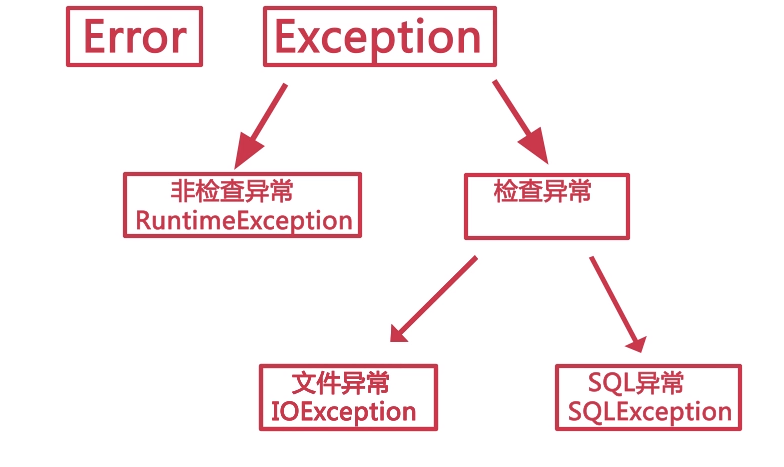
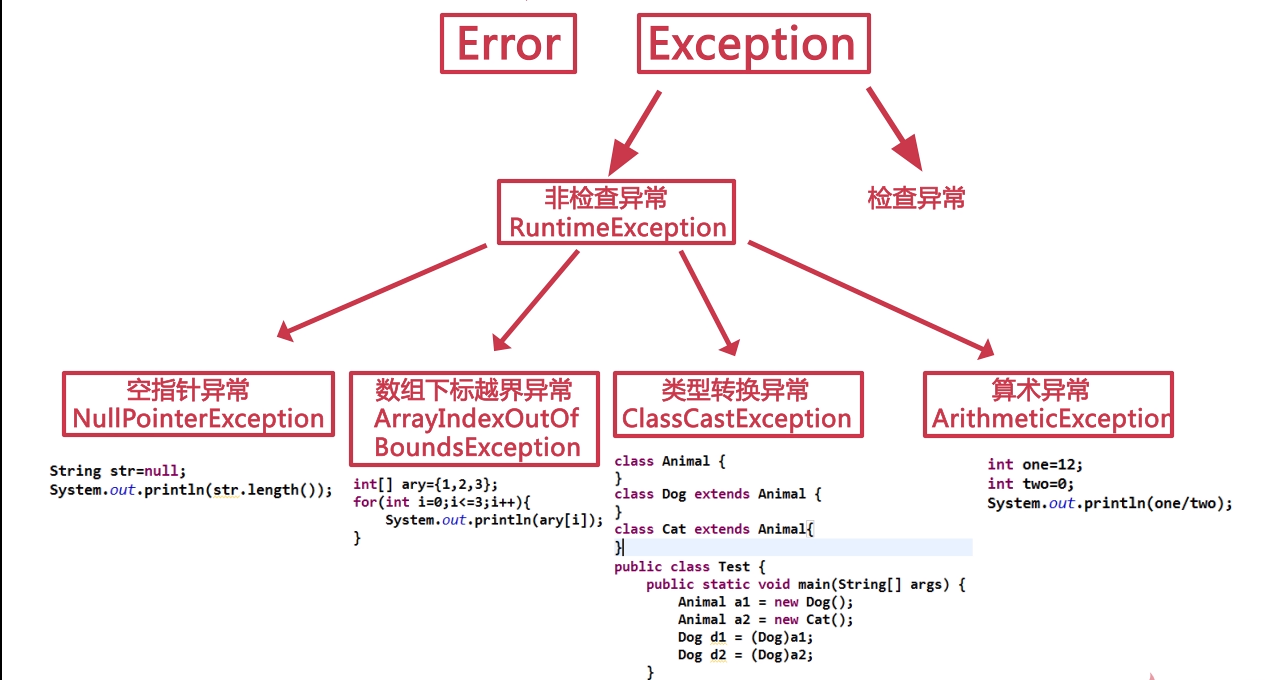
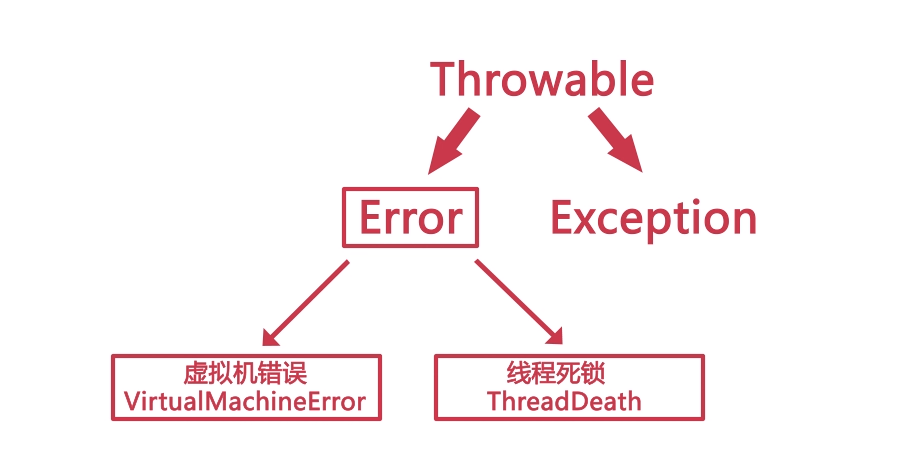
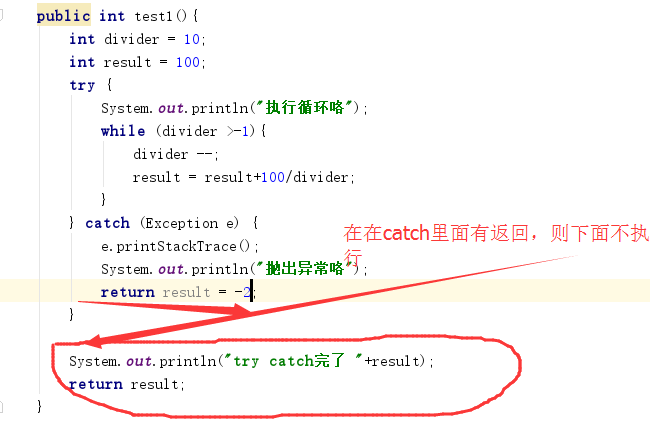
第一章：异常和异常处理

