实现Runnable接口、实现Callable接口通过FutureTask包装器来创建Thread线程、使用ExecutorService、Callable、Future实现有返回结果的多线程。  其中前两种方式线程执行完后都没有返回值，后两种是带返回值的。 |

## 请阐述线程的生命周期？

|  |
| --- |
|  |

## Thread的start()和Runnable的run()有什么区别？

|  |
| --- |
| Thread的start()：启动一个线程是调用start()方法，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由JVM调度并执行。这并不意味着线程就会立即运行。  Runnable的run()：线程的线程体方法。所有线程类都必须实现的run()方法。 |

## sleep() 和 wait() 有什么区别?

|  |
| --- |
| sleep和wait都会导致当前线程进入阻塞状态，被挂起。  sleep不释放锁，睡眠时间到自动醒来，回到就绪状态  wait是会释放锁，要通过notify()或notifyAll()唤醒，回到就绪状态  sleep是在Thread类中声明的一个静态方法，Thread.sleep(毫秒)  wait是在Object类中声明的非静态的方法，必须锁对象调用 |

## 请阐述什么是线程安全问题，如何解决？

|  |
| --- |
| 当满足以下条件时，会出现线程安全问题：  （1）有多个线程  （2）使用共享数据  （3）有多句代码操作共享数据  如何解决？同步，即加锁 |

## 简要的写出进程和线程的区别（简单的写）？

|  |
| --- |
| （1）进程是操作系统资源的分配和调度的一个独立单元，而线程是CPU调度的基本单元  （2）同一个进程中可以包括多个线程，并且线程共享整个进程的资源（寄存器、堆栈、上下文），一个进程至少包括一个线程。 |

# 较难简答题

## Java虚拟机中内存分为哪些区域？每个区域的作用？哪些区域是线程共享的？

|  |
| --- |
| 1、程序计数器(寄存器)：当前线程所执行的字节码行号指示器  2、本地方法栈：同虚拟机栈，只不过本地方法栈为虚拟机使用到的native方法服务。  3、虚拟机栈：每个方法在执行的同时都会创建一个栈帧用来存放存储局部变量表、操作数表、动态连接、方法出口等信息，每一个方法从调用直至执行完成的过程，就对应着一个栈帧在虚拟机栈中入栈到出栈的过程。  4、堆：所有线程共享的一块内存区域。Java虚拟机所管理的内存中最大的一块，因为该内存区域的唯一目的就是存放对象实例。几乎所有的对象实例度在这里分配内存，也就是通常我们说的new对象，同时堆也是垃圾收集器管理的主要区域。  5、方法区：和堆一样，是各个线程共享的内存区域，用于存储已被虚拟机加载的类信息、常量、静态变量、和编译器即时编译后的代码等 |

## 请解释抽象类与接口的区别

JDK1.8之前抽象类与接口的差别很大，JDK1.8之后接口越来越像抽象类了。

|  |  |  |  |
| --- | --- | --- | --- |
|  | | 抽象类 | 接口 |
| 单继承限制 | | 有 | 一个类可以实现多个接口，  而且接口也可以继承多个接口 |
| 成员 | 属性 | 可以有 | 只能有公共的静态的常量属性 |
|  | 构造器 | 有 | 无 |
|  | 代码块 | 可以有 | 无 |
|  | 抽象方法 | 可以有 | 只能是公共的抽象方法 |
|  | 静态方法 | 可以有 | JDK1.8之后可以有公共的静态方法 |
|  | 方法的默认实现 | 可以有 | JDK1.8之后可以有公共的静态方法 |
| 相同点 | | 都不能直接实例化，都是上层的抽象层 | |

## Object类中equals方法的实现是什么？重写一个equals方法有什么注意事项？

|  |
| --- |
| Object类中的equals方法，对于任何非空引用值 x 和 y，当且仅当 x 和 y 引用同一个对象时，此方法才返回 true（x == y 具有值 true）。  在重写equals方法时，要注意满足离散数学上的特性  （1）自反性：对任意引用值x，x.equals(x)的返回值一定为true.  （2）对称性：对于任何引用值x,y,当且仅当y.equals(x)返回值为true时，x.equals(y)的返回值一定为true;  （3）传递性：如果x.equals(y)=true, y.equals(z)=true,则x.equals(z)=true  （4）一致性：如果参与比较的对象没任何改变，则对象比较的结果也不应该有任何改变  （5）非空性：任何非空的引用值x，x.equals(null)的返回值一定为false  注意：当此方法被重写时，通常有必要重写 hashCode 方法，以维护 hashCode 方法的常规协定，该协定声明   1. 相等对象必须具有相等的哈希码， 2. 两个对象的哈希码不相等，那么equals一定不相等。 3. 两个对象的哈希码相等，那么equals结果可能相等也可能不相等 |

