[**Bro\_\_超**](https://www.cnblogs.com/heshuchao/)

[JAVA回调机制(CallBack)详解](https://www.cnblogs.com/heshuchao/p/5376298.html)

**序言**

最近学习java，接触到了回调机制(CallBack)。初识时感觉比较混乱，而且在网上搜索到的相关的讲解，要么一言带过，要么说的比较单纯的像是给CallBack做了一个定义。当然了，我在理解了回调之后，再去看网上的各种讲解，确实没什么问题。但是，对于初学的我来说，缺了一个循序渐进的过程。此处，将我对回调机制的个人理解，按照由浅到深的顺序描述一下，如有不妥之处，望不吝赐教！

开始之前，先想象一个场景：幼稚园的小朋友刚刚学习了10以内的加法。

**第1章. 故事的缘起**

幼师在黑板上写一个式子 “1 + 1 = ”，由小明同学来填空。

由于已经学习了10以内的加法，小明同学可以完全靠自己来计算这个题目，模拟该过程的代码如下：

[复制代码](javascript:void(0);)

1 public class Student

2 {

3 private String name = null;

4

5 public Student(String name)

6 {

7 // TODO Auto-generated constructor stub

8 this.name = name;

9 }

10

11 public void setName(String name)

12 {

13 this.name = name;

14 }

15

16 private int calcADD(int a, int b)

17 {

18 return a + b;

19 }

20

21 public void fillBlank(int a, int b)

22 {

23 int result = calcADD(a, b);

24 System.out.println(name + "心算:" + a + " + " + b + " = " + result);

25 }

26 }

[复制代码](javascript:void(0);)

小明同学在填空(fillBalnk)的时候，直接心算(clacADD)了一下，得出结果是2，并将结果写在空格里。测试代码如下：

[复制代码](javascript:void(0);)

1 public class Test

2 {

3 public static void main(String[] args)

4 {

5 int a = 1;

6 int b = 1;

7 Student s = new Student("小明");

8 s.fillBlank(a, b);

9 }

10 }

[复制代码](javascript:void(0);)

运行结果如下：

小明心算:1 + 1 = 2

该过程完全由Student类的实例对象单独完成，并未涉及回调机制。

**第2章. 幼师的找茬**

课间，幼师突发奇想在黑板上写了“168 + 291 = ”让小明完成，然后回办公室了。

花擦！为什么所有老师都跟小明过不去啊？明明超纲了好不好！这时候小明同学明显不能再像上面那样靠心算来完成了，正在懵逼的时候，班上的小红同学递过来一个只能计算加法的计算器（奸商啊）！！！！而小明同学恰好知道怎么用计算器，于是通过计算器计算得到结果并完成了填空。

计算器的代码为：

[复制代码](javascript:void(0);)

1 public class Calculator

2 {

3 public int add(int a, int b)

4 {

5 return a + b;

6 }

7 }

[复制代码](javascript:void(0);)

修改Student类，添加使用计算器的方法：

[复制代码](javascript:void(0);)

1 public class Student

2 {

3 private String name = null;

4

5 public Student(String name)

6 {

7 // TODO Auto-generated constructor stub

8 this.name = name;

9 }

10

11 public void setName(String name)

12 {

13 this.name = name;

14 }

15

16 @SuppressWarnings("unused")

17 private int calcADD(int a, int b)

18 {

19 return a + b;

20 }

21

22 private int useCalculator(int a, int b)

23 {

24 return new Calculator().add(a, b);

25 }

26

27 public void fillBlank(int a, int b)

28 {

29 int result = useCalculator(a, b);

30 System.out.println(name + "使用计算器:" + a + " + " + b + " = " + result);

31 }

32 }

[复制代码](javascript:void(0);)

测试代码如下：

[复制代码](javascript:void(0);)

1 public class Test

2 {

3 public static void main(String[] args)

4 {

5 int a = 168;

6 int b = 291;

7 Student s = new Student("小明");

8 s.fillBlank(a, b);

9 }

10 }

[复制代码](javascript:void(0);)