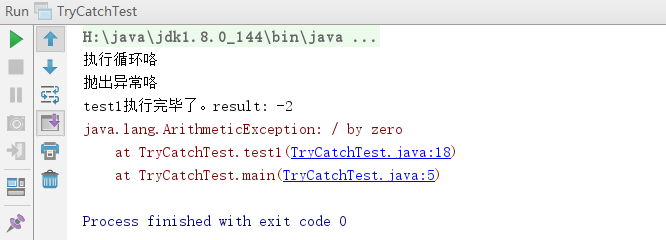
http://blog.csdn.net/jackfrued/article/details/44921941/

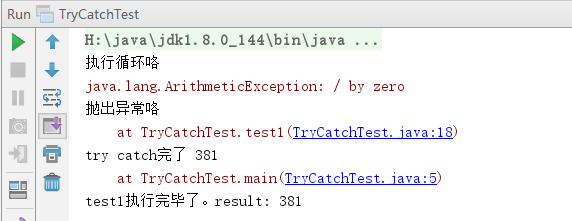
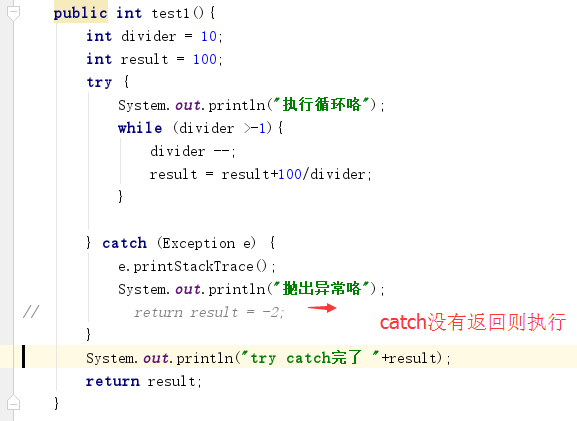
1-1：Java中异常简介：