## 比特(Bit),字节(Byte),字符(char/word),各有什么区别，通常说存储容量为KB,MB,GB,TB又是什么意思？通常说传输速率有bps和Bps有什么区别？

|  |
| --- |
| Bit最小的二进制单位 ，是计算机的操作部分，取值0或者1。  Byte是是计算机信息技术用于计量存储容量的一种计量单位，由8位bit组成，取值（-128-127）。  char/word是用户的可读写的最小单位，在Java里面一个char类型的变量占2个字节，取值（0-65535），但实际一个char存储到文件中占几个字节要看字符编码方式。  1KB = 1024Byte，1MB = 1024KB，1GB = 1024MB，1TB = 1024GB。  bps是 bits per second 的简称，一般用于表示网络或USB等接口的数据传输速率。Bps即是Byte per second 的简称，电脑一般都以Bps 显示速度，如1Mbps 大约等同 128 KBps。 |

## 运行时异常与编译时异常有何异同？请列举一些运行时异常和编译时异常的类型。

|  |
| --- |
| 运行时异常是非受检异常，是RuntimeException的子类，即编译器无法检测，因此也不会强制要求程序员处理。  编译时异常是受检异常，编译器检测到代码抛出编译时异常时，会要求程序员必须对该异常做处理(throws或try...catch)否则，编译不通过。  运行时异常：  数组下标越界异常：ArrayIndexOutOfBoundsException  类型转换异常：ClassCastException  算术异常：ArithmeticException  空指针异常：NullPointerException  编译时异常：  IO操作异常：IOException  文件找不到异常：FileNotFoundException  已到达文件流末尾异常：EOFException  类找不到异常：ClassNotFoundException  没有对应的方法异常：NoSuchMethodException |

## HashMap的底层实现及扩容机制？

|  |
| --- |
| HashMap在JDK1.8之前：底层实现是数组+链表，扩容机制是当table中元素的个数已经达到阈值（table.length\*0.75）时扩容。  （1）先计算key的hashCode值  （2）根据key的hashCode值再与数组table的长度做一个运算（%或&)，得到一个索引位置index，决定它在table那个元素（桶）下面。  （3）如果table[index]是空的，把映射关系构建为一个Entry（Entry是实现了Map.Entry类型）的对象，直接放进去  （4）如果table[index]不是空的，判断table[index]的元素的key与我们新的key调用equals方法是否相等，如果相等，替换  （5）如果table[index]不是空的，判断table[index]的元素的key与我们新的key调用equals方法全部都不相等，  判断是否table中元素的个数已经达到阈值（table.length\*0.75），如果已经达到，先扩容，回到（2）  否则就构建一个Entry的对象，连接到链表的上面。  HashMap在JDK1.8之后：底层实现是数组+链表/红黑树，扩容机制  （1）先计算key的hashCode值  （2）根据key的hashCode值再与数组table的长度做一个&运算，得到一个索引位置index，决定它在table那个元素（桶）下面。  （3）如果table[index]是空的，把映射关系构建为一个Node（Node是实现了Map.Entry类型）的对象，直接放进去  （4）如果table[index]不是空的，判断table[index]的元素的key与我们新的key调用equals方法是否相等，如果相等，就用新的value替换旧的value；  如果都不相等，再继续看  A：如果table[index]下面已经是一棵红黑树，那么直接把映射关系构建为一个TreeNode（TreeNode是Node的子类）的对象，放到某个叶子节点上  B：如果table[index]下面不是一棵红黑树，那么就又要判断它下面的链表的元素个数是否达到8个：  如果没有到达8个，那么直接把映射关系构建为Node的对象，直接连接到链表的下面。  如果已经达到8个，那么就再次判断table.length是否达到64，如果没有达到64，先扩容，回到（2）。如果已经达到64，就把该链表变成一颗红黑树，然后把新的映射关系构建一个TreeNode对象，放到某个叶子节点上。  （5）如果添加后size达到阈值（table.length\*0.75），要扩容重写调整所有元素的位置。 |

