java开发工具

2018年7月1日 14:58

Eclipse
免费
纯Java语言编写
免安装
扩展性强

MyEclipse
WEB开发

eclipse快捷键

2018年7月1日 19:26

Alt+/ 内容辅助键

main放法：main+alt+/ 回车即可
输出语句：syso+alt+/ 回车即可
提示作用，起名字

自动生成构造方法

2018年7月3日 13:53

无参构造方法：
在代码区域右键--sourse--Generate Constructors from Superclass

带参构造方法：
在代码区域右键--sourse--Generate Constructors from using field

可以使用快捷键

@override是注解，表示重写父类方法

jar包

2018年7月3日 15:23

jar是多个class文件的压缩包
别人写好的东西
打jar包-->复制到项目路径下并添加至构建路径
build path

删除和导入项目

2018年7月3日 16:16

删除：
从项目区域中删除
从硬盘上删除

导入
右键import-->general-->existing...

查看项目所在路径
右键-->properties-->Resourse-->Location

导入项目注意事项：
不能出现同名项目
自己随意建立的文件夹不能导入

修改项目问题：
不能随意修改项目名-->要同时修改配置文件

API

2018年7月4日 14:20

API-->Application Programming Interface
指JDK中提供的各种功能的java类

object类方法

2018年7月4日 15:05

```
/*
 * Object:类 Object 是类层次结构的根类。每个类都使用 Object 作为超类
 * 每个类都直接或间接的继承自Object类
 * Object只有无参构造方法
 *
 * Object类的方法:
 *		public int hashCode():返回该对象的哈希码值
 *			注意：哈希值是根据哈希算法计算出来的值，与地址值有关但不是地址值
 *
 *		public final Class getClass():返回此Object的运行类
 *			Class类的方法：
 *				public String getName():
 *				以 String 的形式返回此 Class 对象所表示的实体名称。
 */
public class StudentDemo {
	public static void main(String[] args) {
		Student s1 = new Student();
		System.out.println(s1.hashCode());// 2018699554
		Student s2 = new Student();
		System.out.println(s2.hashCode());// 1311053135
		Student s3 = s1;
		System.out.println(s3.hashCode());// 2018699554
		System.out.println("------------------");

		Student s = new Student();
		Class c = s.getClass();
		String str = c.getName();
		System.out.println(str);// cn.itcast_01.Student

		//链式编程
		String str2 = s.getClass().getName();
		System.out.println(str2);
	}
}
```

toString()

2018年7月4日 15:06

```
package cn.itcast_02;

/*
 * public String toString():返回该对象的字符串表示
 * Integer类下的一个静态方法:
 *              public static String toHexString(int i):把一个整数转换为16进制
 * 这个信息是没有意义的，要重写该方法
 *              把该类的所有成员变量值返回即可
 *              自动生成toString()方法
 *              直接输出一个对象的名称，其实就是调用该对象的toString()方法
 */
public class StudentDemo {
    public static void main(String[] args) {
        Student s = new Student();
        System.out.println(s.hashCode());
        System.out.println(s.getClass().getName());
        System.out.println("------------");
        System.out.println(s.toString());// cn.itcast_02.Student@7852e922

        // toString()方法的值等价于
        // getClass().getName() + '@' + Integer.toHexString(hashCode())

        System.out.println(s.getClass().getName() + "@" + Integer.toHexString(s.hashCode()));
        // cn.itcast_02.Student@7852e922

        //直接输出对象名称
        System.out.println(s);//Student [name=null, age=0]
    }
}
```

```
public String toString() {
        return "Student [name=" + name + ",
        age=" + age + "]";
    }
```

equals()

2018年7月4日 15:31

```
@Override
	public boolean equals(Object obj) {
		// 为了提高效率
		if (this == obj) {
			return true;
		}

		// 为了提高代码的健壮性，先判断obj是否为学生对象，不是直接返回false
		//对象名 instanceof 类名  判断对象名是否为类的对象
		if(!(obj instanceof Student)) {
			return false;
		}

		Student s = (Student) obj;
		/*
		 * if(this.name.equals(s.name) && this.age == s.age) { return true; }else {
		 * return false; }
		 */
		return this.name.equals(s.name) && this.age == s.age;
	}
	//可以自动生成
```

```
package cn.itcast_03;

/*
 * public boolean equals(Object obj):指式此对象与其它对象是否相等
 * 这个方法默认比较地址值,意义不大，因此要重写该方法
 * 一般比较变量是否相同
 * 看源码：
 * public boolean equals(Object obj) {
 *     return (this == obj);
 *   }
 */
public class StudentDemo {
	public static void main(String[] args) {
		Student s1 = new Student("xiao", 22);
		Student s2 = new Student("xiao", 22);
		System.out.println(s1 == s2);// false
		Student s3 = s1;
		System.out.println(s1 == s3);// true
		System.out.println("--------------");
		System.out.println(s1.equals(s2));// false
		System.out.println(s1.equals(s1));// true
		System.out.println(s1.equals(s3));// true
		Student s4 = new Student("feng",30);
		System.out.println(s1.equals(s4));//false

		Demo d = new Demo();
		System.out.println(s1.equals(d));//ClassCastException
	}
}

class Demo{}
```

Clone()

2018年7月4日 16:09

```
package cn.itcast_04;

/*
 * protected void finalize()
 *              当垃圾回收器确定不存在对该对象的更多引用时，由对象的垃圾回收器调用此方法
 * protected Object clone()
 *              创建并返回此对象的一个副本
 *              重写该方法
 *
 * Cloneable:此类实现了Cloneable接口，以指式Object.clone()可以合法的对该类实例进行字段复制
 *              这个接口是标记接口
 */
public class StudentDemo {
    public static void main(String[] args) throws CloneNotSupportedException {
        // 创建学生对象
        Student s = new Student();
        s.setName("xiao");
        s.setAge(22);
        // 克隆学生对象
        Object obj = s.clone();
        Student s2 = (Student) obj;

        System.out.println(s.getName() + "---" + s.getAge());
        System.out.println(s2.getName() + "---" + s2.getAge());

        Student s3 = s;
        s3.setName("feng");
        s3.setAge(30);
        System.out.println(s.getName() + "---" + s.getAge());// feng---30
        System.out.println(s2.getName() + "---" + s2.getAge());// xiao---22不变
        System.out.println(s3.getName() + "---" + s3.getAge());// feng---30
    }
}
```

```
package cn.itcast_04;

public class Student implements Cloneable{
    private String name;
    private int age;

    public Student() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Student(String name, int age) {
        super();
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    protected Object clone() throws
    CloneNotSupportedException {
        return super.clone();
    }
}
```