**Java基本概念：**

1. **标识符P37**

标识符的开头必须是英文字母、下划线或美元符号。

标识符不能包含空格，只能含‘$’，不能包含‘#’、‘&’、‘@’等其他特殊字符

命名规范:

包名：全小写

类名：首字母大写

接口名：首字母大写

方法名：首字母小写，中间每个单词的首字母大写

变量名：全小写

常量名：全大写，多个单词由\_隔开。（使用关键字final修饰。其定义格式为：final Type varName=value）

1. **基本数据类型变量定义、类型转换P40、46、47**

（1）P40

（2）自动类型转换（short、char、byte🡪int 🡪long🡪float 🡪double）

强制类型转换（a = (int)b;）

1. **++，--，//前缀先增减后引用，后缀先引用后增减 P51**

（自增、自减的运算符中的四个符号等级，且高于双目算术运算符。自增、自减只能用于变量，而不能作用于常量或表达式上）

1. **Switch/case语法P68-70**

**有关switch的几点说明：**

**（1）括号内的表达式的返回值类型必须是int，byte，char，short、String类型和枚举类型之一。**

**（2）case子句中的值必须是常量，case后的每个常量表达式必须各不相同。**

**（3）default子句是任选的，并且可以放在任何位置。**

**（4）每个case之后的执行语句可多于一个，但不必加{}**

**（5）需要break语句跳出；**

（1）switch(a){

case 1:语句1；break；

case 2:语句2；break;

default:语句n+1;break;

}

（2）switch(a){

case 1🡪语句1；

case 2🡪语句2；

default🡪语句n+1;

}

1. **数组定义和初始化的语法P80-83**

int[] a = new int [10]; //这里每个数组元素将被初始化为0；

for…each循环遍历

int[] nums={1,2,3,4,5,,6,7,8,9,10};

int result = 0;

for(int e:nums)

result +=e;

1. **访问修饰符P106**

public、private、protected、以及无修饰符

（类本身只有两种访问控制，只要在class前没有使用public 修饰符，源文件的名称可以是一切合法的名称，带有public 修饰符的类的类名必须与源文件名相同）。

1. **类构造方法的基本用法P109**

class Person｛｝

在主函数中调用时：Person p = new Person(); //声明并创建对象。

1. **类的继承、多态。P112-125**

在类的声明中加入extends子句来创建一个类的子类：

class SubClass extends SuperClass

子类可以继承父类中访问权限设定为public、protected、default的成员变量和方法，但是不能继承访问权限为private的成员变量和方法。

**方法重写的条件 P115**

·方法名、参数列表和返回值完全相同。

·访问控制范围不能被缩小

·抛出的异常不能被扩大

**super、this关键字 P117**

（1）this方法代表的是一个构造方法对其他构造方法的调用。this（）方法必须放在构造方法的第一行，即它的前面不能再有其他语句

（2）Super()也必须是子类构造方法中的第一个执行语句；如果调用了Super()，则会调用它的直接超类的构造方法；

Super()可以用来访问被子类的成员隐藏的超类成员。Super()指向这个对象的父类。Super()可以用来引用父类中的（被覆盖的）方法、（被隐藏的）变量及构造方法

**注:**在子类的非默认构造方法执行前，也会先执行父类的默认构造方法（即无参的构造方法）

**对象的比较： P120**

（1）==用来比较两个引用是否指向同一个对象。

（2）equals()需要在类定义中重写，用来识别两个对象是否具有相同的类型和内容。

**多态实现的三个条件 P122**

（1）继承和方法重写

（2）子类对象声明超类类型

（3）运行时类型识别（RTTI）,对于重写的方法，Java运行时系统会根据调用该方法的实例类型来决定将选择哪个方法来调用。

1. **Static语法P131-133**

访问静态变量可以直接通过访问类名来访问

例如，class A1{static int b = 10;}

在另一个类中可以通过A1.b获得b的值，如果不是静态变量，则需要用 newA1().b来获得b的值.

