JavaEE：

1. Java基础阶段（java、MySQL、JDBC）
2. JavaWeb阶段（HTML、CSS、JavaScript、JQuery、Tomcat、XML、Servlet、JSP、。。。）
3. Java框架阶段(Spring、SpringMVC、Mybatis、Maven）
4. 项目一：众筹项目（SVN、微服务框架）
5. Java高级技术：Git、Linux、MySQL高级、Docker、JVM、Tomcat高级、JUC、Dubbo、。。。
6. 项目二：电商
7. 实战
8. 就业阶段

大数据：

1. Java基础阶段（java、MySQL、JDBC）
2. JavaEE阶段（JavaWeb + SSM + Git + Linux）
3. Hadoop阶段
4. Spark阶段
5. 项目阶段（丰富的项目）
6. Flink阶段+项目

Java基础课程概述

第一部分：编程语言核心结构
主要知识点：变量、基本语法、分支、循环、数组、...

第二部分：Java面向对象的核心逻辑
主要知识点：OOP、封装、继承、多态、接口、...

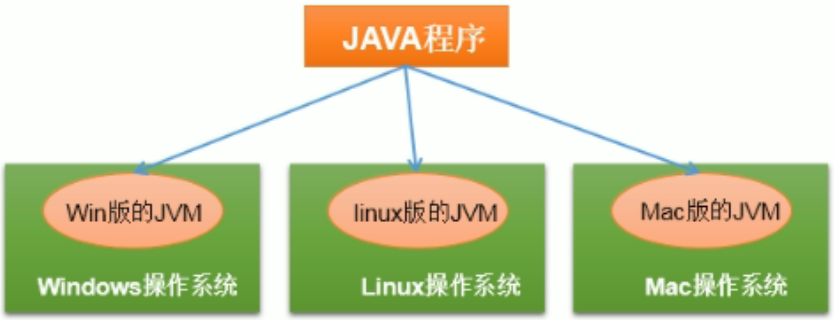
第三部分：开发Java SE高级应用程序
主要知识点：异常、集合、I/O、多线程、反射机制、网络编程、.....

第四部分：实训项目
项目一：家庭收支记账软件
项目二：客户信息管理软件
项目三：开发团队人员调度软件
附加项目一：银行业务管理软件
附件项目二：单机考试管理软件

Java语言的特点

- 特点一：面向对象
  - ✓两个基本概念：类、对象
  - ✓三大特性：封装、继承、多态
- 特点二：健壮性
  - ✓吸收了C/C++语言的优点，但去掉了其影响程序健壮性的部分（如指针、内存的申请与释放等），提供了一个相对安全的内存管理和访问机制
- 特点三：跨平台性
  - ✓跨平台性：通过Java语言编写的应用程序在不同的系统平台上都可以运行。**“Write once , Run Anywhere”**
  - ✓原理：只要在需要运行 java 应用程序的操作系统上，先安装一个Java虚拟机 (JVM Java Virtual Machine) 即可。由JVM来负责Java程序在该系统中的运行。

Java语言的特点：跨平台性



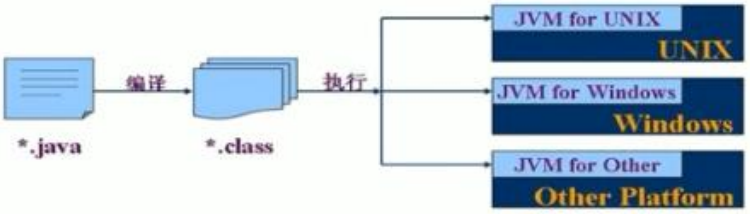
因为有了JVM，同一个Java 程序在三个不同的操作系统中都可以执行。这样就实现了Java 程序的跨平台性。

