```
package com.sxt.reflection;

import java.lang.reflect.Constructor;
import java.lang.reflect.Field;
import java.lang.reflect.Method;

import com.sxt.bean.User;

/**
 * 通过反射API动态的操作：构造器，方法，属性
 * @author 江
 *
 */

public class Demo03 {
  public static void main(String[] args) {
         String path="com.sxt.bean.User";
         try {
                Class<User> clazz=(Class<User>)Class.forName(path);

                //通过反射API动态调用构造方法，构造对象
                User u1=clazz.newInstance(); //获得无参实例对象

        Constructor<User>
constructor=clazz.getConstructor(int.class,int.class,String.class);
        User u2=constructor.newInstance(1001,18,"莫崽");
                System.err.println(u2.getUname());

                //通过反射API调用普通方法
                User u3=clazz.getConstructor().newInstance();
                Method method=clazz.getDeclaredMethod("setUname", String.class);
                method.invoke(u3, "三玖");    //u2.setUname("江");
                System.err.println(u3.getUname());

                //通过反射API操作属性
                User u4=clazz.getConstructor().newInstance();
                Field field=clazz.getDeclaredField("uname");
                field.setAccessible(true);    //禁止进行安全检查,可以对私有的属
性进行操作，并且能提高性能
                field.set(u4, "多罗罗");
                System.err.println(u4.getUname());
                System.err.println(field.get(u4));

        } catch (Exception e) {
                e.printStackTrace();
        }



}
}
```