在方法使用变量时，需要注意以下规则：

·实例变量或者类变量在定义初始化时，都不能超前引用。

·实例方法既可以使用实例变量，又可以使用静态变量；而静态方法只能使用静态变量，不能直接使用实例变量

1. **抽象类、接口的基本语法P135-139**

（1）抽象类：抽象类必须被继承，抽象方法必须被重写。抽象方法只需声明，无须实现。

子类只有覆盖实现父类中的所有抽象(abstract)方法才能被定义成非抽象类，否则也只能被定义为抽象类。

注：abstract class A{}不能使用A a = new A();来调用

静态方法不能为抽象方法。

（2）接口（一种特殊的抽象类）。

（a）接口中的变量必须是用public static final修饰的，方法必须是用public static 修饰的。

（b）接口定义示例：

interface A{

void mothod1(int i);

}//系统会自动给方法加上public static修饰符

（c）接口的实现使用implements关键字

（d）利用接口可以实现多重继承，即一个类可以实现多个接口，在implements子句中用逗号分隔

class sofa extends Chair implements Lie,HealthCare{}//继承与接口之间无需用逗号隔开

（e）接口不允许被实例化，所以接口中没有构造方法

接口的优点：

·可以实现不相关类的相同行为，而无需考虑这些类之间的层次关系。

·可以指明多个类需要实现的方法。

·可以了解对象的交互界面，而无需了解对象所对应的类。

1. **内部类**

**对象成员内部类：**

**创建非静态内部类很容易，只需要定义一个类让该类作为其他类的非静态成员。该非静态内部类和成员变量或成员方法没有区别，同样可以在非静态内部类前面加可以修饰成员的修饰符。**

**在外部类外访问内部类：**

**(1)Outer.Inner on = new Outer().new Inner();**

**(2)Outer ot = new Outer();**

**ot.Inner on = ot.new Inner();**

**//如果内部类被修饰成private，成为私有的成员，那么就不能在外部类外来访问私有的内部类了。**

**静态内部类：**

**创建静态内部类的形式和创建非静态内部类的形式很相似的，只是使用static修饰符来声明一个内部类。**

**局部内部类：**

**定义在方法体或更小的语句块中的类；**

**匿名内部类：**

**没有类名的类，没有构造方法，也没有任何修饰符来声明它；（判断标志，如果方法中还有定义方法，那么内层方法其实是匿名内部类的方法）**

1. **自动装箱和拆箱**

Integer A = 5; //装箱

A = A+ 1; //拆箱再装箱

1. 枚举

Enmu num{1,2,3,4,5}

1. **注解：**

@override

当我们想要重写父类中的方法时，我们需要使用该注解去告知编译器我们想要重写这个方法。这样一来当父类中的方法移除或者发生更改时编译器将提示错误信息。

@Deprecated

当我们希望编译器知道某一方法不建议使用时，我们应该使用这个注解。Java在javadoc 中推荐使用该注解，我们应该提供为什么该方法不推荐使用以及替代的方法。

@SuppressWarings

这个是告诉编译器忽略特定的警告信息，它会一直作用于该程序元素的所有子元素，如果使用@SuppressWarnings修饰某个类取消显示某个编译器警告，同时又修饰该类里的某个方法取消显示另一个编译器警告，那么该方法将会同时取消显示着这两个编译器警告。

@SafeVarargs

Java7中加入了参数安全类型注解@SafeVarargs。它的目的是提醒开发者不要用参数做一些不安全的操作,它的存在会阻止编译器产生 unchecked 这样的警告。当开发者不想看到这样的警告，就可使用@SafeVarargs修饰引发该警告的方法或者构造器，Java9中增强了这种注解，允许使用该注解修饰私有方法。

@FunctionalInterface

Java8中引入了新特性叫函数式接口注解， java8中规定如果接口中只有一个抽象方法，那么该接口就是函数式接口。@FunctionalInterface就是用来指定某个接口必须是函数式接口，而且它只能用来修饰接口，不能修饰其它元素。

1. **Lambda表达式**

**Lambda表达式由3个部分组成：**

**·形参列表**

**·箭头（->）**

**·代码段**

1. **List/Set/Map接口基本用法P190-195**

import java.util.\*;

