```
package main.java.Test1;

import com.sun.org.apache.xpath.internal.objects.XObject;
import com.sun.scenario.effect.impl.sw.sse.SSEBlend_SRC_OUTPeer;

import java.io.*;

public class ClassloaderDemo extends ClassLoader {
    private String classLoaderName;

    private final String fileExtension=".class";

    private String path;

    public ClassloaderDemo(ClassLoader parent, String classLoaderName) {
        super(parent);  //将系统类加载器当做该类加载器的父类加载器
        this.classLoaderName = classLoaderName;
    }

    public ClassloaderDemo(String classLoaderName) {
        super();   //显示的指定该类加载器的父类加载器
        this.classLoaderName = classLoaderName;
    }

    public void setPath(String path){
        this.path=path;
    }

    @Override
    public String toString() {
        return "["+this.classLoaderName+"]";
    }

    private byte[] loadClassData(String name){
        InputStream is=null;
        byte[] data=null;
        ByteArrayOutputStream baos=null;

        name=name.replace(".","\\");

        try{
            baos=new ByteArrayOutputStream();
            is=new FileInputStream(new File(this.path+name+this.fileExtension));
            int ch=0;
            while((ch=is.read())!=-1){
                baos.write(ch);
```

```
            }
            data=baos.toByteArray();
            return data;
        }catch(Exception e){
            e.printStackTrace();
            return null;
        }finally {
            if(null!=baos){
                try {
                    baos.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }

            if(null!=is){
                try {
                    is.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    @Override
    protected Class<?> findClass(String name) throws ClassNotFoundException {
        byte[] data=loadClassData(name);
        System.out.println(this.classLoaderName);
        return this.defineClass(name,data,0,data.length);
    }

    public static void main(String[] args) throws IllegalAccessException,
InstantiationException, ClassNotFoundException {
        //每一个加载器都有一个自己的命名空间
        //一个加载器只能加载一个类一次
        //不同类可以加载
        ClassloaderDemo demo=new ClassloaderDemo("loader1");
        demo.setPath("D:\\二级C语言VIP题库\\");
        Class<?> clazz=demo.loadClass("main.java.Test1.SuperClass");
        System.out.println("class:"+clazz.hashCode());
        Object obj=clazz.newInstance();
        System.out.println(obj);

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

        //不同加载器
```

```
        ClassloaderDemo demo2=new ClassloaderDemo("loader2");
        demo2.setPath("D:\\二级C语言VIP题库\\");
        Class<?> clazz2=demo2.loadClass("main.java.Test1.SuperClass");
        System.out.println("class:"+clazz2.hashCode());
        Object obj2=clazz2.newInstance();
        System.out.println(obj2);

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

        //双亲委托
        ClassloaderDemo demo3=new ClassloaderDemo(demo2,"loader3");
        demo3.setPath("D:\\二级C语言VIP题库\\");
        Class<?> clazz3=demo3.loadClass("main.java.Test1.SuperClass");
        System.out.println("class:"+clazz3.hashCode());
        Object obj3=clazz3.newInstance();
        System.out.println(obj3);

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

        //-XX:+loadClassUnloading  测试卸载
        //当引用对象为0时,该加载器中的类对象将会被卸载
        //demo的三个参数直接设置为null时,可以直接卸载回收
        //demo2和demo3因为是父子关系,所以有两套引用
        demo2=null;
        obj2=null;
        clazz2=null;
        System.gc();

        demo3=new ClassloaderDemo("loader3");

        demo3.setPath("D:\\二级C语言VIP题库\\");
        clazz3=demo3.loadClass("main.java.Test1.SuperClass");
        System.out.println("class:"+clazz3.hashCode());
        obj3=clazz3.newInstance();
        System.gc();
        System.out.println(obj3);
    }
}
```