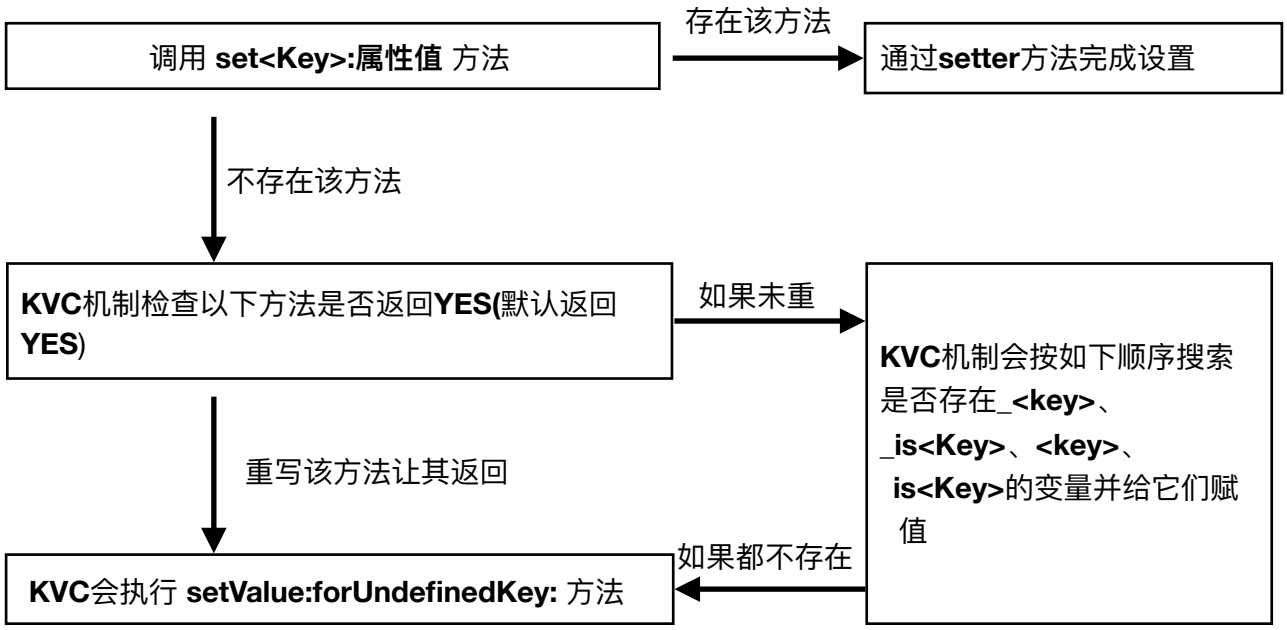
KVC

KVC（Key-Value Coding）键值编码。就是指iOS的开发中，可以允许开发者通过Key名直接访问对象的属性，或者给对象的属性赋值，而不需要调用明确的存取方法。这样就可以在运行时动态地访问和修改对象的属性，而不是在编译时确定。这也是iOS开发中的黑魔法之一，很多高级的iOS开发技巧都是基于KVC实现的。

一、KVC是怎么寻找key的?