（1）List 的用法：继承了Collection接口，List接口是一个允许重复项的有序集合

（a）LinkList类

（b）ArrayList类

（2）Set的用法：继承了Collection接口，它的特性是存放在里面的元素是没有特定顺序的，并且元素不可以重复。（另：Set的一个子接口：SortedSet,放在这个接口中的数据是有序的）

（a）HashSet类

（b）TreeSet类

（c）SortedSet类 例子：

class A{

public static void main(String args[]){

Set set = new HashSet();//在主函数中使用；无序的

set.add(new Interger(5));

set.add(“haiery”)

System.out.println(set);

}

}

（3）Map的用法（键值对）：

（a）HashMap类的使用

（b）TreeMap类；

（c）添加：

HashMap hm = new HashMap();

hm.put(1,”A”);……………………

Object o = hm.get(2);//获得指定键为2所对应的值，Object 具有自动打包功能

String s = (String)o;

hm.remove(3)；//移除指定的键值

1. **泛型的使用：P196**

泛型类

public class GenericClass <T> {

private T t;

public void print(T t) {

System.out.println(t);

}

public static void main(String[] args) {

GenericClass<Integer> integervar = new GenericClass< >();

GenericClass<String> stringvar = new GenericClass< >();

integervar.print(new Integer(3));

stringvar.print(new String("three"));

}

}

泛型方法

public static <E> void genericMethods(E[] arrays) {

// 输出数组元素 < E >

for (E array : arrays) {

System.out.print(array + ",");

}

}

public static void main(String args[]) {

// 创建不同类型的数组，Integer和String类型

Integer[] intArray = { 1, 2, 3, 4, 5 };

String[] stringArray = { "one", "two", "three", "four", "five" };

System.out.println("整型数组元素为:");

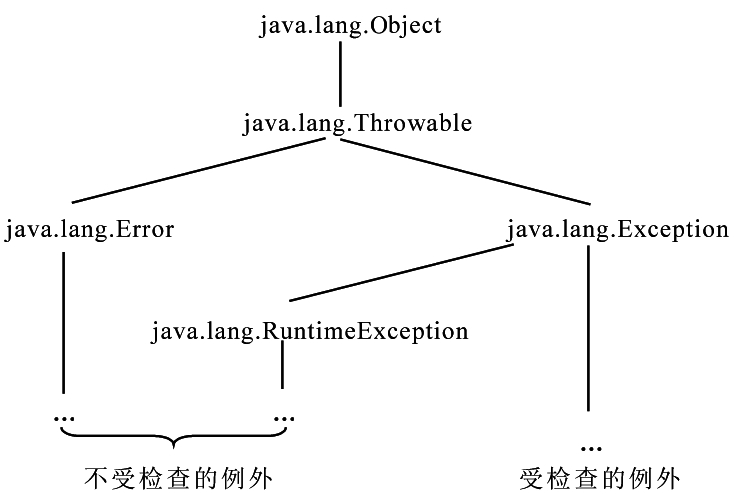
genericMethods(intArray); // 输出整型数组

System.out.println("\n字符串型数组元素为:");

genericMethods(stringArray); // 输出字符串型数组

}

1. **异常类的层次结构 P211**



1. **异常处理**

**处理异常分为以下两个步骤：**

**（1） 抛出异常**

**在程序运行时当语义规则被违反时，将会抛出(throw)异常，即产生一个异常事件，生成一个异常对象。**

**（2）捕获异常**

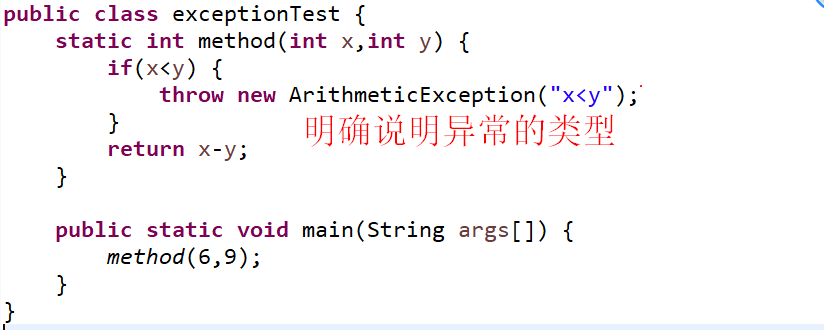
**异常抛出后，异常对象提交给运行系统，系统就会从产生异常的代码处开始，沿着方法调用栈进行查找，直到找到包含相应处理的方法代码，并把异常对象交给该方法进行处理，这个过程成为捕获(catch)异常。**

