**1.（P7）java程序的开发过程；**

编写源文件（使用编辑器）编译文件（产生）字节码（使用解释码）执行字节码

1. 编写源文件：*源文件的扩展名必须是java*
2. 编译源文件：使用*java编译器（javac.exe）*编译源文件，得到*字节码文件.class*
3. 运行程序：使用javaSE平台中的*Java解释器（Java.exe）*来执行字节码文件。

**2.(P16)第一章课后题**

⑴JDK提供的编译器是*java.exe*

⑵Java应用程序主类中正确的main方法是*public static void main(String args[ ])*

**3.(P19)标识符和关键字**

标识符的命名规则（**选择题：判断标识符是否合法★**）

1. *标识符由字母、下划线、美元符号和数字组成，长度不受限制*
2. 标识符的*第一个字符不能是数字字符*
3. 标识符*不能是关键字*
4. *标识符不能是true、false和null*（尽管true、false和null不是Java关键字）

关键字

关键字是Java中已经被*赋予特定意义的一些单词*，*不可以把关键字当做标识符来用*。

**4.(P20)基本数据类型**

基本数据类型也成为简单数据类型。Java中有8种。分为4类：

1. 逻辑类型：boolean
2. 整数类型：byte、short、int、long
3. 字符类型：char
4. 浮点类型：float、double

补充：枚举类型 menu

**5.(P22)浮点类型**

浮点型 float型和double型

float 类型的数字若写浮点数后必须*加上小写f作为区别的符号，*但是*写成整数可不加*。

Double类型的数字在*定义时可以是浮点数也可以是整数*，但是*输出时必须是浮点数*。

**6.(P28)数组**

注：与C/C++不同，Java不允许在声明数组中的方括号内指定数组元素的个数。若声int a[5]或者int [12]a时，将导致语法错误。

数组的创建

1.动态初始化格式：  
数据类型 [ ] 数组名称 = new 数据类型[数组长度]  
2.静态初始化格式：  
数据类型 [ ] 数组名称 = new 数据类型[ ]{元素1，元素2，元素3…}  
3.静态初始化省略格式：  
数据类型 [ ] 数组名称 = {元素1，元素2，元素3…}

数组的赋值，直接有下标的引用方法如：a[0]=6;

**7.(P38)第二章课后题**

⑴ float变量的声明正确的是 *float foo = 1*

⑵ float型变量的声明正确的是 *float foo =0x123*

⑶ char型变量的声明正确的是 *char ch = ‘ \\ ’*

**8.(P41)自增、自减运算符**

int a=1,b=2;

++a; *//因为++在前所以先进行+1的操作,再使用变量*

++b;*//b=b+1;b;*

a++;*//因为变量在前，所以先使用，再运算*

b++;*//相当于b;b=b+1;*

**9.(P53)break和continue语句**

break和continue语句是用关键字break或continue加上分号构成的语句。在循环体重可以使用break语句和continue语句，*在一个循环中，如果在某次循环中执行break语句，那么整个循环语句就结束了*。*如果在某次循环中执行了continue语句，那么本次循环就结束了*，即不再执行本次循环中循环体的continue语句后面的语句，而转入进行下一次循环。*总的来说continue和break执行了之后都不会执行循环内其后的语句。*

**10.(P73)构造方法（判断☆）**

需要特别注意，*构造方法没有类型，*并且*方法名和类名完全相同*。

Class Point{

Int x；

Point（）{… }*//是构造方法*

Point（int a）{… }*//是构造方法，带参数*

void Point（）{… }*//不是构造方法（该方法的类型是void）*

、}

**11.(P97)包**

包语句：

*如果源文件有package语句，package语句必须作为Java源文件的第一条语句*（源文件至多可以有一条package语句），指明该源文件定义的类所在的包，即为该源文件中声明的类知名包名。Package语句一般的格式为:

Package 包名;

*如果源程序中 省略了package语句，源文件所定义命名的类被隐含地认为是无名包的一部分*。

***类中的package命令不是非得放在类的第一行，其上可以写注释和空行，但是对于import、类、变量等命令，则一定不能放在package命令之上。***

**12.(P99)包**

引入类库的类：为了使用java提供的类，可以使用import语句引入包中两类。*在一个Java源程序中可以有多个import语句（加入有package语句），它们必须写在package语句和源文件中类的定义之间*。