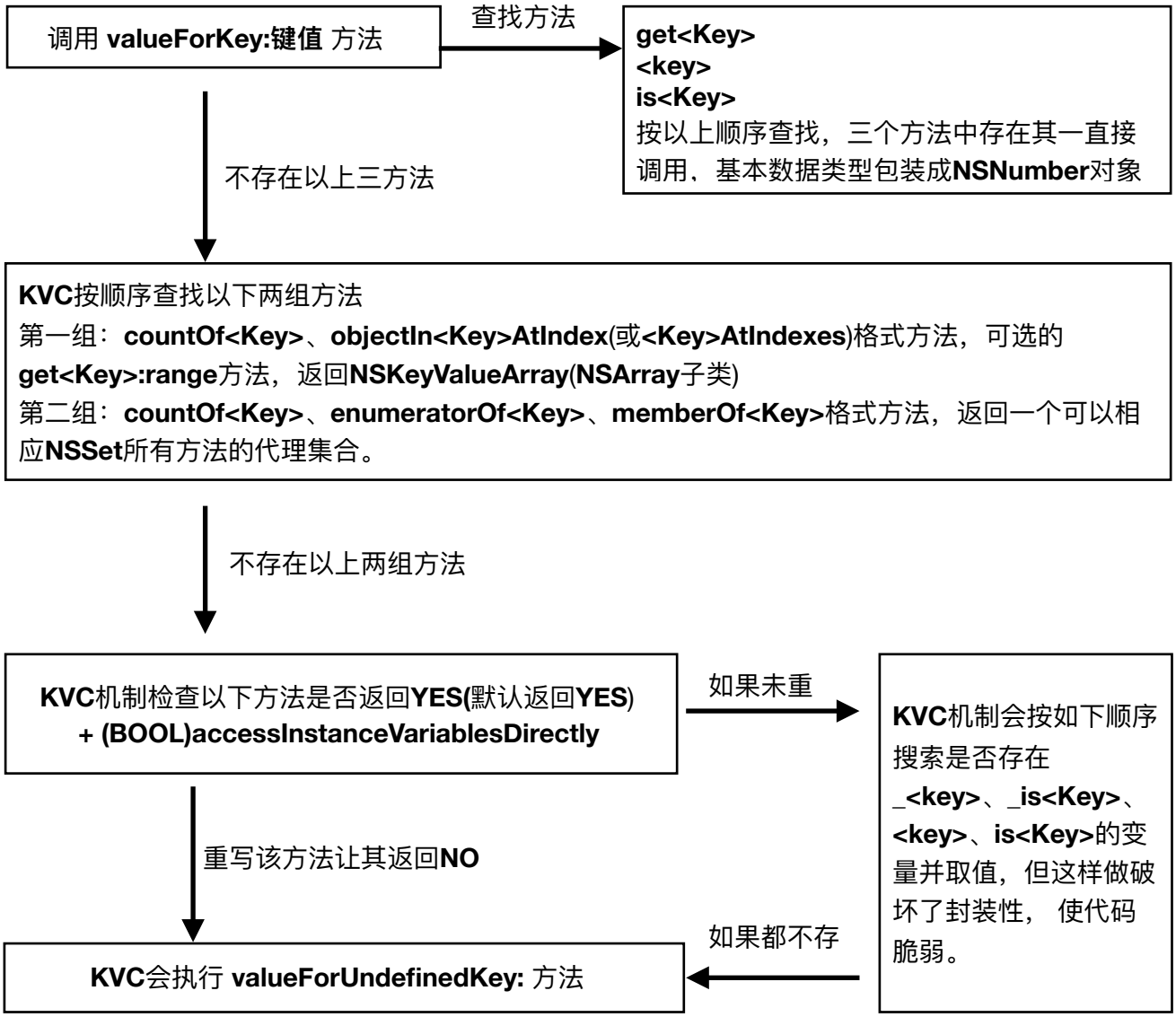
1、设置值

方法名：**- (void)setValue:(nullable id)value forKey:(NSString *)key;**
当调用 **setValue:属性值 forKey:键值** 的代码时，底层的执行机制如下



2、获取值

方法名：**- (nullable id)valueForKey:(NSString *)key;**
当调用 **valueForKey:键值** 的代码时，底层的执行机制如下



3、在KVC中使用keyPath

类的成员变量是自定义类或其他的复杂数据类型，获取或设置该属性可使用**KVC**的**keyPath**方法：
通过KeyPath来取值：**- (nullable id)valueForKeyPath:(NSString *)keyPath;**
通过KeyPath来设值：**- (void)setValue:(nullable id)value forKeyPath:(NSString *)keyPath;**

4、KVC如何处理异常

KVC中最常见的异常就是不小心使用了错误的**key**，或者在设值中不小心传递了**nil**的值，**KVC**中有专门的方法来处理这些异常，所以一般而言最好还是重写以下方法。

