# 一、java基础加强

jdk5开始提供的一些java中的新特性。

1.静态导入

import static ...

2.自动封箱拆箱

java中的八种基本数据类型都提供了对应的包装类型。jdk5以后，在需要的时候会自动在基本类型和对应的包装类型之间进行转换。

3. 增强for循环

只有实现了iterator接口的对象才能增强for

只能遍历不能修改！

可变参数

可变参数在方法调用时可以被赋予多个值

而对于方法本身，可变参数可以当做一个数组来处理

可变参数只能位于参数列表的最后，这也导致了一个方法中最多只能有一个可变参数。

枚举

枚举可以自定义构造方法、成员变量和成员方法。

枚举也可以包含抽象方法。

# 反射

什么时反射：

剖析类，获取类的信息，对类的信息进行操作的过程称为对类的反射。

Class 代表类（字节码）的类

获取类的方式：

（1）.可以通过 类名.class 来获取该类的Class对象

（2）.可以通过 类的对象.getClass() 方法来获取Class对象

（3）.可以通过Class.forName("类的全路径名");来获取Class对象

常用API：

T cast(Object obj)

ClassLoader getClassLoader()

Constructor<T> getConstructor(Class<?>... parameterTypes)

Constructor<T> getDeclaredConstructor(Class<?>... parameterTypes)

Constructor<?>[] getConstructors()

Constructor<?>[] getDeclaredConstructors()

Field getField(String name)

Field[] getFields()

Field getDeclaredField(String name)

Field[] getDeclaredFields()

Method getMethod(String name, Class<?>... parameterTypes)

Method getDeclaredMethod(String name, Class<?>... parameterTypes)

Method[] getMethods()

Method[] getDeclaredMethods()

Class<?>[] getInterfaces()

boolean isAssignableFrom(Class<?> cls)

boolean isInterface()

newInstance()//想要调用此方法，必须保证，该Class类中必须有无参的公有构造。

Field 代表成员变量的类 ： setAccessible(true);//暴力反射可以操作 私有成员

String getName()

Object get(Object obj)

boolean getBoolean(Object obj)

byte getByte(Object obj)

char getChar(Object obj)

double getDouble(Object obj)

float getFloat(Object obj)

int getInt(Object obj)

long getLong(Object obj)

short getShort(Object obj)

void set(Object obj, Object value)

void setBoolean(Object obj, boolean z)

void setByte(Object obj, byte b)

void setChar(Object obj, char c)

void setDouble(Object obj, double d)

void setFloat(Object obj, float f)

void setInt(Object obj, int i)

void setLong(Object obj, long l)

void setShort(Object obj, short s)

Method 代表成员方法的类

Class<?> getDeclaringClass()

Class<?>[] getExceptionTypes()

String getName()

Class<?>[] getParameterTypes()

Class<?> getReturnType()

Object invoke(Object obj, Object... args)

Constructor 代表构造方法的类

String getName()

Class<?>[] getExceptionTypes()

Class<?>[] getParameterTypes()

T newInstance(Object... initargs)

注解

泛型

内省

动态代理

# 二、Map

Map是一个独立的体系结构。可以存储键值对应信息的存储结构。键不能重复，一个键最多对应一个值。

Map

|--HashTable -- 利用hash算法实现的Map，是同步的，线程安全的，性能比较差。不允许使用null作为键作为值。

|--Properties 可以作为配置文件来使用，可以通过流写入或写出

|--HashMap -- 利用hash算法实现的Map，不是同步的，线程不安全，性能高（如果需要同步，可以自己处理同步，或者通过Collections.synchronizedMap装饰为带同步的Map）。允许使用null作为键 null作为值。

，因为hash算法具有散列的特点，所以HashMap本身是无序的。这种无需体现在取出顺序和存入顺序的不同，也体现在，不能保证取出顺序永久不变。注意的一点是，HashMap通过 hashCode 和 equals 方法来判断两个键是否是同一个。

|--LinkedHashMap -- 在HashMap的基础上增加了一个双向链表，利用此双向链表保存数据存入的顺序，从而实现了可以按序获取获取数据。

|--TreeMap -- 利用二叉树算法实现的Map，可以指定方式实现排序。想要实现排序的对象必须实现Comparable接口

常用API：

设置数据

V put(K key, V value)

void putAll(Map<? extends K,? extends V> m)

获取数据

V get(Object key)

Set<K> keySet()

entrySet()

获取信息

boolean containsKey(Object key)

boolean containsValue(Object value)

boolean isEmpty()

int size()

Collection<V> values()

清除数据

V remove(Object key)

void clear()

利用Map存储对象，自定义hashCode和equals

我们可以通过重写对象的equals和hashCode方法来控制什么情况下使HashMap将对象认为是同一个键。