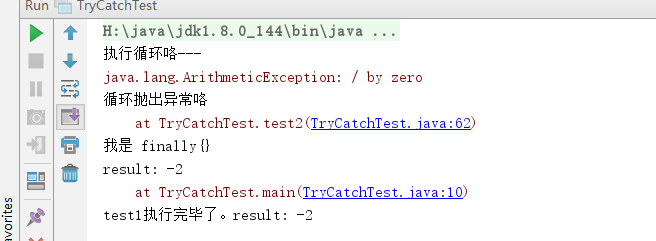
1-2:使用try…catch …finally 捕获异常：

情况1：

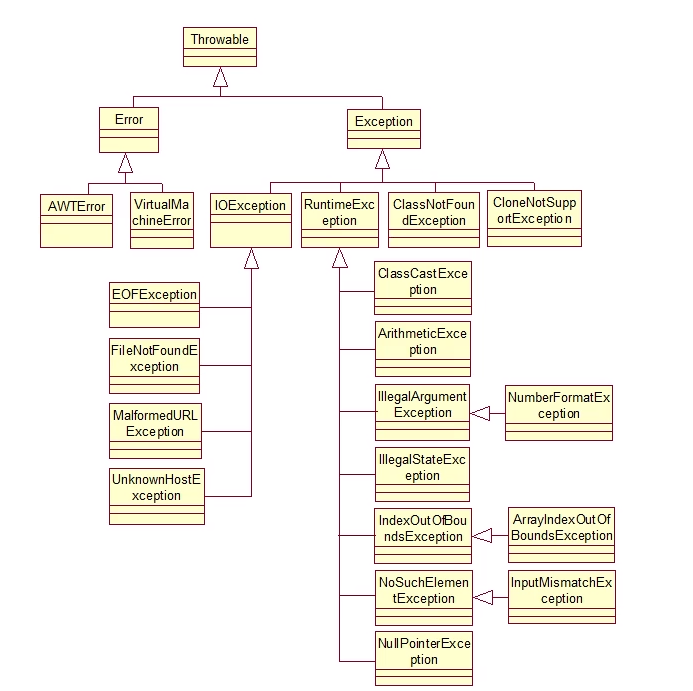
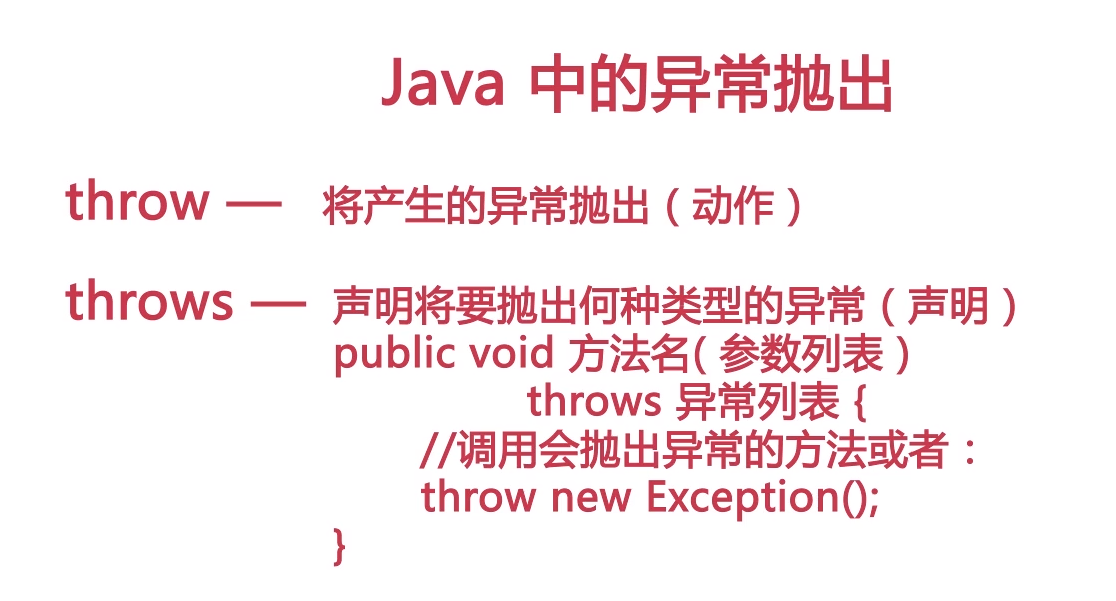