**13.(P167)自定义异常类（程序填空☆☆☆☆）**

throw是Java的关键字，该关键字的作用就是抛出异常，throw和throws是两个不同的关键字。

*相似的例题：*

1. **定义异常消息类：怎么操作呢？？？**

**IntegerException.java**

//定义了IntegerException类，必须继承Exception类，因为他是异常的父类

**public** **class** IntegerException **extends** Exception {

String message;*//先定义错误提示*

**public** IntegerException(**int** m) {//IntegerException的含参构造方法

message = "年龄"+m+"不合理";*//自定义文本，注意若加上自定义的提示文本需要字符串拼接*

}

**public** String toString() {//定义了输出字符串的函数，并规定返回值为字符串

**return** message;

}

}

1. **定义bean类People，定义了一个私有方法age，进行赋值和获取值**

**People.java**

**public** **class** People {

**private** **int** age = 1;

//定义的People类中含有一个私有的属性age

**public** **void** setAge(**int** age) **throws** IntegerException{

*//在进行set赋值的时候抛出异常，注意抛出异常使用throws*

**if**(age>=160||age<=0) {

//异常的判断，自定义的条件，如果age大于等于160或者小于等于0，就执行下面的语句，*注意||的短路特性*

**throw** **new** IntegerException(age);

*//上面的语句是抛出异常，并且由IntegetException的方法体内语句执行处理*

}**else** {

*//若无异常正常赋值*

**this**.age=age;

}

}

**public** **int** getAge() {

//进行get获取，下面进行了一个自定义的print输出

System.***out***.println("年龄"+age+"合理");

**return** age;

}

}

1. **定义一个主类，用来实例化People 的对象**

**Example.java**

**public** **class** Example {

**public** **static** **void** main(String args[]) {

//实例化People类对象，因为*实例化了两个wang，zhang可以同时实例化，使用逗号隔开*

People wang = **new** People(),

zhang = **new** People();

//下面使用了try...catch异常的捕获

**try** {

//首先对对象wang的age进行赋值，在我们前面的定义中可以知道180是会报告异常的

wang.setAge(180);

//因为上面180是异常的，所以会抛出异常，不会执行下面的print语句

//因为在赋值的时候180是会报告异常，*直接执行 throw new IntegerException(age)语句*

System.***out***.println(wang.getAge());

}**catch**(IntegerException e){

*//因为上面捕捉到了异常，所以要执行catch语句，输出了上面设置的message*

*//所以输出 年龄180不合理*

System.***out***.println(e.toString());

}

**try** {

//对对象zhang的age进行赋值，*在我们前面的定义中可以知道37是不会捕捉的错误的，会执行this.age=age正常赋值*

zhang.setAge(37);

//输出获取的上面设置的age，因为return是不会主动输出的，所以使用了print输出

System.***out***.println(zhang.getAge());

//因为调用了getAge()方法，先输出了定义的print，然后输出一个message

}**catch**(IntegerException e){

*//因为上面没有发现错误，所以catch语句不会被执行*

System.***out***.println(e.toString());

}

}

}

**运行截图**

文本

中度可信度描述已自动生成

**14.(补充)throw关键字和throws关键字的区别**

throw和throws作为Java中两种异常抛出关键字，虽然两个长的很像，但是却有着很大的区别。

1. 区别1:

throws:跟在方法声明后面，后面跟的是异常类名

throw:用在方法体内，后面跟的是异常类对象名

1. 区别2:

throws:可以跟多个异常类名，用逗号隔开

throw:只能抛出一个异常对象名

1. 区别3:

throws:表示抛出异常，由该方法的调用者来处理

throw:表示抛出异常，由该方法体内的语句来处理

1. 区别4:

throws: throws表示有出现异常的可能性，并不一定出现这些异常

throw: throw则是抛出了异常，执行throw一定出现了某种异常

**15.(P179)根据UML图写代码（*注意看我上课实验文件夹里面的 11.3的文件*）**

可以查看下面的三个网址

1. <https://www.cnblogs.com/gaochundong/p/uml_class_diagram_notation.html>
2. <https://kb.cnblogs.com/page/129490/>
3. <https://blog.csdn.net/Return_head/article/details/88632366?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163935717116780261913033%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=163935717116780261913033&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-3-88632366.pc_search_result_control_group&utm_term=%E6%A0%B9%E6%8D%AEUML%E5%9B%BE%E7%9A%84%E5%85%83%E7%B4%A0%E8%A1%A8%E7%A4%BA&spm=1018.2226.3001.4187>

**16.(P188) 类行为定义--“高内聚-低耦合”原则（*这句话一定会考*）**

如果类中的方法是一组相关的行为，则称该类是*高内聚*的，反之成为低内聚。高内聚便于类的维护，而低内聚不利于类的维护。*低耦合*就是尽量不要让一个类含有太多的其他类的实例的引用。

**17.(P335) 文件字符流和字节流**

图形用户界面, 文本

描述已自动生成

*仔细观看下面的网址：*

<https://blog.csdn.net/qq_21491877/article/details/88943939?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163935794716780271940339%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request_id=163935794716780271940339&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~first_rank_ecpm_v1~rank_v31_ecpm-4-88943939.pc_search_result_control_group&utm_term=%E6%96%87%E4%BB%B6%E5%AD%97%E7%AC%A6%E6%B5%81&spm=1018.2226.3001.4187>

**18.(P363) 链表**

⑴ ArrayList<String> list = **new** ArrayList<String>();

1. LinkedList<String> link= **new** LinkedList<String>();
2. HashSet<String> set= **new** HashSet<String>();
3. HashSet<Student> stu =**new** HashSet<Student>();

***代码看文件夹链表***

注意几个函数的使用：

Public boolean add(E element) 向链表末尾加入新结点，参数是指定数据

Public void add(int index,E element) 向链表的指定位置加入新结点

Public void clear( ) 删除链表的所有结点，使当前链表成为空链表

Public E remove(int index) 删除指定位置上的结点

Public boolean remove(E element) 删除指定位置上的节点

Public E get(int index) 得到链表中指定位置处结点中的数据

参考网站：

<https://blog.csdn.net/yutian_1999/article/details/104238092?ops_request_misc=%257B%2522request%255Fid%2522%253A%2522163935815416780357226959%2522%252C%2522scm%2522%253A%252220140713.130102334..%2522%257D&request_id=163935815416780357226959&biz_id=0&utm_medium=distribute.pc_search_result.none-task-blog-2~all~top_positive~default-1-104238092.pc_search_result_control_group&utm_term=java%E9%93%BE%E8%A1%A8&spm=1018.2226.3001.4187>

**19.(P390) JDBC**

1）下载JDBC-MySQL数据库连接器

2）加载JDBC-MySQL数据库连接器

**import** java.sql.Connection;

**import** java.sql.DriverManager;

**public** **static** Connection getCon()

{

Connection con = **null**;

**try**{

//加载JDBC驱动。

Class.*forName*(*jdbcName*);

//创建连接数据库的对象con

con = DriverManager.*getConnection*(*dburl*,*dbUserName*,*dbPassword*);

}**catch** (ClassNotFoundException e) {

System.***out***.println("数据库驱动错误！");

e.printStackTrace();

} **catch** (SQLException e) {

System.***out***.println("获取数据库连接异常！");

e.printStackTrace();

}

**return** con;

}

3）连接数据库

//URL指向要访问的数据库名

**private** **static** String *dburl* = "jdbc:mysql://localhost:3306/test?useUnicode=true&characterEncoding=utf8";

//连接数据库的用户名

**private** **static** String *dbUserName* = "root";

//MySQL数据库用户的密码

**private** **static** String *dbPassword* ="123456";

//驱动程序名

**private** **static** String *jdbcName* = "com.mysql.jdbc.Driver";

con = DriverManager.*getConnection*(*dburl*,*dbUserName*,*dbPassword*);