**Java的异常处理是通过5个关键词try、catch、throw、throws、finally来实现的。**

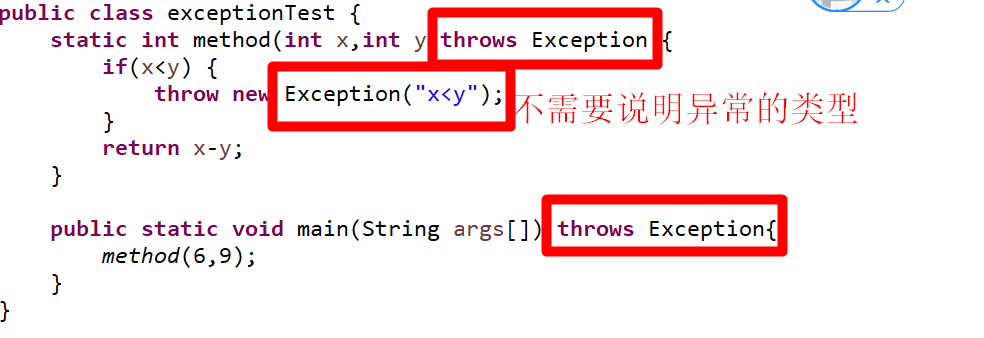
**如果程序运行中引发的异常是不受检查的异常，那么当异常产生时可以不进行处理，程序中断执行，并由运行系统调用默认的处理程序进行处理，但是如果引发的受检查的异常，那就必须进行处理或者声明抛出。**

1. **Throw也属于转移语句**

throw语句用于引发异常



throws语句用于抛出异常



1. **try/ catch/ finally语法P215-217,219**
2. try｛

//此处是可能发生异常的代码

throw new ArthmeticException();

｝catch(ArthmeticException e){

//异常的处理代码

}finally{

}

（2）finally：控制流不管以何种原因离开try语句，都要先执行finally子句（有对应的异常，先处理相应的异常，然后执行finally语句，退出try语句）

1. 各界面布局的主要特点P230-238。了解

图形界面设计的步骤：选择容器、确定布局、向容器中添加组件、进行事件处理

AWT的五种布局管理器：**FlowLayout、****BorderLayout、GridLayout、CardLayout、GridBagLayout。**

1. GUI容器、组件P239-244 了解

外部容器：Frame 默认布局：**BorderLayout**

内部容器：Panel 默认布局：**FlowLayout**

**AWT组件：标签（Label）、按钮（Button）、下拉式菜单（Choice）、文本框（TextField）、文本区（TextArea）、列表（List）、复选框（Checkbox）、复选框组（CheckboxGroup）、画布（Canvas）菜单（Menu）、对话框（Dialog）、文件对话框（FileDialog）**

**Swing组件：面板（JPanel）、滚动窗口（JScrollPane）、选项板（JTabbedPane）、工具栏（JToolBar）、按钮（JButton）、复选框（JCheckBox）、单选框（JRadioButton）、选择框（JComboBox）、标签（JLabel）、菜单（JMenu）、进度条（JProgressBar）、滑动条（JSlider）、表格（JTable）、树（JTree）、**

1. 事件监听器/适配器P252-256

ActionListener listener = new MyListener();

