Java中子类和父类相关方法的执行顺序

无意中看到下面一个题目，大家一起来看看最后的输出结果是什么。反正我看完之后，用IDE测试后感觉知识点得到巩固了。

```
1 /**
2  * 函数执行顺序测试
3  * Created by 萌小Q on 2017/5/17 0017.
4  */
5 public class ExeSeqTest {
6
7     public static void main(String [] args){
8         System.out.println(new B().getValue());
9     }
10    static class A{
11        protected int value;
12        public A(int v) {
13            setValue(v);
14        }
15        public void setValue(int value){
16            this.value = value;
17        }
18        public int getValue(){
19            try{
20                value++;
21                return value;
22            } catch(Exception e){
23                System.out.println(e.toString());
24            } finally {
25                this.setValue(value);
26                System.out.println(value);
27            }
28            return value;
29        }
30    }
31    static class B extends A{
32        public B() {
33            super(5);
34            setValue(getValue() - 3);
35        }
36        public void setValue(int value){
37            super.setValue(2 * value);
38        }
39    }
40 }
```

执行结果：

View Code

②JavaWeb学习笔记(9)
③JavaScript学习笔记(8)
④数据库学习(MySQL、Oracle)(7)
⑤Java框架与工具(3)
⑥JQuery学习笔记(2)
⑦Java开发知识查缺补漏(5)
Java面试宝典(1)
博客美化(1)
开发指南和问题整理(10)

文章分类(5)

Java查缺补漏(2)
Java面试(1)
Java序列化(2)

文章档案(5)

2016年7月 (5)

学习网站

Java面试题
Java入门基础教程
JQuery插件库
菜鸟要飞
码农网！！！！
慕课网
手册网
在线编辑、展示、分享、交流你的JavaScript 代码

在线手册大全

CSS3.0中文手册
JAVA学习笔记
jQuery 1.8 参考手册
SQL基础教程
在线手册大全汇总

积分与排名

积分 - 96658
排名 - 3227

最新评论

1. Re:Java提高篇——equals()与hashCode()方法详解
厉害呀妹子

--t飞

2. Re:框架基础——全面解析Java注解
真的很细，很厉害

你们答对了么？哈哈，现在来看一下代码具体执行情况：

1、首先是main方法，new了一个B对象，然后就是调用该对象的getValue()方法

```
public class ExeSeqTest {

    public static void main(String [] args){  args: {}
        System.out.println(new B().getValue());
    }
    static class A{
        protected int value;
        public A(int v) {
            setValue(v);
        }
```

2、执行B类的构造方法

```
    public static void main(String [] args){
        System.out.println(new B().getValue());
    }
    static class A{
        protected int value;  value: 0
        public A(int v) {
            setValue(v);
        }
        public void setValue(int value){
            this.value = value;
        }
        public int getValue(){
            try{
                value++;
                return value;
            } catch(Exception e){
                System.out.println(e.toString());
            } finally {
                this.setValue(value);
                System.out.println(value);
            }
            return value;
        }
    }
    static class B extends A{
        public B() {
            super( v: 5);
            setValue(getValue() - 3);
        }
        public void setValue(int value){
            super.setValue(2 * value);
        }
    }
```

3、执行B类构造方法里面的super方法，即执行B的父类A的构造方法。

--victor&selina

3. Re:Java提高篇——静态代码块、构造代码块、构造函数以及Java类初始化顺序

thanks，熬夜学习中ing

--Simplebirds

4. Re:Java提高篇——静态代码块、构造代码块、构造函数以及Java类初始化顺序

静态代码块：用staitc声明，jvm加载类时执行，仅执行一次

大妹纸，发现有错别字，static写成了staitc

--Alan_Lee

5. Re:Java文件操作①——XML文件的读取

@Testworm试试注释if判断，只打印节点名，你会发现共有9个节点...