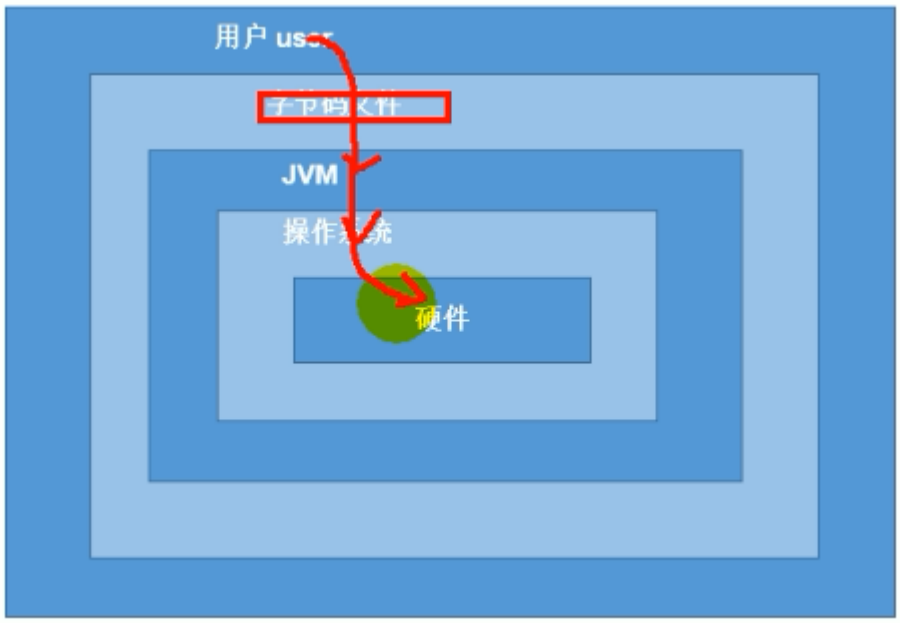
Java两种核心机制

- Java虚拟机（Java Virtal Machine）
- 垃圾收集机制（Garbage Collection）

核心机制—Java虚拟机

- **JVM是一个虚拟的计算机，具有指令集并使用不同的存储区域。负责执行指令，管理数据、内存、寄存器。**
- 对于不同的平台，有不同的虚拟机。
- 只有某平台提供了对应的java虚拟机，java程序才可在此平台运行
- Java虚拟机机制屏蔽了底层运行平台的差别，实现了“一次编译，到处运行”





核心机制—垃圾回收

- 不再使用的内存空间应回收—— 垃圾回收。
  - 在C/C++等语言中，由程序员负责回收无用内存。
  - Java 语言消除了程序员回收无用内存空间的责任：它提供一种系统级线程跟踪存储空间的分配情况。并在JVM空闲时，检查并释放那些可被释放的存储空间。
- 垃圾回收在Java程序运行过程中自动进行，程序员无法精确控制和干预。

- Java程序还会出现内存泄漏和内存溢出问题吗？ Yes!