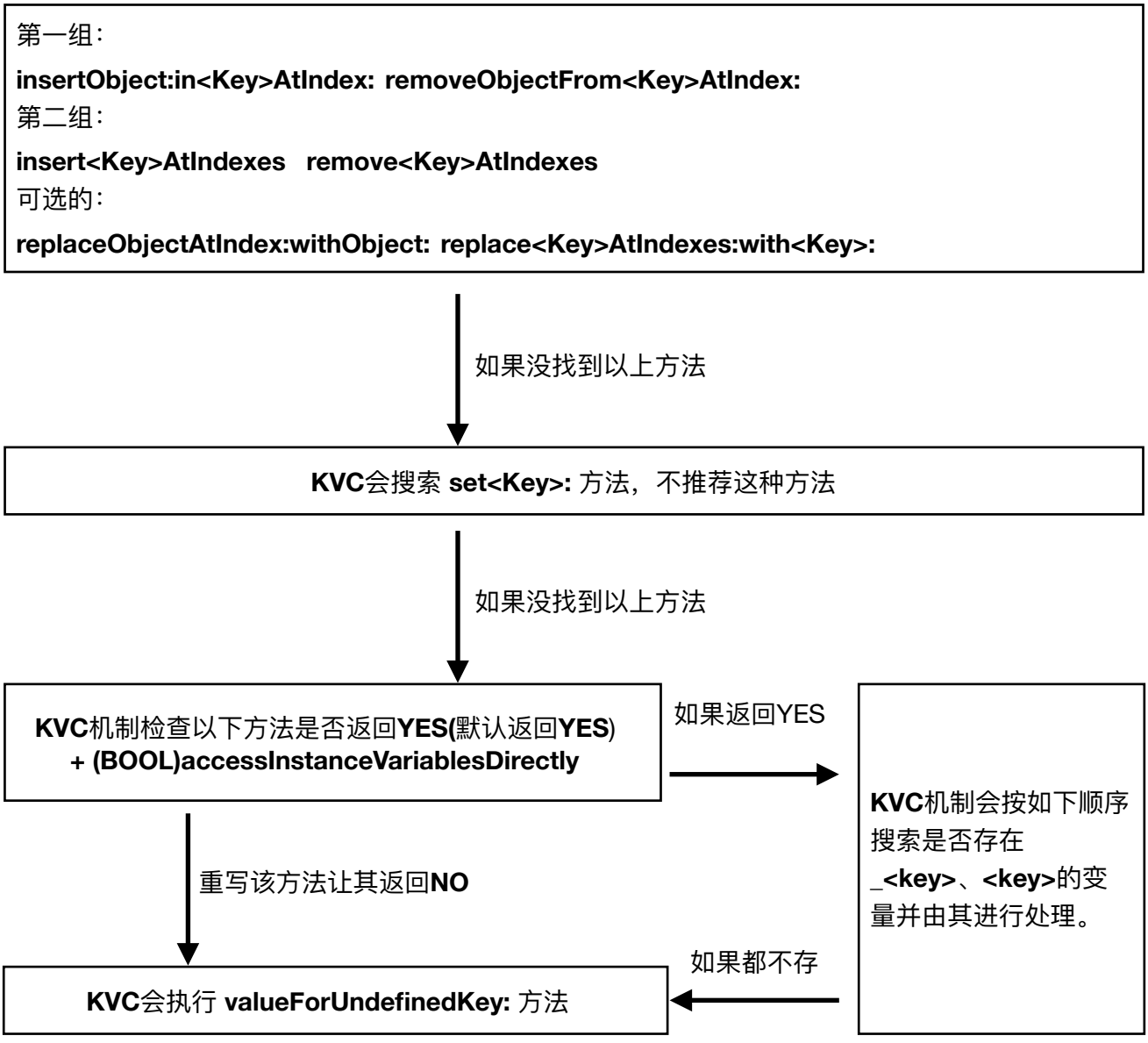
① **- (void)setValue:(id)value forUndefinedKey:(NSString *)key;**
② **- (void)setNilValueForKey:(NSString *)key;**
③ **- (id)valueForUndefinedKey:(NSString *)key**