--dioag

阅读排行榜

1. Java提高篇——对象克隆（复制）(55024)
2. Java文件操作①——XML文件的读取(36358)
3. Java数据库连接——JDBC基础知识（操作数据库：增删改查）(30229)
4. window.location.href和window.open的几种用法和区别(22818)
5. Java提高篇——equals()与hashCode()方法详解(17025)
6. Oracle数据库相关问题之ORA-12541:TNS:无监听程序(10093)
7. Java提高篇——Java 异常处理(9500)
8. 请求转发（Forward）和重定向（Redirect）的区别(7781)
9. Java提高篇——JVM加载class文件的原理机制(7602)
10. java 性能优化：35 个小细节，让你提升 java 代码的运行效率(7225)

评论排行榜

```
static class A{
    protected int value;  value: 0
    public A(int v) {  v: 5
        setValue(v);  v: 5
    }
    public void setValue(int value){
        this.value = value;
    }
    public int getValue(){
        try{
            value++;
            return value;
        } catch(Exception e){
            System.out.println(e.toString());
        } finally {
            this.setValue(value);
            System.out.println(value);
        }
        return value;
    }
}
static class B extends A{
    public B() {
        super( v: 5);
        setValue(getValue() - 3);
    }
    public void setValue(int value){
        super.setValue(2 * value);
    }
}
```

4、接下来就是执行setValue()方法了，但是此时A类和B类都有一个setValue()方法，到底执行哪一个呢，我一开始认为是A类的setValue()方法。

但在A类的构造方法中执行setValue()方法时，你是不是看到了，它执行的是子类B的setValue()方法。

显然这需要巩固一个知识点：**当子类重写了父类的函数，那么子类的对象如果调用该函数，一定调用的是重写过后的函数。可以通过super关键字进行父类的重写函数的调用。**

因为现在正在执行的是B类的构造方法，所以默认先会调用B类中的方法，如果B类中没有，才会调用其父类A中的方法。

1. Java提高篇——对象克隆（复制）(12)
2. 文件传输基础——Java IO流(9)
3. Java数据库连接——JDBC基础知识（操作数据库：增删改查）(9)
4. Oracle数据库之SQL基础（二）(6)
5. Java提高篇——equals()与hashCode()方法详解(5)

推荐排行榜

1. Java提高篇——对象克隆（复制）(24)
2. Java提高篇——静态代码块、构造代码块、构造函数以及Java类初始化顺序(16)
3. Java提高篇——equals()与hashCode()方法详解(15)
4. Java文件操作①——XML文件的读取(15)
5. Java数据库连接——JDBC基础知识（操作数据库：增删改查）(11)
6. Java提高篇——Java 异常处理(8)
7. Java中Native关键字的作用(7)
8. Java提高篇——JVM加载class文件的原理机制(6)
9. 请求转发（Forward）和重定向（Redirect）的区别(6)
10. window.location.href和window.open的几种用法和区别(6)

```
public static void main(String [] args){
    System.out.println(new B().getValue());
}
static class A{
    protected int value;  value: 0
    public A(int v) {
        setValue(v);
    }
    public void setValue(int value){
        this.value = value;
    }
    public int getValue(){
        try{
            value++;
            return value;
        } catch(Exception e){
            System.out.println(e.toString());
        } finally {
            this.setValue(value);
            System.out.println(value);
        }
        return value;
    }
}
static class B extends A{
    public B() {
        super( v: 5);
        setValue(getValue() - 3);
    }
    public void setValue(int value){  value: 5
        super.setValue(2 * value);  value: 5
    }
}
```

5、接下来到super.setValue(2 * value)，即执行A类的setValue()方法，这时，A类的成员变量value应该就变成了10

```
    public static void main(String [] args){
        System.out.println(new B().getValue());
    }
    static class A{
        protected int value;  value: 0
        public A(int v) {
            setValue(v);
        }
        public void setValue(int value){  value: 10
            this.value = value;  value: 0  value: 10
        }
        public int getValue(){
            try{
                value++;
                return value;
            } catch(Exception e){
                System.out.println(e.toString());
            } finally {
                this.setValue(value);
                System.out.println(value);
            }
            return value;
        }
    }
    static class B extends A{
        public B() {
            super( v: 5);
            setValue(getValue() - 3);
        }
        public void setValue(int value){
            super.setValue(2 * value);
        }
    }
```