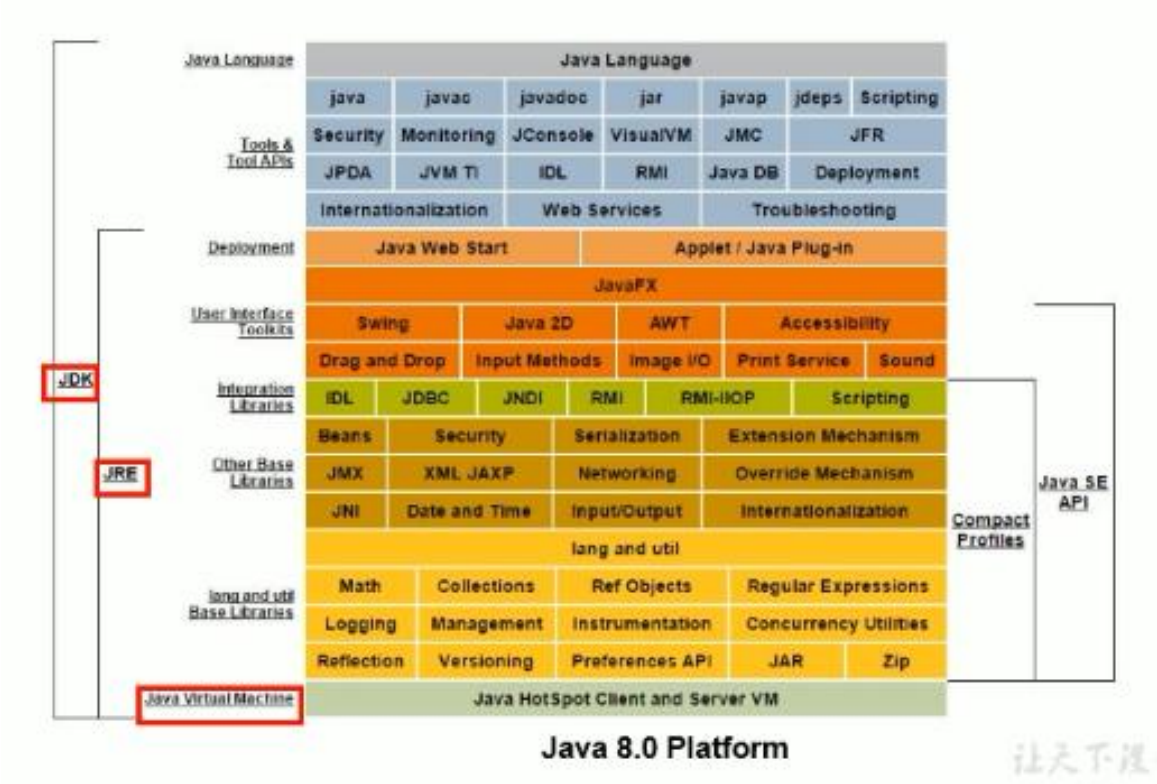
栈溢出!!! 回收不了

内存没回收也不用 导致泄漏和溢出

什么是JDK，JRE

JDK(Java Development Kit　Java开发工具包)
JDK是提供给Java开发人员使用的，其中包含了java的开发工具，也包括了JRE。所以安装了JDK，就不用在单独安装JRE了。 ➢ 其中的开发工具：编译工具(javac.exe) 打包工具(jar.exe)等
JRE(Java Runtime Environment　Java运行环境)
包括Java虚拟机(JVM Java Virtual Machine)和Java程序所需的核心类库等，如果想要运行一个开发好的Java程序，计算机中只需要安装JRE即可。

简单而言，使用JDK的开发工具完成的java程序，交给JRE去运行。



Java 8.0 Platform

1.3 Java语言的环境搭建：下载并安装JDK

- 官方网址：
  - www.oracle.com
  - java.sun.com
- 安装JDK
  - 傻瓜式安装，下一步即可。
  - 建议：安装路径不要有中文或者特殊符号如空格等。
  - 当提示安装 JRE 时，可以选择不安装。

配置环境变量 path

●在dos命令行中敲入javac，出现错误提示：

```
D:\developer_tools\Java\jdk1.8.0_131>javac
'javac' 不是内部或外部命令，也不是可运行的程序
或批处理文件。
```

●错误原因：当前执行的程序在当前目录下如果不存在，windows系统会在系统中已有的一个名为path的环境变量指定的目录中查找。如果仍未找到，会出现以上的错误提示。所以进入到 jdk安装路径\bin目录下，执行javac，会看到javac参数提示信息。

```
D:\developer_tools\Java\jdk1.8.0_131\bin>javac
用法: javac <options> <source files>
其中, 可能的选项包括:
  -g                         生成所有调试信息
```

●明确什么是JDK, JRE

●下载 JDK

●安装 JDK

●配置环境变量

➢ **path**：windows系统执行命令时要搜寻的路径。

●验证是否成功：javac　java

●选择合适的文本编辑器或 IDE 开发

1.4 开发体验 — HelloWorld

●步骤：

1. 将 Java 代码编写到扩展名为 .java 的文件中。
2. 通过 javac 命令对该 java 文件进行编译。
3. 通过 java 命令对生成的 class 文件进行运行。

●用于注解说明解释程序的文字就是注释。

●Java中的注释类型：

- 单行注释
- 多行注释
- 文档注释 (java特有)

●提高了代码的阅读性；调试程序的重要方法。

●注释是一个程序员必须要具有的良好编程习惯。

●将自己的思想通过注释先整理出来，再用代码去体现

```
1
2  /*
3  多行注释：
4
5
6  1. java 程序分为三种注释方式：单行注释、多行注释、文档注释
7  2. 单行注释和多行注释，在使用javac.exe命令进行编译时，不会参与编译，进而不会出现在.class字节码文件中
8  3. 单行注释、多行注释的作用：① 增加程序的可读性 ② 协助调试程序
9  */
10
11 class HelloJava{
12     /*
13     如下是main()方法，作为程序的入口。
14     格式是固定的！
15     */
16     public static void main(String[] args){
17         //单行注释：如下是程序的输出语句。
18         //输出格式是：System.out.print()
19         System.out.print("hello world!");
20     }
21 }
```

```
/*
多行注释：


1. java 程序分为三种注释方式：单行注释、多行注释、文档注释
2. 单行注释和多行注释，在使用javac.exe命令进行编译时，不会参与编译，进而不会出现在.class字节码文件中
3. 单行注释、多行注释的作用：① 增加程序的可读性 ② 协助调试程序
4. 多行注释使用时的注意点：多个多行注释不能嵌套使用。
*/

