1. myeclipse 的安装和使用

Eclipse 是一个免费的开发工具

Myeclipse 是一个收费的插件

安装目录不能有中文和空格

安装完成后选择一个空间，这个空间不能有中文和空格

破解myeclipse

运行run.bat文件，但是运行之前必须要安装jdk，同时配置环境变量

Myeclipse的使用

1. 创建工程 javaProject 和webproject常用的两种

可以选择以来的jdk 可以使用自带的jdk，也可是以哦那个安装的jdk

1. 创建包 package

XX.XX.XX

1. 在包里面创建一个类

类的命名规范 :多单词时首字母大写

比如：TestDemol

1. 在类中创建方法：

Public void test1（参数列表）

{方法体和}

\*定义变量

-变量的命名规范

\*\*首字母小写，第二个单词的首字母要大写，比如userName

\*这些命名还有一种方式

\*\*使用汉语拼音命名 yonghuming mima

\*\*不能把汉语拼音和英文字母混合使用

userMing

\*命名的最基本的原则：看到名字知道是什么含义

\*代码需要有缩进

\*运行程序 run as java application

debug as java application】

1. debug调试模式（断点调试模式）

使用这种模式可以调试模式，调试程序（可以观察程序中数据的变化）

1. 设置一个断点（让程序停止到这一行）

右键左边的行 show linenumber 然后点击设置的行，点击debug

2、2、debug的调试模式（断点调试模式）

\*使用这种模式，调试程序（看到程序里面数据的变化）

\*使用debug第一步需要设置一个断点（让程序运行停止在这一行）

-显示出来行号

-双击左边，出现一个圆点，表示设置了一个断点

\*使用debug as方式，运行程序

-提示是否进入到调试界面，yes

-在断点那一个，有一个绿色条，表示程序停止在这一行，没有向下运行

\*可以让程序向下执行，

-使用step over快捷键是F6(单步执行）

-resume F8:表示调试结束，直接向下运行

\*\*比如当前的断点之后还有断点，跳到下一个断点，

\*\*如果当前断点后面没有断点，程序直接运行结束

\*debug另外一个用途

\*\*查看程序的源代码

\*\*F7step into:进入到方法

\*\*step return：返回T

3、myeclipse的快捷键的使用

\*代码提示alt/

\*快速导包 ctrl shifto

\*单行注释ctrl/

\*去掉单行注释ctrl/

\*多行注释ctrl shift/

\*去掉多行注释 ctrl shift\

\*删除行ctrld

4、　单元测试

　单元测试

　　测试对象是 是一个类中的方法

　　juint不是javase的一部分，想要使用导入jar包

　　　　但是，在myeclipse中自带了junit的jar包

　　首先junit版本 3.x 4.x

　　　　单元测试方法时候，方法命名规则 public void 方法名() {}

　　使用注解方式运行测试方法, 在方法的上面

　　　　@Test：表示方法进行单元测试

public class TestJunit {

　　　　public void testAdd(int a,int b) {

　　　　　　System.out.println(a+b);

　　　　}

}

　　　　---  @Test

　　　　public void testAdd1() {

　　　　　　TestJunit test01 = new TestJunit();

　　　　　　　　test01.testAdd(2, 3);

　　　　}

　　　　- 选中方法名称，右键运行 点击run as --- junit  test

　　　　- 当出现绿色条，表示方法测试通过

　　　　- 当出现了红棕色条，表示方法测试不通过

 　　　　--- 要运行类中的多个测试方法，点击类中的其他位置，run as --- junit  test

 　　　　@Ignore ：表示这个方法不进行单元测试

  　　　　@Before: 在每个方法执行运行

　　　　 @After：在每个方法之后运行

　　断言（了解）

　　- Assert.assertEquals("测试期望的值", "方法运行的实际的值")

@Test

public void test02() {

int a = 3;

int b = 5;

int sum = a+b;

//使用断言

//Assert.assertEquals("测试期望的值", "方法运行的实际的值")

Assert.assertEquals(80, sum);

}

5、泛型的简介

　为什么要使用泛型？

　　- 一般使用在集合上

　　【比如现在把一个字符串类型的值放入到集合里面，这个时候，这个值放入到集合之后，失去本身的类型，只能是object类型，

　　这个时候，比如想要对这个值进行类型转换，很容易出现类型转换错误，怎么解决这个问题，可以使用泛型来解决】

　　在集合上如何使用泛型

　　　　- 常用集合 list  set  map

　　　　- 泛型语法 集合<String>  比如 List<String>

　　在泛型里面写是一个对象，String 不能写基本的数据类型 比如int ()

　　写基本的数据类型对应的包装类

　　　　byte -- Byte

　　　　short -- Short

　　　　int -- Integer

 　　　　 long -- Long

　　　　float -- Float

　　　　double -- Double

　　　　char   -- Character

　　　　boolean -- Boolean

　　在list上使用泛型

　　list的三种实现 ArrayList  linkedList  Vector

　　代码：

//泛型在list上的使用

@Test

public void testList() {

List<String> list = new ArrayList<String>();

list.add("aaa");

list.add("bbb");

list.add("ccc");

//遍历list集合 有几种方式 三种

//普通for循环 迭代器 增强for

//普通for循环

for(int i=0;i<list.size();i++) {

String s = list.get(i);

System.out.println(s);

}

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

//使用增强for

for (String s1 : list) {

System.out.println(s1);

}

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

//使用迭代器遍历

Iterator<String> it = list.iterator();

while(it.hasNext()) {

System.out.println(it.next());

}

}

　　　　作业1： ArrayList  linkedList  Vector 这三个区别

　　在set上使用泛型

　　代码：

//泛型使用set集合上

@Test

public void testSet() {

Set<String> set = new HashSet<String>();

set.add("www");

set.add("qqq");

set.add("zzz");

//set.add("qqq"); //无序元素不重复

//遍历set 有几种方式 两种

//迭代器 增强for

//使用增强for遍历

for (String s2 : set) {

System.out.println(s2);

}

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

//使用迭代器遍历

Iterator<String> it1 = set.iterator();

while(it1.hasNext()) {

System.out.println(it1.next());

}

}

　　在map上面使用泛型

　　- map结构：key-valu形式

　　代码：

//在map上使用泛型

@Test

public void testMap() {

Map<String,String> map = new HashMap<String,String>();

map.put("aaa", "111");

map.put("bbb", "222");

map.put("ccc", "333");

//遍历map 有几种遍历方式 两种

// 1、获取所有的key，通过key得到value 使用get方法

// 2、获取key和value的关系

//使用第一种方式遍历

//获取所有的key

Set<String> sets = map.keySet();

//遍历所有key返回的set

for (String key : sets) {

//通过key得到value

String value = map.get(key);

System.out.println(key+" : "+value);

}

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

//得到key和value的关系

Set<Entry<String, String>> sets1 = map.entrySet();

//遍历sets1

for (Entry<String, String> entry : sets1) {

//entry是key和value关系

String keyv = entry.getKey();

String valuev = entry.getValue();

System.out.println(keyv+" : "+valuev);

}

}