**一、代码 snippest**

**class** Foo {

**public** Foo(String name) {

**super**();

System.*out*.println("Foo for " + name);

}

}

**class** Parent {

**private** Foo foo = **new** Foo("Parent");

**static** {

System.*out*.println("in static of Parent");

}

**public** Parent() {

**super**();

System.*out*.println("in Parent ctr");

}

}

**class** Child **extends** Parent {

**public** **static** **final** **int** *CONSTANT\_VAL* = 10;

**private** Foo foo = **new** Foo("Child");

**static** {

System.*out*.println("in static of Child");

}

**public** Child() {

**super**();

System.*out*.println("in Child ctr");

}

**public** **static** **void** staticTest() { };

}

**二、直接实例化**

**public** **class** Test {

**public** **static** **void** main(String[] args) {

**new** Child();

System.*out*.println("------------");

**new** Child();

}

}

**输出为**

in static of Parent

in static of Child

Foo for Parent

in Parent ctr

Foo for Child

in Child ctr

------------

Foo for Parent

in Parent ctr

Foo for Child

in Child ctr

**结论，**

1. 静态变量/静态块先初始化，从父类到子类. 且一个类中的静态变量/静态块只初始化一次，这些静态变量存放在JVM的方法区中

2. 父类先初始化(先父类的非静态成员，然后是父类的构造函数)

3. 子类最后初始化(先子类的非静态成员，然后是子类的构造函数)

**三、直接调用类中的静态方法或静态变量**

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Child.*staticTest*();

Child.*staticTest*();

}

}

输出：

in static of Parent

in static of Child

结论， 如果直接调用一个类中的静态方法/或静态变量，**并不会引起类的实例化，但是会对静态方法/静态块进行初始化(注意，只会一次)**

**三、直接引用static final变量**

直接引用类中static final的变量，并不会引起类中静态方法/静态变量的初始化