```

- 文档注释（java特有）
  - 格式：/**
    @author 指定java程序的作者
    @version 指定源文件的版本

    */
- 注释内容可以被JDK提供的工具 javadoc 所解析，生成一套以网页文件形式体现的该程序的说明文档。
- 操作方式

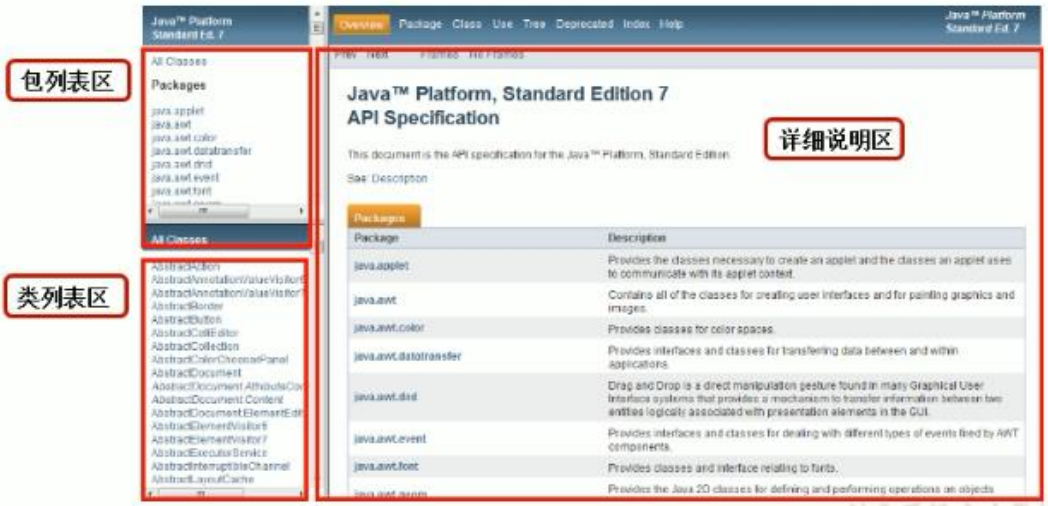
```
D:\javase\code\unit1>javadoc -d mydoc -author -version HelloWorld.java
```

```
/**
文档注释

1.作用：注释内容可以被JDK提供的工具 javadoc 所解析，生成一套以网页文件形式体现的该程序的说明文档。


@author shkstart
@version v1.0

这是我的第一个java程序。非常开心。么么~~


*/

```

- API （Application Programming Interface,应用程序编程接口）是 Java 提供的基本编程接口。
- Java语言提供了大量的基础类，因此 Oracle 也为这些基础类提供了相应的API文档，用于告诉开发者如何使用这些类，以及这些类里包含的方法。
- 下载API：

  http://www.oracle.com/technetwork/java/javase/downloads/index.html
  - Additional Resources-Java SE 8 Documentation下载。
- 详见：JDK8的下载-安装-配置.doc

1.7 Java API的文档



1.8 良好的编程风格

- **正确的注释和注释风格**
  - 使用文档注释来注释整个类或整个方法。
  - 如果注释方法中的某一个步骤，使用单行或多行注释。
- **正确的缩进和空白**
  - 使用一次tab操作，实现缩进
  - 运算符两边习惯性各加一个空格。比如：2 + 4 * 5。
- **块的风格**
  - Java API 源代码选择了行尾风格

```
public class Test {
    public static void main(String[] args){
        System.out.println("Block Style!");
    }
}
```