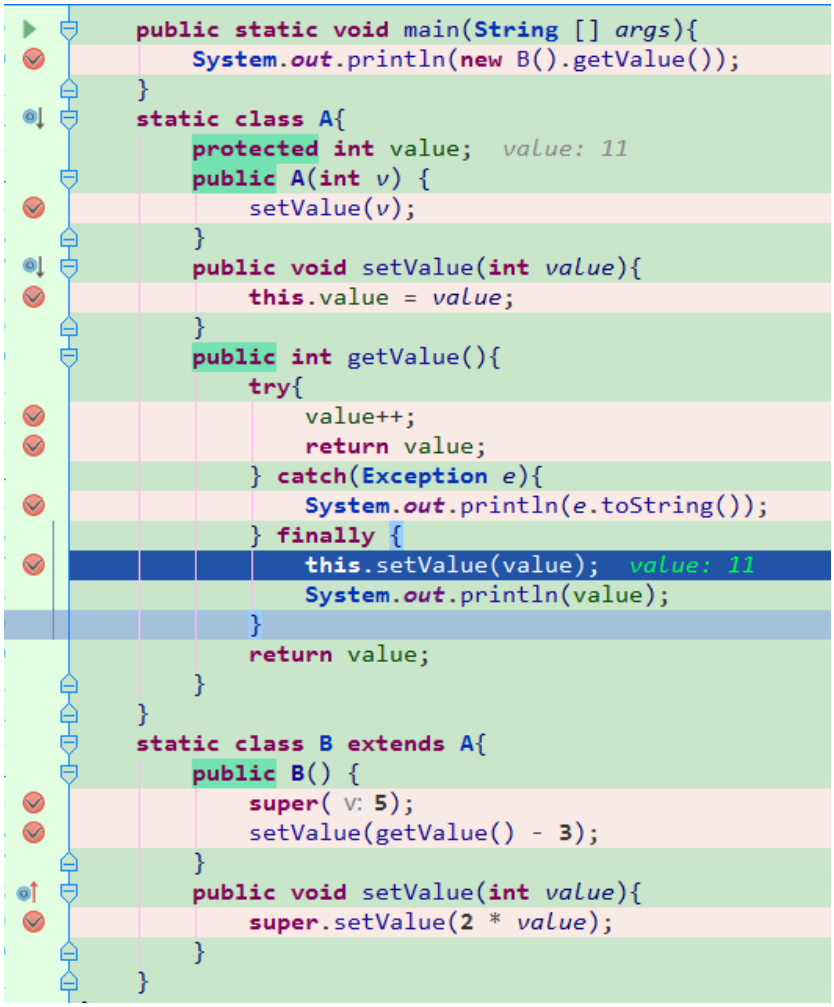
6、这时B类的构造方法中的super（5）就执行完了，然后就到了**setValue(getValue() - 3)**方法

```
        }
    }
    static class B extends A{
        public B() {
            super( v: 5);
            setValue(getValue() - 3);
        }
        public void setValue(int value){
            super.setValue(2 * value);
        }
    }
```

7、接着执行getValue()方法，首先在B类中找，但B类没有getValue()方法，所以就执行A类中的getValue()方法，A类中肯定是有的，要不然编译就不会通过

```
    public static void main(String [] args){
        System.out.println(new B().getValue());
    }
    static class A{
        protected int value;  value: 10
        public A(int v) {
            setValue(v);
        }
        public void setValue(int value){
            this.value = value;
        }
        public int getValue(){
            try{
                value++;  value: 10
                return value;
            } catch(Exception e){
                System.out.println(e.toString());
            } finally {
                this.setValue(value);
                System.out.println(value);
            }
            return value;
        }
    }
    static class B extends A{
        public B() {
            super( v: 5);
            setValue(getValue() - 3);
        }
        public void setValue(int value){
            super.setValue(2 * value);
        }
    }
```

8、然后就开始执行try、catch、finally这一块，给A的成员变量value自增，从之前的10变为11，然后直接返回value，没有捕获异常，继续到finally里面的this.setValue(value)



9、然后这个this指的到底是A类还是B类呢，答案是B类，因为现在是在执行B的构造方法，所以this指的应该是B类，即调用B类的setValue(int value)方法。

```
    public static void main(String [] args){
        System.out.println(new B().getValue());
    }
    static class A{
        protected int value;  value: 11
        public A(int v) {
            setValue(v);
        }
        public void setValue(int value){
            this.value = value;
        }
        public int getValue(){
            try{
                value++;
                return value;
            } catch(Exception e){
                System.out.println(e.toString());
            } finally {
                this.setValue(value);
                System.out.println(value);
            }
            return value;
        }
    }
    static class B extends A{
        public B() {
            super( v: 5);
            setValue(getValue() - 3);
        }
        public void setValue(int value){  value: 11
            super.setValue(2 * value);  value: 11
        }
    }
```

