```
package com.sxt.test;
/**
 * JVM内存分析
 * @author 江
 *
 */

public class Demo01 {
        static {
                System.err.println("静态初始化Demo01");
        }


   public static void main(String[] args) throws ClassNotFoundException {
            System.err.println("Demo01的main方法!");
            A a=new A();
            System.err.println(A.width);
            A a2=new A();

            //类的主动引用,会初始化
            /**
             * 1.new一个类的对象
             * 2.调用类的静态成员（除了final常量）和静态方法
             * 3.使用反射
             * 4.当虚拟机启动时，则一定会初始化Hello类，说白了就是先启动main方法所在的类
             * 5.当初始化一个类，如果其父类没有没有被初始化，则会先初始化它的父类
             */
            new A();
            System.err.println(A.width);
            Class.forName("com.sxt.test.A");

            //类的被动引用,不会初始化
            /**
             *1.当访问一个静态域时，只有真正声明这个域的类才会被初始化
```

```
        *2.通过数组定义类引用，不会触发此类的初始化
        *3.引用常量不会触发此类的初始化(常量在编译阶段就存入调用类的常量池中了)
         */
        System.err.println(B.width);
        A[] as=new A[10];
        System.err.println(A.MAX);
}

}

class A_Father extends Object{
        static {
                System.err.println("静态初始化A_Father");
        }
}

class A extends A_Father{
        public static int width=100;    //静态变量，静态域，field
        public static final int MAX=100;

        static {
                System.err.println("静态初始化类A");
                width=300;
        }

        public A() {
                System.err.println("创建A类的对象");
        }
}

class B extends A{
        static {
                System.err.println("静态初始化B");
        }
}
```