行尾风格

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Block Style!");
    }
}
```

次行风格

1.9 常用的Java开发工具

- 文本编辑工具：
  记事本　　UltraEdit　　EditPlus
  TextPad　　NotePad
- Java集成开发环境（IDE）：
  IntelliJ IDEA　　Eclipse
  MyEclipse　　NetBeans

EditPlus：
如果正确配置Java的编译器“Javac”以及解释器“Java”后，可直接使用EditPlus编译执行Java程序

Eclipse：
一个开放源代码的、基于Java的可扩展开发平台。

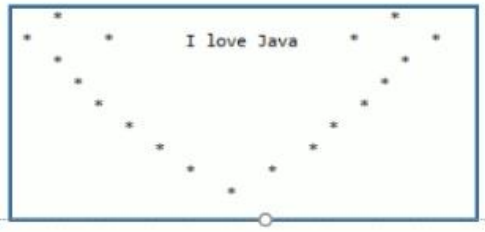
IntelliJ IDEA：
在代码自动提示、代码分析等方面的具有很好的功能。

NetBeans：
开放源码的Java集成开发环境，适用于各种客户机和Web应用。

MyEclipse：
由Genuitec公司开发的一款商业化软件，是应用比较广泛的Java应用程序集成开发环境。

作业

1. 独立编写HelloJava程序，并配上必要的注释。
2. 将个人的基本信息（姓名、性别、籍贯、住址）打印到控制台上输出。各条信息分别占一行。