10、然后又super.setValue(2 * value)；执行父类A的setValue(int value)，把2 * 11作为参数传递，A类的setValue(int value)把传进来的value值赋给了A的成员变量value，变成了22。

```
    }
    static class A{
        protected int value;  value: 11
        public A(int v) {
            setValue(v);
        }
        public void setValue(int value){  value: 22
            this.value = value;  value: 11  value: 22
        }
        public int getValue(){
            try{
                value++;
                return value;
            } catch(Exception e){
                System.out.println(e.toString());
            } finally {
                this.setValue(value);
                System.out.println(value);
```

11、然后this.setValue(value)就执行完了，最后输出value，22

```
    public static void main(String [] args){
        System.out.println(new B().getValue());
    }
    static class A{
        protected int value;  value: 22
        public A(int v) {
            setValue(v);
        }
        public void setValue(int value){
            this.value = value;
        }
        public int getValue(){
            try{
                value++;
                return value;
            } catch(Exception e){
                System.out.println(e.toString());
            } finally {
                this.setValue(value);
                System.out.println(value);  value: 22
            }
            return value;
        }
    }
    static class B extends A{
        public B() {
            super( v: 5);
            setValue(getValue() - 3);
        }
        public void setValue(int value){
            super.setValue(2 * value);
        }
    }
```

12、到这儿getValue()方法就执行完了，但是有一点需要注意，此时的value为22，但是getValue()的返回值确是11，因为在try{ }中已经return了，所以这个方法的返回值已经保存下来了，是11，即使finally{ }里面又对value的值做出了改变，但是getValue()的返回值是不会变的(除了finally里面也有return返回值，它会覆盖前面try的返回值)。接着继续执行B类构造方法中的setValue(getValue() - 3)；getValue()是11，所以B的setValue(int value)方法的参数就为8，接着又到了super.setValue(2 * value)

```
    }
    static class B extends A{
        public B() {
            super( v: 5);
            setValue(getValue() - 3);
        }
        public void setValue(int value){  value: 8
            super.setValue(2 * value);  value: 8
        }
    }
```

13、调用A类的setValue(int value)方法，同时将参数赋值给A类的成员变量value，此时value变为16

```
    public static void main(String [] args){
        System.out.println(new B().getValue());
    }
    static class A{
        protected int value;  value: 22
        public A(int v) {
            setValue(v);
        }
        public void setValue(int value){  value: 16
            this.value = value;  value: 22  value: 16
        }
        public int getValue(){
            try{
                value++;
```

14、到这儿B类的构造方法就全部执行完了，也就是new B()，然后又调用了该对象 的getValue()方法，B类没有，但是父类A有，

```
public class ExeSeqTest {

    public static void main(String [] args){
        System.out.println(new B().getValue());
    }
    static class A{
        protected int value;  value: 16
        public A(int v) {
            setValue(v);
        }
        public void setValue(int value){
            this.value = value;
        }
        public int getValue(){
            try{
                value++;  value: 16
                return value;
            } catch(Exception e){
                System.out.println(e.toString());
            } finally {
                this.setValue(value);
                System.out.println(value);
            }
            return value;
        }
    }
```

继续try{ }、catch{ }、finally{ }，A类的成员变量value为16，然后value++，再返回，这时getValue()的返回值已经确定了，就是17，即使在finally中对value做出改变，其返回值不会变。然后到finally{ }，又是this.setValue(value)，前面已经说过了，这个this指的是B类的this，所以调用B类的setValue(int value)

```
    }
    static class B extends A{
        public B() {
            super( v: 5);
            setValue(getValue() - 3);
        }
        public void setValue(int value){  value: 17
            super.setValue(2 * value);  value: 17
        }
    }
```

接着又是super.setValue(2 * value)，调用A类的setValue()，并把2 * 17作为参数传递过去

```
    }
    static class A{
        protected int value;  value: 17
        public A(int v) {
            setValue(v);
        }
        public void setValue(int value){  value: 34
            this.value = value;  value: 17  value: 34
        }
        public int getValue(){
            try{
```

把参数赋给A的成员变量value，这时this.setValue(value)就执行完了，此时的value为34。最后输出value。

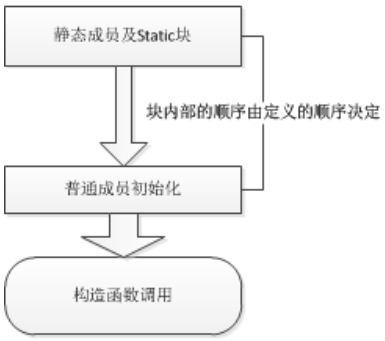
```
    static class A{
        protected int value;  value: 34
        public A(int v) {
            setValue(v);
        }
        public void setValue(int value){
            this.value = value;
        }
        public int getValue(){
            try{
                value++;
                return value;
            } catch(Exception e){
                System.out.println(e.toString());
            } finally {
                this.setValue(value);
                System.out.println(value);  value: 34
            }
            return value;
        }
    }
```