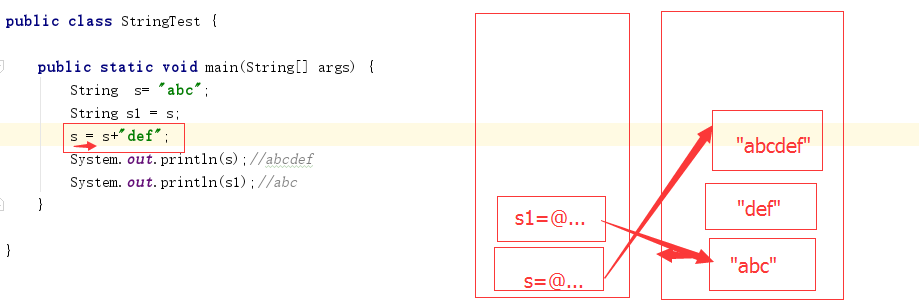
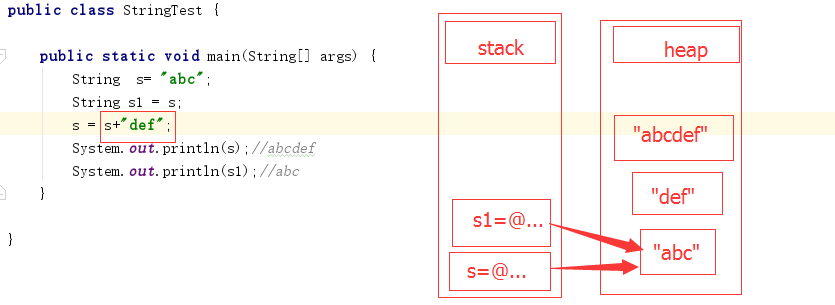
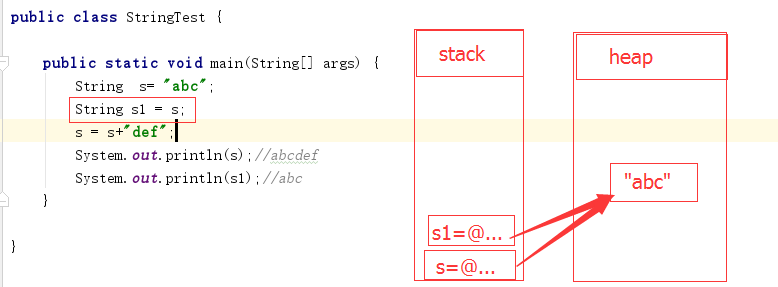
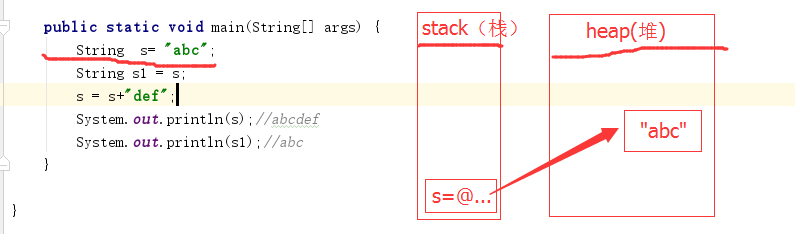
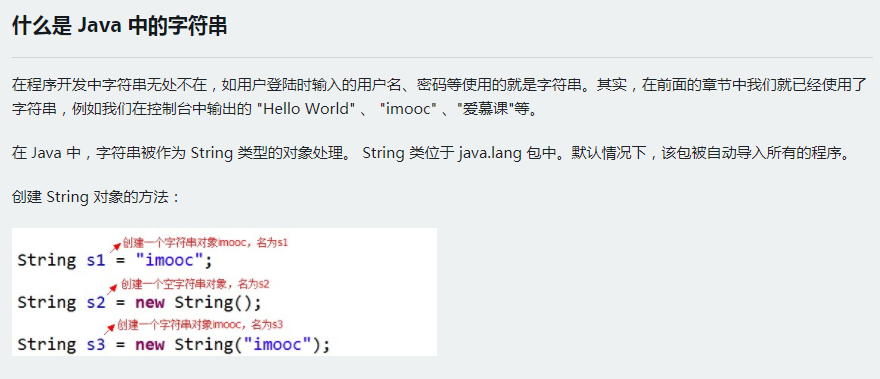
情况二：try{} catch(){} finally{}