JButton btn = new JButton(“Hi”);

btn.addActionListener(listener);

class MyListener implementsActionListener{

public void actionPerformed(ActionEvent event){

//对按钮事件做响应

}

1. I/O常用类P268-279
2. 字节流和字符流的区别

如果数据被格式化为以字节（8位）为基本单位的数据流，称为字节流。

如果数据被格式化为以字符（16位）为基本单位的数据流，称为字符流。

字符流与字节流的主要区别在于，它是以字符作为数据流的基本单位，而不是以字节为单位

（二）常用输入类

1、FileInputStream类

2、ByteArrayInputStream类

3、PipedInputStream类

4、DataInputStream类

（三）常用输出类

1、FileOutputStream类

2、ByteArrayOutputStream类

3、PipedOutputStream类

4、DataOutputStream类

（四）对文件和目录的操作的个类

1、File类

2、RandomAccessFile类 P282

（五）对象的序列化操作

对象的序列化是指将实现了序列化接口（Serializable接口）的对象转化成字节序列进行保存或传输，而以后还能够根据该字节序列将对象完全还原。

1. **创建线程语法P295-297**

**1、通过扩展Thread类创建线程**

**我们可以通过扩展Thread类，并覆盖Thread类中的run ( )方法来创建一个线程。其中，run( )方法中的代码就是让线程完成的工作。**

**2、通过实现Runnable接口创建线程**

**除了通过扩展Thread类来创建线程外，也可以通过实现Runnable接口来创建一个线程。该接口只有一个run()方法需要实现。一个类实现Runnable接口后，如果要创建一个线程，需要先创建该类的一个实例，然后将该实例作为参数传递给Thread类的一个构造方法来创建一个Thread类实例。**

**启动一个线程应该调用start()方法，而不是直接调用run()方法。启动start()方法后，具体该线程何时执行以及分配多长时间执行，都交由操作系统去分配，而不是由程序员来控制。**

class threadTest extends Thread{}

----Thread T1 = new threadTest();

class threadTest2 implements Runnable{}

------Thread T2 = new Thread(new threadTest2());

1. **线程的生命周期P297-298**

新建状态、可运行状态、阻塞状态、死亡状态

1. **多线程同步基本语法P301 synchronized（同步） 与notify（唤醒下一个线程可进入临界区）**

例如：synchronized void booking(int num){}

1. Socket编程基本语法P310了解
2. UDP通信基本语法P317了解

**闭卷笔试，编程题不记得的方法，可以写中文伪码。**

一．选择题，共15题 （共计30分，每题2分）

二、判断题，共5题 （共计10分，每题2分）

三、填空题, 共5题 （共计20分，每题4分）

1. 读程序写结果, 共2题（共计10分，每题5分）

五、读程序填写正确语句(共2题，共10分)

**1. Java GUI事件处理程序。**

**2. StringTokenizer。(字符串标记)**

（1）字符串类有：String、StringBuffer 和StringTokenizer类

（2）String和StringBuffer位于java.lang包中

StringTokenizer 位于java.util包中

**3. I/O类型的编程**

**4.创建多线程编程**

**六、程序设计，共2题 （共计20分，每题10分）**

1. 数字类简单算法题。（注释写出算法思想和关键语句的注释）

2. 类、继承基本编程题（关键语句的注释）

3. 数字类算法（注释写出算法思想和关键语句的注释）

4. 接口、抽象类基本编程题（关键语句的注释）

七、附加题, 共1-2题（共计30分）

1. 多线程编程。（关键语句的注释）

2.集合/遍历。**（**关键语句的注释）(List ,Map,Set)

3.数字类算法（注释写出算法思想和关键语句的注释）

二、填空题：

第五章：

内部类可以分为 内部类 ， 局部内部类 ， 静态内部类 ， 匿名内部类 。

第六章：

2． 哪些接口存放的数据是有序的？

Map、Set

3． 哪些接口存放的数据可以重复？

Collection、List

第七章

3. 在java中是如何处理异常的？

答：在java程序执行过程中，如果出现了异常事件，就会发生一个异常对象，系统将捕获这个异常，并对其进行处理。如果没有相应的异常处理程序，程序将终止。

第八章

3、框架（Frame）和面板（Panel）的默认布局管理器是什么？

答：Frame使用的BorderLayout布局管理器，Panel使用的是FowLayout布局管理器

4、监听器和适配器的作用是什么？为什么要引入适配器？

答：监听器是监听某个事件执行某个操作的接口，适配器是实现这个接口的类。

由于适配器类实现了监听器的所有抽象方法，因此程序员通过继承适配器类，可以只覆盖感兴趣的方法，减少开发工作量。

第九章

2、下列表达式中，能够创建一个DataOutputStream流的表达式是（C）

A) new DataOutputStream(new Writer(“o.txt”));