## 如何实现序列化，有什么意义

|  |
| --- |
| 如何实现序列化（5分）：  （1）实现Serializable接口或Externalizable接口，并且视情况而定指定一个序列化版本ID（serialVersionUID）值；而且要保留公共的无参构造。  （2）如果某个对象的属性也是引用数据类型，那么该数据类型也要实现Serializable接口或Externalizable接口；  （3）如果要序列化，则使用一个输出流来构造一个对象输出流ObjectOutputStream并通过writeObject(Object obj)方法就可以将实现对象写出(即保存其状态)；如果需要反序列化则可以用一个输入流建立对象输入流ObjectInputStream，然后通过readObject方法从流中读取对象。  （4）如果某些属性不参与序列化，如果是实现Serializable接口的，直接在属性前面加transient修饰，如果是实现Externalizable接口，那么只要在重写writeExternal()和readExternal()方法时，不处理该属性即可。  （5）注意：static修饰的属性不会被序列化  意义（3分）：  序列化就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化，即把对象的内容转成二进制数据。可以对流化后的对象进行读写操作，也可将流化后的对象传输于网络之间。序列化是为了解决对象流读写操作时可能引发的问题（如果不进行序列化可能会存在数据乱序的问题）。 |

## synchronized关键字的用法？

|  |
| --- |
| synchronized关键字是解决线程安全问题的方式之一。共有两种用法：  1、同步代码块  语法格式：  synchronized(锁对象){  需要加锁的代码  }  注意锁：  （1）任意类型的对象都可以当做锁  （2）多个线程之间共用一把锁，即多个线程之间共用同一个锁对象  （3）同步代码块的范围：不能太大，太小  2、同步方法  语法结构：  synchronized 【修饰符】 返回值类型 方法名（【形参列表】）【抛出异常列表】  同步方法的锁对象：  静态方法：当前类的Class对象，即当前类名.class  非静态方法：当前对象this（需要谨慎，确保是同一个this） |

## 请列出你所知道的设计模式，并做简单说明？

|  |  |
| --- | --- |
| 单例设计模式 | 单例就是该类只能返回一个实例。  单例所具备的特点：  1.私有化的构造函数  2.静态的全局变量存储唯一的实例 |
| 工厂设计模式 |  |
| 代理设计模式 |  |
| 生产者与消费者设计模式 |  |
| 模板设计模式 |  |
| 装饰者设计模式 |  |
| 适配器模式 |  |
| 观察者模式 |  |

# 较难编程题

## 判断101-200之间有多少个素数，并输出所有素数

|  |
| --- |
|  |

## 一个球从100米高度自由落下，每次落地后反跳回原高度的一半，再落下，求它在第10次落地时，共经过多少米？第10次反弹多高？

|  |
| --- |
|  |

## 用100元钱买100支笔，其中钢笔3元/支，圆珠笔2元/支，铅笔0.5元/支，问钢笔、圆珠笔和铅笔可以各买多少支？请写main方法打印需要买的数目。

|  |
| --- |
|  |

## 通项公式如下：f(n)=n + (n-1) + (n-2) + .... + 1，其中n是大于等于5并且小于10000的整数，例如：f(5) = 5 + 4 + 3 + 2 + 1，f(10) = 10 + 9 + 8 + 7+ 6 + 5 + 4 + 3 + 2 + 1，请用递归的方式完成方法long f( int n)的方法体。

|  |
| --- |
|  |

## 求1+2！+3！+...+20！的和

|  |
| --- |
|  |

## 第一个人10，第2个比第1个人大2岁，以此类推，请用递归方式计算出第8个人多大？

|  |
| --- |
|  |

## 有n步台阶，一次只能上1步或2步，共有多少种走法？

|  |
| --- |
|  |
|  |

## 输入整型数98765，输出是56789

|  |
| --- |
|  |

## 有一个字符串，其中包含中文字符、英文字符和数字字符，请统计和打印出各个字符的字数。

举例说明： String content = “中中国55kkfff”;

统计出：

中：2

国：1

5：2

k：2

f：3

|  |
| --- |
|  |

## 斐波纳契数列（Fibonacci Sequence），又称黄金分割数列。

一列数的规则如下：1、1、2、3、5、8、13、21、34....求第n位数是多少？

在数学上，斐波纳契数列以如下被以递归的方法定义：F0=0，F1=1，Fn=F(n-1)+F(n-2)（n>=2，n∈N\*）在现代物理、准晶体结构、化学等领域，斐波纳契数列都有直接的应用

|  |
| --- |
|  |
|  |

## 请使用二分查找算法查找字符数组{“a”,”b”,”c”,”d”,”e”,”f”,”g”,”h”}中”g”元素的位置？

|  |
| --- |
|  |

## 消除下面集合中重复元素？

List list = Arrays.asList(1,2,3,3,4,4,5,5,6,1,9,3,25,4);

|  |
| --- |
|  |

## 请用wait()和notify()方法编写一个生产者消费者设计模式程序?

|  |
| --- |
|  |