需要注意的是，此时的getValue()方法的返回值是17，这个前面已经提到了，到这儿，整个new B().getValue()就执行完了，最后又输出了getValue的返回值，也就是17。**所以整个过程执行完后的输出结果是22、34、17。。。。。。**

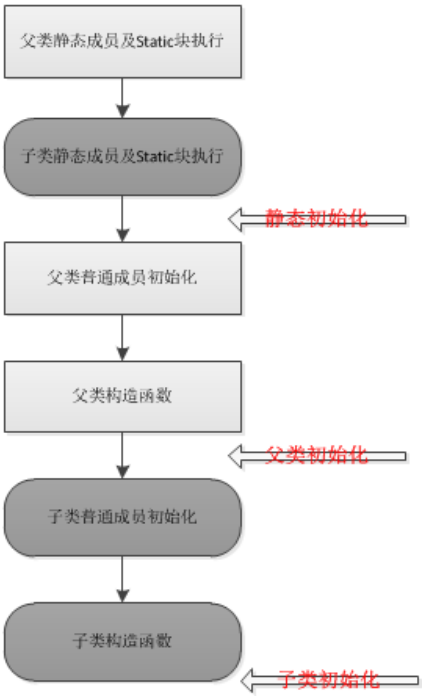
这道题虽然饶了很多弯，但是我们做完后发现整体过程其实并不是很复杂，就是子类继承父类，调用方法时先是调用子类中的方法，如果没有就调用父类中的方法，还有一点就是try{ }、catch{ }、finally{ }返回值的问题，一旦try{ }中返回了某一个值，如果finally有返回值，finally中的返回值会覆盖try的返回值，如果finally没有返回值，就是try中的返回值。掌握了这些，这道题就显得很简单了。

Java初始化顺序如图：

1 普通类初始化

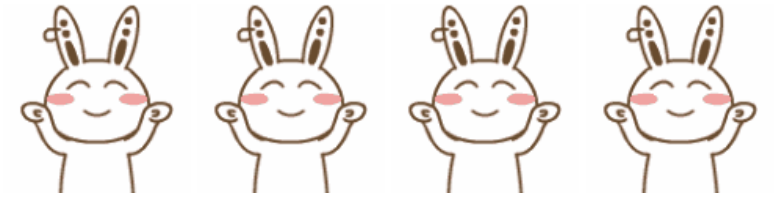


2 继承体系初始化



----------我是低调的分割线----------

如果对你有帮助，可以点击"推荐"哦~ (*∩_∩*)'



分类: ①Java基础学习笔记,⑦Java开发知识查缺补漏

标签: java基础

好文要顶 关注我 收藏该文

萌小Q
关注 - 6
粉丝 - 346
+加关注

2 0

« 上一篇：java 性能优化：35 个小细节，让你提升 java 代码的运行效率

posted @ 2017-05-17 17:33 萌小Q 阅读(973) 评论(2) 编辑 收藏

评论列表

#1楼 2017-09-08 10:33 一道桥

咦..笨笨 发现你写的很好啊

支持(0) 反对(0)

#2楼 2017-12-05 19:43 等待九月

往上延伸就是类加载机制（双亲委托）

支持(0) 反对(0)

注册用户登录后才能发表评论，请 登录 或 注册，访问网站首页。

【推荐】超50万VC++源码: 大型工控、组态\仿真、建模CAD源码2018！

【活动】杭州云栖·2050大会-全世界年青人因科技而团聚-源点

【抢购】新注册用户域名抢购1元起

最新IT新闻:

· 雷军最新单曲上线：如此鬼畜 境界无敌
· 乐视网：贾跃亭所有股票质押式回购交易均已违约
· Spotify未来靠何生存？ 或需涉足硬件业务推智能音箱
· 股东大会下周二召开 高通和博通发起股东争夺战
· 区块链公司密集注册 2月份共55家公司名称获预先核准

» 更多新闻...

最新知识库文章:

· 写给自学者的入门指南
· 和程序员谈恋爱
· 学会学习
· 优秀技术人的管理陷阱
· 作为一个程序员，数学对你到底有多重要

» 更多知识库文章...