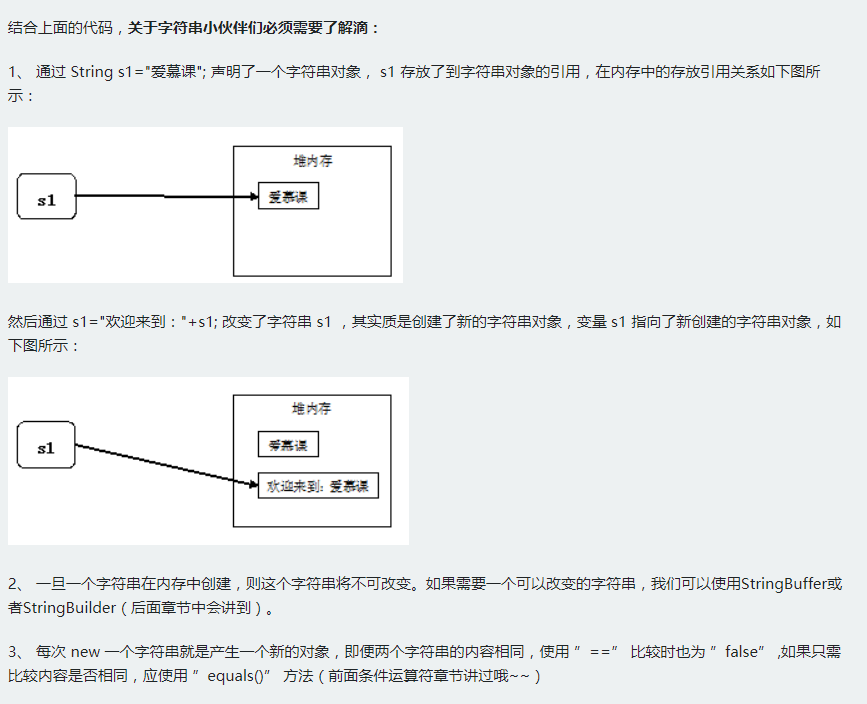
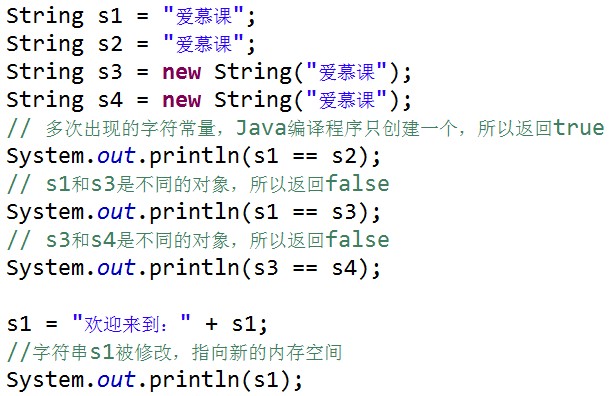
1-4:Java中抛出异常：