运行结果如下：

小明使用计算器:168 + 291 = 459

该过程中仍未涉及到回调机制，但是部分小明的部分工作已经实现了转移，由计算器来协助实现。

**3. 幼师回来了**

发现小明完成了3位数的加法，老师觉得小明很聪明，是个可塑之才。于是又在黑板上写下了“26549 + 16487 = ”，让小明上课之前完成填空，然后又回办公室了。

小明看着教室外面撒欢儿的小伙伴，不禁悲从中来。再不出去玩，这个课间就要废了啊！！！！ 看着小红再一次递上来的计算器，小明心生一计：让小红代劳。

小明告诉小红题目是“26549 + 16487 = ”，然后指出填写结果的具体位置，然后就出去快乐的玩耍了。

这里，不把小红单独实现出来，而是把这个只能算加法的计算器和小红看成一个整体，一个会算结果还会填空的超级计算器。这个超级计算器需要传的参数是两个加数和要填空的位置，而这些内容需要小明提前告知，也就是小明要把自己的一部分方法暴漏给小红，最简单的方法就是把自己的引用和两个加数一块告诉小红。

因此，超级计算器的add方法应该包含两个操作数和小明自身的引用，代码如下：

[复制代码](javascript:void(0);)

1 public class SuperCalculator

2 {

3 public void add(int a, int b, Student xiaoming)

4 {

5 int result = a + b;

6 xiaoming.fillBlank(a, b, result);

7 }

8 }

[复制代码](javascript:void(0);)

小明这边现在已经不需要心算，也不需要使用计算器了，因此只需要有一个方法可以向小红寻求帮助就行了，代码如下：

[复制代码](javascript:void(0);)

1 public class Student

2 {

3 private String name = null;

4

5 public Student(String name)

6 {

7 // TODO Auto-generated constructor stub

8 this.name = name;

9 }

10

11 public void setName(String name)

12 {

13 this.name = name;

14 }

15

16 public void callHelp (int a, int b)

17 {

18 new SuperCalculator().add(a, b, this);

19 }

20

21 public void fillBlank(int a, int b, int result)

22 {

23 System.out.println(name + "求助小红计算:" + a + " + " + b + " = " + result);

24 }

25 }

[复制代码](javascript:void(0);)

测试代码如下：

[复制代码](javascript:void(0);)

1 public class Test

2 {

3 public static void main(String[] args)

4 {

5 int a = 26549;

6 int b = 16487;

7 Student s = new Student("小明");

8 s.callHelp(a, b);

9 }

10 }

[复制代码](javascript:void(0);)

运行结果为：

小明求助小红计算:26549 + 16487 = 43036

执行流程为：小明通过自身的callHelp方法调用了小红（new SuperCalculator()）的add方法，在调用的时候将自身的引用（this）当做参数一并传入，小红在使用计算器得出结果之后，回调了小明的fillBlank方法，将结果填在了黑板上的空格里。

灯灯灯！到这里，回调功能就正式登场了，小明的fillBlank方法就是我们常说的回调函数。

通过这种方式，可以很明显的看出，对于完成老师的填空题这个任务上，小明已经不需要等待到加法做完且结果填写在黑板上才能去跟小伙伴们撒欢了，填空这个工作由超级计算器小红来做了。回调的优势已经开始体现了。

**第4章. 门口的婆婆**

幼稚园的门口有一个头发花白的老婆婆，每天风雨无阻在那里摆着地摊卖一些快过期的垃圾食品。由于年纪大了，脑子有些糊涂，经常算不清楚自己挣了多少钱。有一天，她无意间听到了小明跟小伙伴们吹嘘自己如何在小红的帮助下与幼师斗智斗勇。于是，婆婆决定找到小红牌超级计算器来做自己的小帮手，并提供一包卫龙辣条作为报酬。小红经不住诱惑，答应了。

回看一下上一章的代码，我们发现小红牌超级计算器的add方法需要的参数是两个整型变量和一个Student对象，但是老婆婆她不是学生，是个小商贩啊，这里肯定要做修改。这种情况下，我们很自然的会想到继承和多态。如果让小明这个学生和老婆婆这个小商贩从一个父类进行继承，那么我们只需要给小红牌超级计算器传入一个父类的引用就可以啦。