3. 结合\n(换行)，\t(制表符)，空格等在控制台打印出如下图所示的效果。



Java基础是学习JavaEE、大数据、Android开发的基石！

举例：Spring – Rest(Spring MVC)

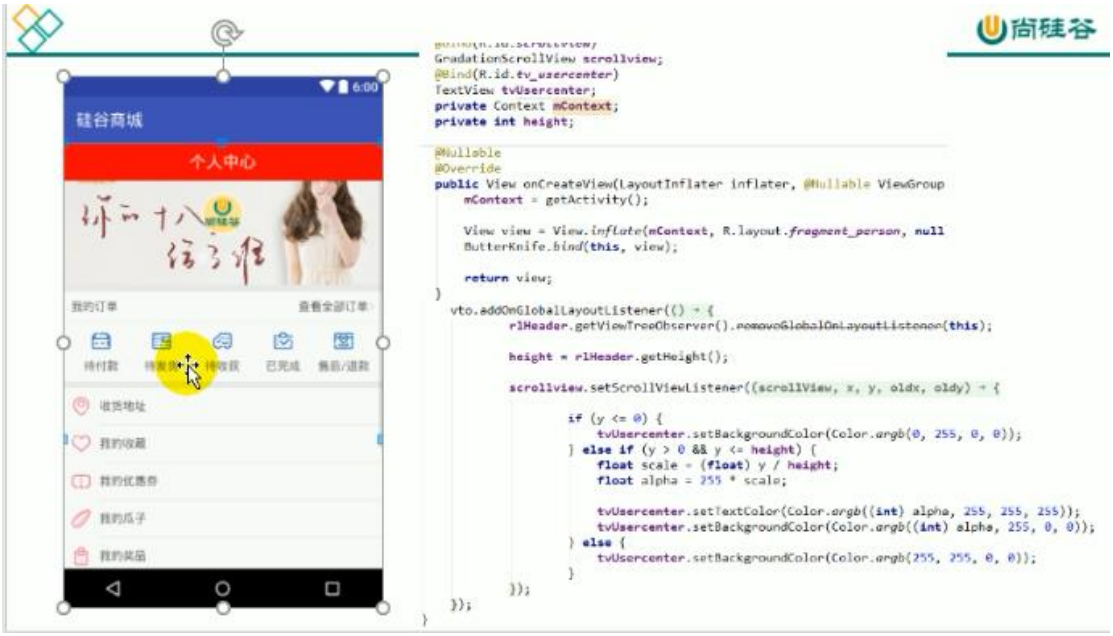
核心代码：

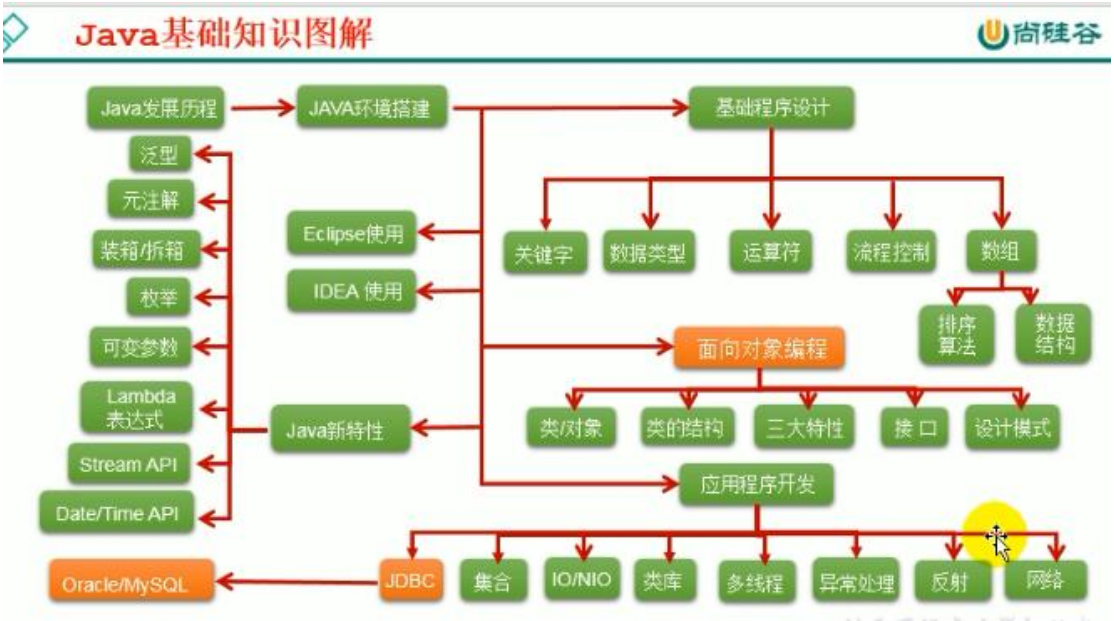
```
public  Model getPieData(Model model) {
    List<Count> counts = countRepository.findAll();
    JSONArray result = new JSONArray();
    for (Count count:counts) {
        JSONObject o = new JSONObject();
        o.put("name",count.getProvince());
        o.put("value",count.getSum());
        result.add(o);
    }
    model.addAttribute("data",result);
    return model;
}
```

举例：Spark – Spark Streaming

核心代码：