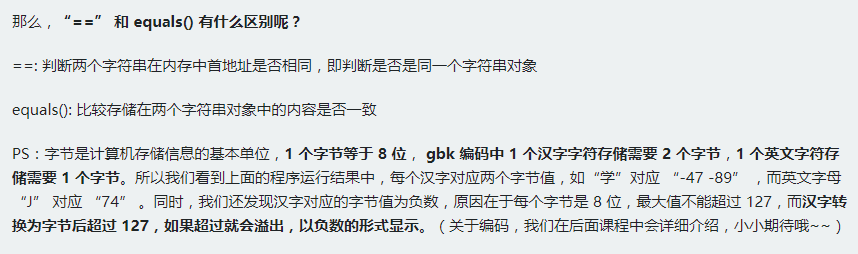
第二章：Java字符串：



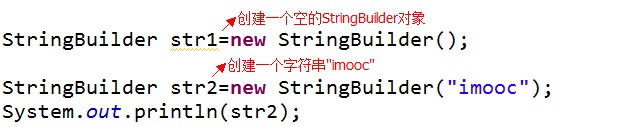
2-2：Java字符串的不变性



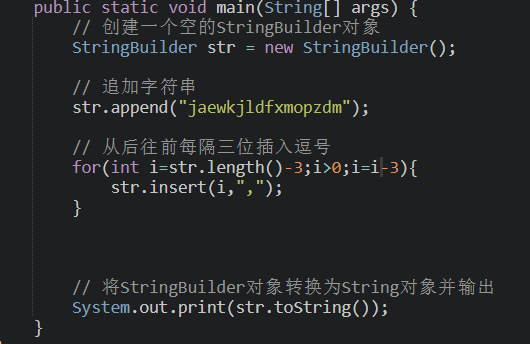
2-3：Java中String 类常用方法：



2-3：Java中的StringBuilder类

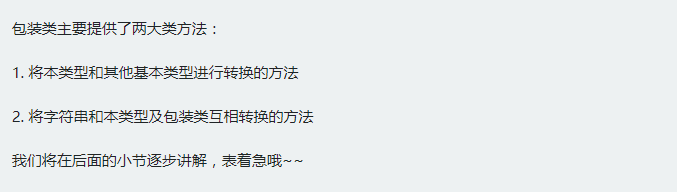
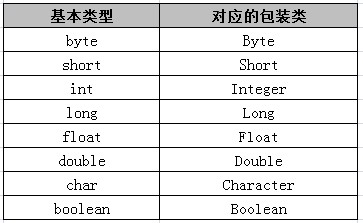
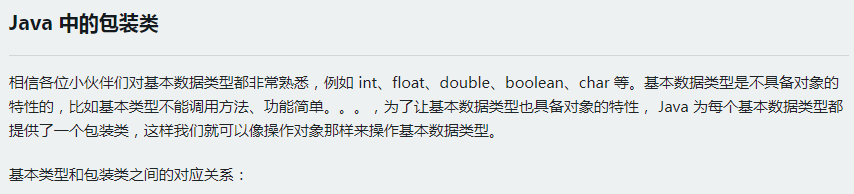


2-5：Java中的StringBuilder类：



第三章：Java常用的类：

3-1：java中的包装类：

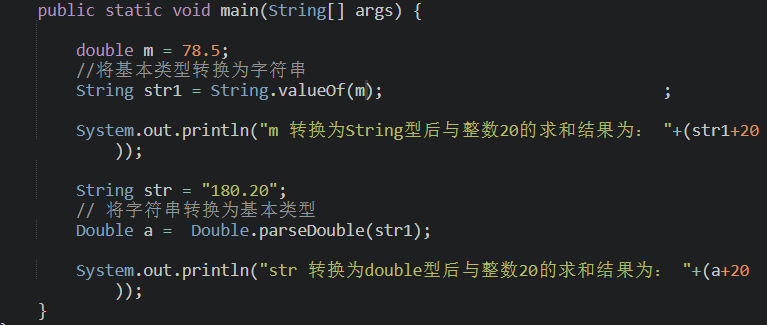


3-2：Java中基本类型和包装类之间的转换：

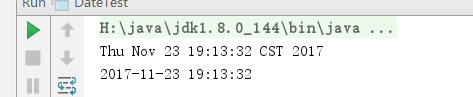
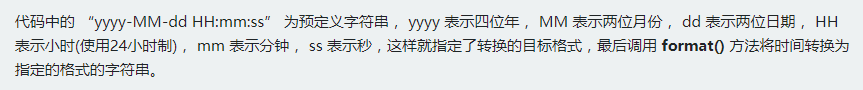
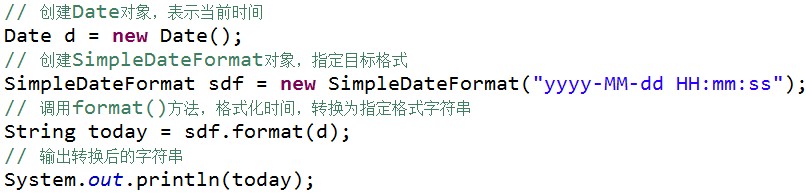