5、KVC处理非对象和自定义对象

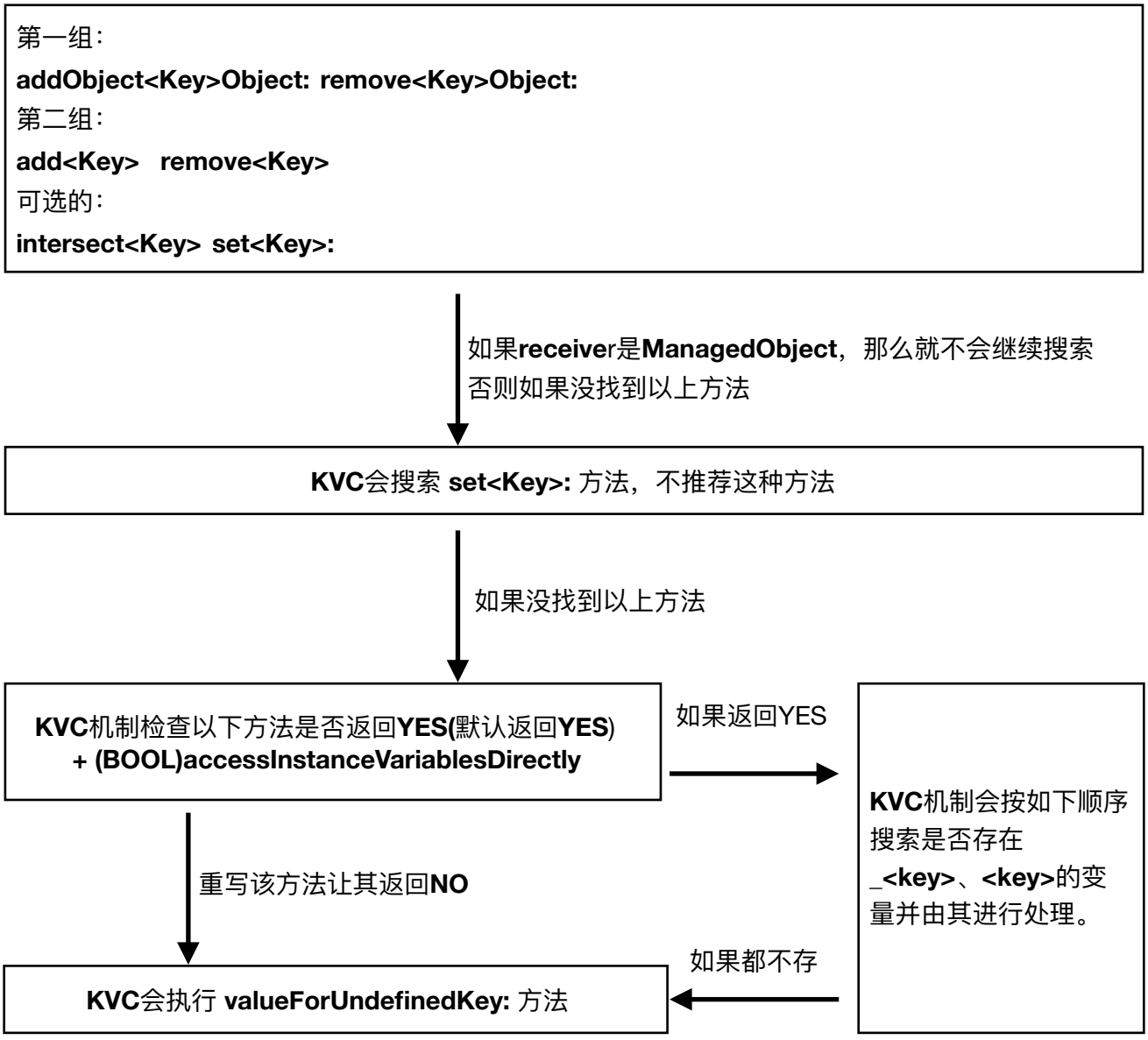
①**valueForKey:** 总是返回一个**id**对象，如果原本的变量类型是值类型或者结构体，返回值会封装成 **NSNumber** 或者 **NSValue** 对象。
②**setValue:forKey:** 必须手动将值类型转换成 **NSNumber** 或者 **NSValue** 类型，才能传递过去。
③自定义对象，**KVC**也会正确地设值和取值。因为传递进去和取出来的都是**id**类型，所以需要开发者自己担保类型的正确性，运行时**Objective-C**在发送消息的会检查类型，如果错误会直接抛出异常。

6、KVC与容器类

①不可变的有序容器属性(**NSArray**)和无序容器属性(**NSSet**)一般可以使用**valueForKey:**来获取。
②当对象的属性是可变的有序容器时用如下方法并返回一个可变有序数组：
- (NSMutableArray *)mutableArrayValueForKey:(NSString *)key;
调用以上方式时，KVC的搜索顺序如下：



③当对象的属性是可变的无序容器时用如下方法并返回一个可变无序数组：
- (NSMutableSet *)mutableSetValueForKey:(NSString *)key;
调用以上方式时，KVC的搜索顺序如下：



二、KVC的正确性验证

KVC提供了用来验证key对应的Value是否可用的如下方法：
- (BOOL)validateValue:(inout id __nullable * __nonnull)ioValue forKey:(NSString *)inKey error:(out NSError **)outError;
上面方法的默认实现是去探索类里面是否有一个如下的方法：
- (BOOL)validate<Key>:error:
如果有上述方法，就调用这个方法来返回，没有的话就直接返回YES
注意：KVC在设值时不会主动去做验证，需要开发者手动去验证。所以即使你在类里面写了验证方法，但是KVC因为不会去主动验证，所以还是能够设值成功。

三、KVC的使用

①动态地取值和设值
②用KVC来访问和修改私有变量
③Model和字典转换，充分地运用了KVC和Objctive-C的Runtime组合的技巧
④修改一些控件的内部属性，如UITextField中的placeHolderText
⑤操作集合
⑥用KVC实现高阶消息传递：valueForKey:将会被传递给容器中的每一个对象，而不是容器本身进行操作。结果会被添加进返回的容器中，这样，开发者可以很方便的操作集合来返回另一个集合。
⑦用KVC中的函数操作集合
1）简单集合运算符**@avg， @count ， @max ， @min ，@sum5**
2）对象运算符**@distinctUnionOfObjects，@unionOfObjects** 它们的返回值都是NSArray，区别是前者返回的元素都是唯一的，是去重以后的结果；后者返回的元素是全集。
3）Array和Set操作符
@distinctUnionOfArrays：该操作会返回一个数组，这个数组包含不同的对象，不同的对象是在从关键路径到操作器右边的被指定的属性里。
@unionOfArrays：该操作会返回一个数组，这个数组包含的对象是在从关键路径到操作器右边的被指定的属性里和**@distinctUnionOfArrays**不一样，重复的对象不会被移除。
@distinctUnionOfSets：和@distinctUnionOfArrays类似。
⑧KVO，KVO是基于KVC实现的。