```
public void call(Iterator<ConsumerRecord<String, String>> consumerRecords) {
    KafkaProducer<String,String> kafkaproducer = new KafkaProducer<>(props);
    while(consumerRecords.hasNext()){
        ConsumerRecord<String,String> record = consumerRecords.next();
        String[] params = record.value().split(regex: ",");
        String geo = "["+params[params.length-2] +","+params[params.length-1]+"]";
        kafkaproducer.send(new ProducerRecord<String, String>(topicBroadcast.getValue(), geo));
        System.out.println(geo);
    }
    kafkaproducer.close();
}
```


尚硅谷
硅谷商城
个人中心
我的订单
查看全部订单
待付款
已完成
售后/退款
收货地址
我的收藏
我的优惠券
我的瓜子
我的奖品
GradationScrollView scrollview;
@Bind(R.id.tv_usercenter)
TextView tvUsercenter;
private Context mContext;
private int height;
@Nullable
@Override
public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup
mContext = getActivity();
View view = View.inflate(mContext, R.layout.fragment_person, null
ButterKnife.bind(this, view);
return view;
}
vto.addOnGlobalLayoutListener(() -> {
rlHeader.getViewTreeObserver().removeGlobalOnLayoutListener(this);
height = rlHeader.getHeight();
scrollview.setScrollViewListener((scrollView, x, y, oldx, oldy) -> {
if (y <= 0) {
tvUsercenter.setBackgroundColor(Color.argb(0, 255, 0, 0));
} else if (y > 0 && y <= height) {
float scale = (float) y / height;
float alpha = 255 * scale;
tvUsercenter.setTextColor(Color.argb((int) alpha, 255, 255, 255));
tvUsercenter.setBackgroundColor(Color.argb((int) alpha, 255, 0, 0));
} else {
tvUsercenter.setBackgroundColor(Color.argb(255, 255, 0, 0));
}
});
});
}


Java基础知识图解
尚硅谷
Java发展历程
JAVA环境搭建
基础程序设计
泛型
元注解
装箱/拆箱
枚举
可变参数
Lambda 表达式
Stream API
Date/Time API
Eclipse使用
IDEA 使用
Java新特性
关键字
数据类型
运算符
流程控制
数组
排序算法
数据结构
面向对象编程
类/对象
类的结构
三大特性
接 口
设计模式
应用程序开发
Oracle/MySQL
JDBC
集合
IO/NIO
类库
多线程
异常处理
反射
网络

Java基础课程概述

第一部分：编程语言核心结构
主要知识点：变量、基本语法、分支、循环、数组、...

第二部分：Java面向对象的核心逻辑
主要知识点：OOP、封装、继承、多态、接口、...

第三部分：开发Java SE高级应用程序
主要知识点：异常、集合、I/O、多线程、反射机制、网络编程、.....

第四部分：实训项目
项目一：家庭收支记账软件
项目二：客户信息管理软件
项目三：开发团队人员调度软件
附加项目一：银行业务管理软件
附件项目二：单机考试管理软件

Java基础课程体系

- 第1章 Java语言概述
- 第2章 基本语法
- 第3章 数组
- 第4章 面向对象编程(上)
- 第5章 面向对象编程(中)
- 第6章 面向对象编程(下)
- 第7章 异常处理
- 第8章 枚举类&注解
- 第9章 Java集合
- 第10章 泛型
- 第11章 IO流
- 第12章 多线程
- 第13章 Java常用类
- 第14章 Java反射机制
- 第15章 网络编程
- 第16章 Lambda表达式与Stream API
- 第17章 Java 9 & 10 新特性

2.1 关键字与保留字

- 关键字(keyword)的定义和特点
  - 定义：被Java语言赋予了特殊含义，用做专门用途的字符串（单词）
  - 特点：关键字中所有字母都为小写
  - 官方地址：https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

用于定义数据类型的关键字				
class	interface	enum	byte	short
int	long	float	double	char
boolean	void			
用于定义流程控制的关键字				
if	else	switch	case	default
while	do	for	break	continue
return				
用于定义访问权限修饰符的关键字				
private	protected	public		

2.1 关键字与保留字

用于定义类，函数，变量修饰符的关键字				
abstract	final	static	synchronized	
用于定义类与类之间关系的关键字				
extends	implements			
用于定义建立实例及引用实例，判断实例的关键字				
new	this	super	instanceof	
用于异常处理的关键字				
try	catch	finally	throw	throws
用于包的关键字				
package	import			
其他修饰符关键字				
native	strictfp	transient	volatile	assert
* 用于定义数据类型值的字面值				
true	false	null		

保留字(reserved word)

- Java保留字：现有Java版本尚未使用，但以后版本可能会作为关键字使用。自己命名标识符时要避免使用这些保留字

  goto 、const

2.2 标识符(Identifier)

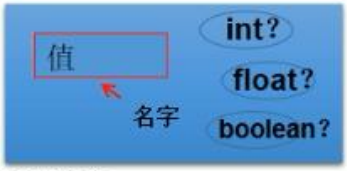
尚硅谷

- 标识符：
  - Java 对各种变量、方法和类等要素命名时使用的字符序列称为标识符
  - 技巧：凡是自己可以起名字的地方都叫标识符。
- 定义合法标识符规则：
  - 由26个英文字母大小写，0-9 ，_或 $ 组成
  - 数字不可以开头。
  - 不可以使用关键字和保留字，但能包含关键字和保留字。
  - Java中严格区分大小写，长度无限制。
  - 标识符不能包含空格。

练习：miles, Test, a++, --a, 4#R, $4, #44, apps, class, public, int, x, y, radius