3-3：Java中基本类型和字符串之间的转换：

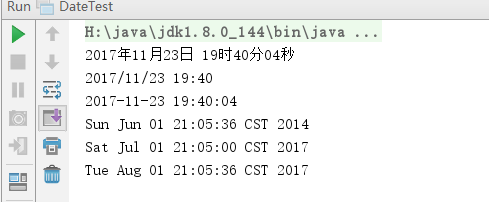
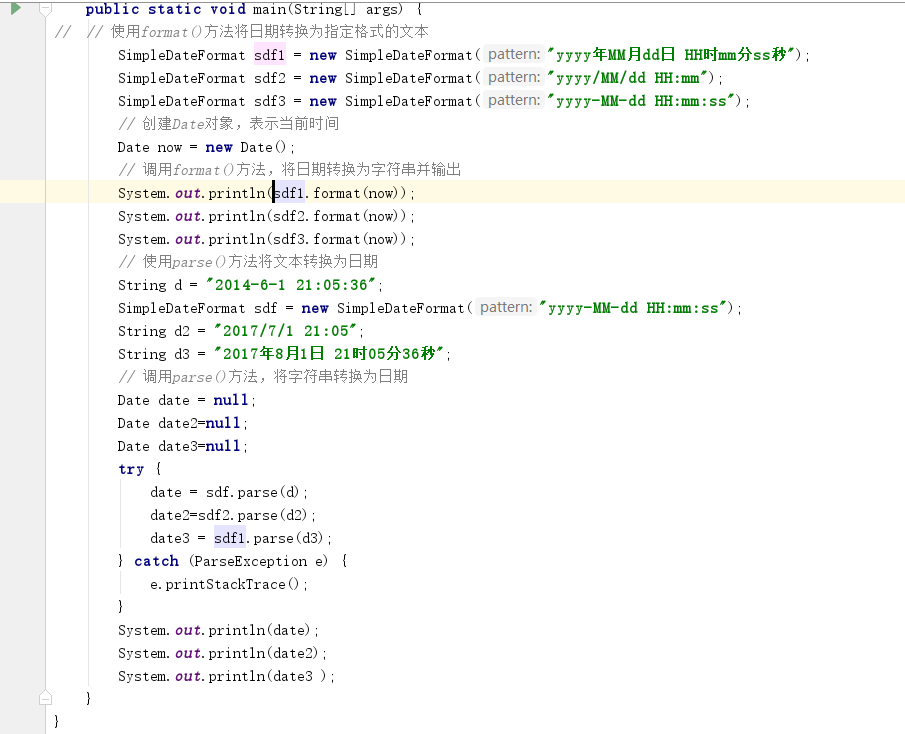
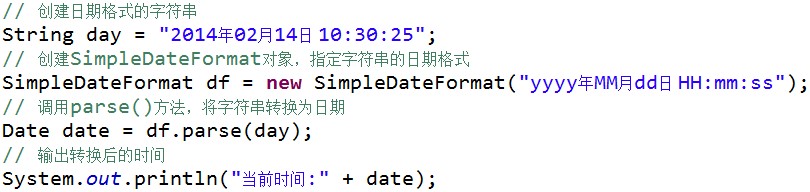




