**JAVA泛型**

**出现的原因：**

为了编写通用性代码，当要使用的数据类型不确定时，在没有泛型机制的时，一般把数据类型声明为Object类型，因为Object类型是所有类型的父类。但是这样会出现类型转换的问题，比如声明一个Object类型的数组，那么它就既可以放String类型，double,自己定义的类，当提取出来使用的时候，就需要强制类型转换，很可能就会发生错误，发生错误是运行时错误，不方面调试。

在使用泛型后，编译时期就可以保证类型安全，消除强制类型转换

**使用场景：**

1. 编写适用于多种类型的代码
2. 让编译器检查容器的类型是否正确

**定义：**

泛型的实质就是类型参数化，即把类型当成参数传递。

泛型通配符：指定类型的参数范围

**分类：**

泛型类

使用场景：容器类

泛型接口

使用场景：策略模式的公共策略（Comparator）

泛型方法

核心概念：

泛型擦除：

例子：

我们有一间教室是为运动员培训理论知识，我告诉门口的保安只能让运动员进入（编译器检查），但运动员进入后，我就开始上课，这个时候我把他们当成学生（**泛型擦除**）而不是运动员，他们只能听课、做笔记获者提问，不能再教室里跑步，跳高等（**失去类的具体信息**）。教室的门口就是**边界**，我们保证就来的是运动员，当他们就来后就知识学生，出去后他们是运动员，可以去跳高等。

边界：

T extends A

例子：

某老师要给某领导的儿子或孙子开小灶，他告诉门口的保安只有某领导的儿子或孙子才能进，并且只能进儿子或者孙子

? extentdsA

例子：

某老师要给某领导的儿子或孙子开小灶，他告诉门口的保安只有某领导的儿子或孙子才能进，并且只能进儿子或者孙子，保安无法保证那个是儿子或者是孙子，所以他都不放进来，但如果你出去的话，他会放你出去，它知道肯定是某领导的儿子或孙子。

·· ？ super A

例子：

某老师要给某领导的儿子或孙子开小灶，他告诉门口的保安只有某领导的儿子或孙子才能进，并且只能进儿子或者孙子，保安无法保证那个是儿子或者是孙子，但他把你放进来，但如果你出去的话，他不会放你出去，他不知道你到底是某领导的儿子还是孙子。