```
/*
标识符的使用：

1.理解：凡是自己可以起名字的地方都叫标识符。比如：类名、变量名、方法名、接口名、包名...

2.标识符命名的规则。----> 必须严格遵守。否则，编译不通过！！
> 由26个英文字母大小写，0-9 ，_或 $ 组成
> 数字不可以开头。
> 不可以使用关键字和保留字，但能包含关键字和保留字。
> Java中严格区分大小写，长度无限制。
> 标识符不能包含空格。


3.标识符的命名规范：
> 包名：多单词组成时所有字母都小写：xxxyyyzzz
> 类名、接口名：多单词组成时，所有单词的首字母大写：XxxYyyZzz
> 变量名、方法名：多单词组成时，第一个单词首字母小写，第二个单词开始每个单词首字母大写：xxxYyyZzz
> 常量名：所有字母都大写。多单词时每个单词用下划线连接：XXX_YYY_ZZZ
```

2.3 变 量



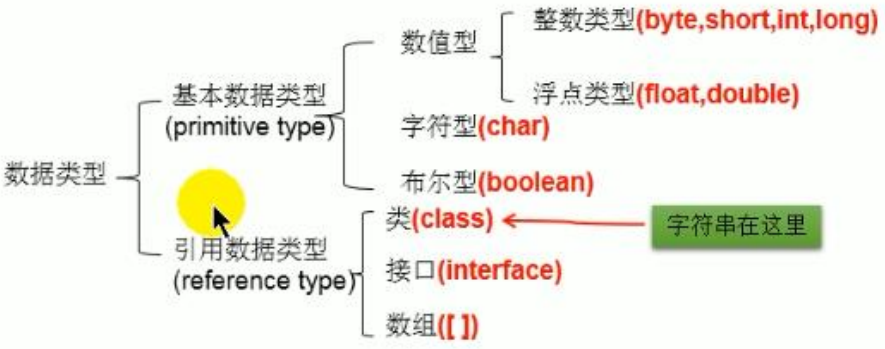
- 变量的概念：
  - 内存中的一个存储区域
  - 该区域的数据可以在同一类型范围内不断变化
  - 变量是程序中最基本的存储单元。包含变量类型、变量名和存储的值
- 变量的作用：
  - 用于在内存中保存数据
- 使用变量注意：
  - Java中每个变量必须先声明，后使用
  - 初始化值
  - 变量的作用域：一对{}之间有效
  - 同一个作用域内，不能定义重名的变量
  - 使用变量名来访问这块区域的数据

```
1
2  /*
3  变量的使用
4  1. 变量的定义：数据类型  变量名 = 变量值
5
6  2. 变量使用的注意点：
7  ① Java中每个变量必须先声明，后使用
8  ② 变量只在其定义的一对{}内有效。
9  ③ 不能在同一个作用域内定义同名的变量
10 */
```

2.3 变 量

变量的分类-按数据类型

- 对于每一种数据都定义了明确的具体数据类型（强类型语言），在内存中分配了不同大小的内存空间。



```
1.分类方式一：按照数据类型区分

1.1 基本数据类型：(8种)
        整型：byte \ short \ int \ long
        浮点型：float \ double
        字符型：char
        布尔型：boolean


1.2 引用数据类型
        类(包含String)、接口、数组


2.分类方式二：按照声明的位置的不同 （了解）
    成员变量(属性) vs 局部变量


*/
```

2.3 变 量

补充：变量的分类-按声明的位置的不同

- 在方法体外，类体内声明的变量称为**成员变量**。
- 在方法体内部声明的变量称为**局部变量**。

- 注意：二者在初始化值方面的异同：
  同：都有生命周期　　异：局部变量除形参外，需显式初始化。

2.3 变 量

整数类型：byte、short、int、long

- Java各整数类型有固定的表数范围和字段长度，不受具体OS的影响，以保证java程序的可移植性。
- java的整型常量默认为 int 型，声明long型常量须后加‘l’或‘L’
- java程序中变量通常声明为int型，除非不足以表示较大的数，才使用long

类 型	占用存储空间	表数范围
byte	1字节=8bit位	-128 ~ 127
short	2字节	-2^{15} ~2^{15}-1
int	4字节	-2^{31} ~ 2^{31}-1 (约21亿)
long	8字节	-2^{63} ~ 2^{63}-1

500MB 1MB = 1024KB 1KB= 1024B B= byte ? bit?
bit: 计算机中的最小存储单位。byte:计算机中基本存储单元。

浮点类型：float、double

- 与整数类型类似，Java 浮点类型也有固定的表数范围和字段长度，不受具体操作系统的影响。
- 浮点型常量有两种表示形式：
  - 十进制数形式：如：5.12 512.0f .512 (必须有小数点)
  - 科学计数法形式:如：5.12e2 512E2 100E-2
- float:单精度，尾数可以精确到7位有效数字。很多情况下，精度很难满足需求。
  double:双精度，精度是float的两倍。通常采用此类型。
- **Java 的浮点型常量默认为double型，声明float型常量，须后加‘f’或‘F’。**

类 型	占用存储空间	表数范围
单精度float	4字节	-3.403E38 ~ 3.403E38
双精度double	8字节	-1.798E308 ~ 1.798E308

```
//1.byte范围：-128 ~ 127
//2.一般情况下，我们习惯将整型定义为int
//3.定义long型变量时，需要以l或L结尾
//4.整型常量，默认为int型。
byte b1 = 12;
byte b2 = 127;

//b2 = 128;//编译不通过
```

```
//浮点型：float(4个字节) \ double(8个字节)
//1. float的表数范围比long还大。
//2. float:单精度  double:双精度
//3. 通常声明浮点型变量时，习惯使用double
//4. 定义float型变量时，需要以f或F结尾
//5. 浮点型常量，默认为double型。
```