B) new DataOutputStream(new OutputStream(“o.txt”));

C) new DataOutputStream(new FileOutputStream(“o.txt”));

D) new DataOutputStream(new FileWriter(“o.txt”));

4、什么叫对象的序列化操作？

答：对象的序列化是指将实现了序列化接口（Serializable接口）的对象转化成字节序列进行保存或传输，而以后还能够根据该字节序列将对象完全还原。

第十章

3、线程有哪几种状态？状态之间是如何转换的？

线程的新建状态（New Thread）、可运行状态（Runnable）、阻塞状态（Blocked）和死亡状态（Dead）。

当使用New关键字新创建一个线程时，线程处于新建状态（New Thread）；

一旦一个线程调用了start()方法，线程就进入可运行状态（Runnable）；

如果出现下述情况，线程进入阻塞状态：

1. 线程调用了sleep()方法；
2. I/O流中发生了线程阻塞；
3. 线程调用了wait()方法；
4. 线程调用了suspend()方法（已过时）；

线程要锁住一个对象，但该对象已被另一个线程锁住。

处于阻塞状态的线程在下述情况还会回到可运行状态：

处于阻塞状态的线程要想重新回到可运行状态，需要分别满足不同的特定条件：

1. 如果由于调用sleep()而进入阻塞状态，则当睡眠时间到达所规定的毫秒数，就可以离开该状态；
2. 如果由于等待I/O操作，那么当这个I/O操作完成后，就可以离开该状态；
3. 如果由于调用了wait()方法，那么只有等另一个线程调用了notify()或notifyAll()方法后，才能离开该状态；
4. 如果由于调用suspend()方法被挂起，则需调用resume()后才能离开该状态；
5. 如果正在等待其他线程拥有的对象锁，只有该线程放弃锁后，才能离开该状态。

如果出现下述情况，线程进入死亡状态：

1. 线程执行完，自然死亡。
2. 异常终止run()方法（如调用stop()方法，stop()方法已过时）。

4、什么叫“临界资源”？什么叫“临界区”？什么叫“同步方法”？

我们把一次只允许一个线程使用的资源称为“临界资源”，访问这种临界资源的代码称为“临界区”。 如果一个方法被关键字synchronized修饰，这个方法被称为“同步方法”。 同步方法在其执行期间是不会被中断的。

5、每个对象具有一把锁，我们将其称为“对象锁”，当线程没有访问同步方法时，对象锁的状态是打开的。如果线程要进入对象中的同步方法，即被synchronized修饰的实例方法，需要先检查其对象锁的状态，如果锁是打开的，则线程可以进入该同步方法，同时关闭对象所，此时线程持有该对象锁。如果此时其他线程也要访问该同步方法，就会进入阻塞状态，到对象的锁等待池中等待，直到前面的线程退出同步方法释放对象锁后，才能继续执行。

第十一章java网络编程

2、TCP/IP协议和UDP/IP协议的主要区别是什么？

答：TCP（Transfer Control Protocol）协议是面向连接的协议，保证传输的可靠性。发送方和接收方的Socket之间需要建立连接，以保证得到的是一个顺序、无差错的数据流。与TCP协议不同，UDP（User Datagram Protocol）协议是一种无连接协议，因此每个数据报向目的地传送的路径并不固定，它可能通过任何可能的路径到达目的地。至于每个数据报是否能最终到达以及内容的正确性都是无法保证的。

3、使用Socket编程的基本步骤是什么？

使用Socket编程的基本步骤包括下面3个方面：

1）创建Socket。

2）打开连接到Socket上的I/O流，遵照某种协议对Socket进行读/写操作。

3）关闭Socket。