不过，实际使用中，考虑到java的单继承，以及不希望把自身太多东西暴漏给别人，这里使用从接口继承的方式配合内部类来做。

换句话说，小红希望以后继续向班里的小朋友们提供计算服务，同时还能向老婆婆提供算账服务，甚至以后能够拓展其他人的业务，于是她向所有的顾客约定了一个办法，用于统一的处理，也就是自己需要的操作数和做完计算之后应该怎么做。这个统一的方法，小红做成了一个接口，提供给了大家，代码如下：

1 public interface doJob

2 {

3 public void fillBlank(int a, int b, int result);

4 }

因为灵感来自帮小明填空，因此小红保留了初心，把所有业务都当做填空（fillBlank）来做。

同时，小红修改了自己的计算器，使其可以同时处理不同的实现了doJob接口的人，代码如下：

[复制代码](javascript:void(0);)

1 public class SuperCalculator

2 {

3 public void add(int a, int b, doJob customer)

4 {

5 int result = a + b;

6 customer.fillBlank(a, b, result);

7 }

8 }

[复制代码](javascript:void(0);)

小明和老婆婆拿到这个接口之后，只要实现了这个接口，就相当于按照统一的模式告诉小红得到结果之后的处理办法，按照之前说的使用内部类来做，代码如下：

小明的：

[复制代码](javascript:void(0);)

1 public class Student

2 {

3 private String name = null;

4

5 public Student(String name)

6 {

7 // TODO Auto-generated constructor stub

8 this.name = name;

9 }

10

11 public void setName(String name)

12 {

13 this.name = name;

14 }

15

16 public class doHomeWork implements doJob

17 {

18

19 @Override

20 public void fillBlank(int a, int b, int result)

21 {

22 // TODO Auto-generated method stub

23 System.out.println(name + "求助小红计算:" + a + " + " + b + " = " + result);

24 }

25

26 }

27

28 public void callHelp (int a, int b)

29 {

30 new SuperCalculator().add(a, b, new doHomeWork());

31 }

32 }

[复制代码](javascript:void(0);)

老婆婆的：

[复制代码](javascript:void(0);)

1 public class Seller

2 {

3 private String name = null;

4

5 public Seller(String name)

6 {

7 // TODO Auto-generated constructor stub

8 this.name = name;

9 }

10

11 public void setName(String name)

12 {

13 this.name = name;

14 }

15

16 public class doHomeWork implements doJob

17 {

18

19 @Override

20 public void fillBlank(int a, int b, int result)

21 {

22 // TODO Auto-generated method stub

23 System.out.println(name + "求助小红算账:" + a + " + " + b + " = " + result + "元");

24 }

25

26 }

27

28 public void callHelp (int a, int b)

29 {

30 new SuperCalculator().add(a, b, new doHomeWork());

31 }

32 }

[复制代码](javascript:void(0);)

测试程序如下：

[复制代码](javascript:void(0);)

1 public class Test

2 {

3 public static void main(String[] args)

4 {

5 int a = 56;

6 int b = 31;

7 int c = 26497;

8 int d = 11256;

9 Student s1 = new Student("小明");

10 Seller s2 = new Seller("老婆婆");

11

12 s1.callHelp(a, b);

13 s2.callHelp(c, d);

14 }

15 }

[复制代码](javascript:void(0);)

运行结果如下：

小明求助小红计算:56 + 31 = 87

老婆婆求助小红算账:26497 + 11256 = 37753元

**最后的话**

可以很明显的看到，小红已经把这件事情当做一个事业来做了，看她给接口命的名字doJob就知道了。

有人也许会问，为什么老婆婆摆摊能挣那么多钱？ 你的关注点有问题好吗！！这里聊的是回调机制啊！！

我只知道，后来小红的业务不断扩大，终于在幼稚园毕业之前，用挣到的钱买了人生的第一套房子。

完！！！