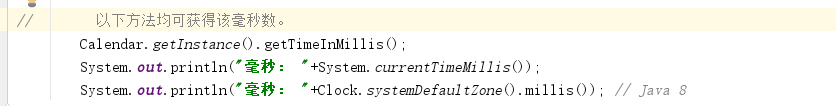
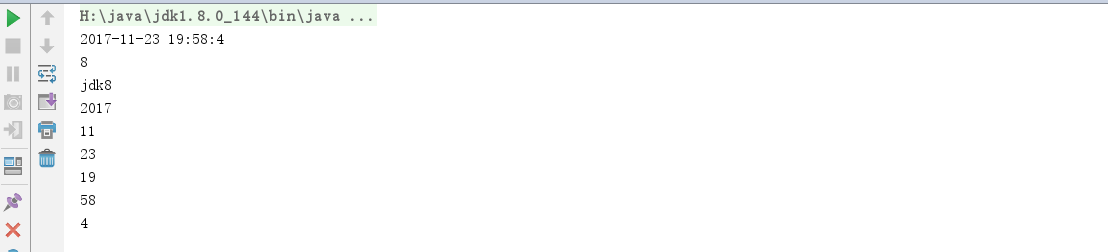
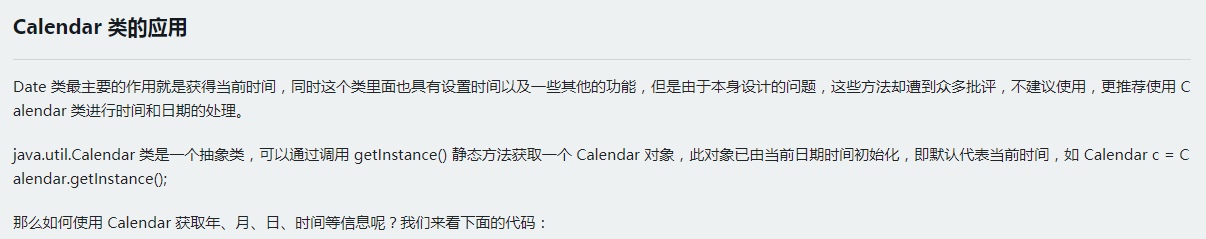
3-4：使用Date 和SimpleDateFormat:



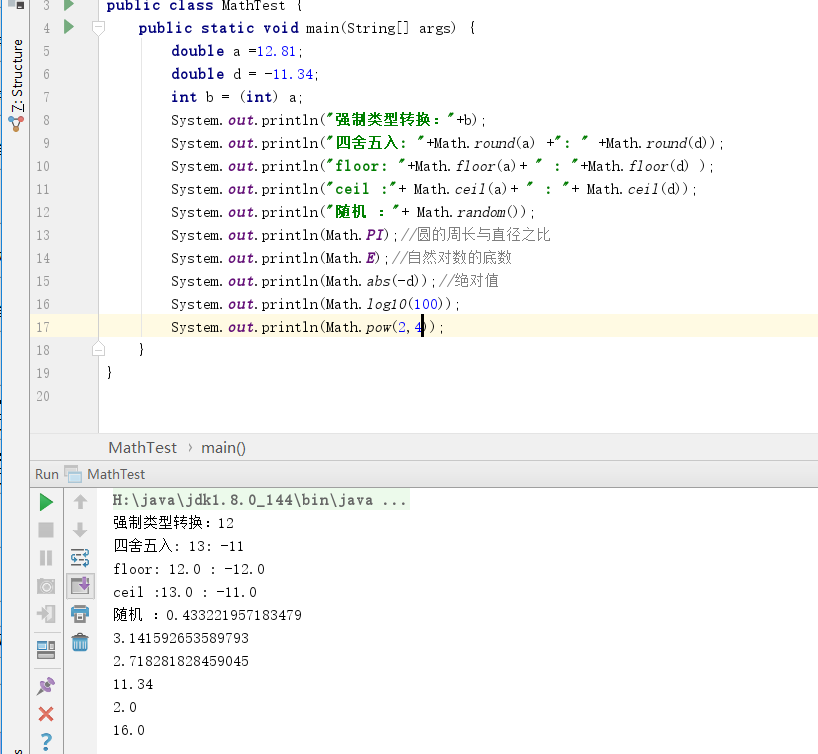
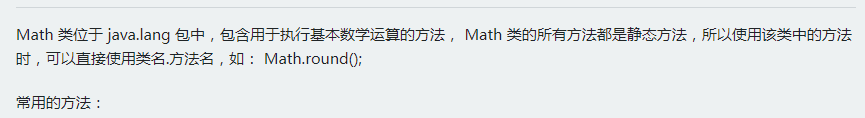




3-5:Calendar类的应用：

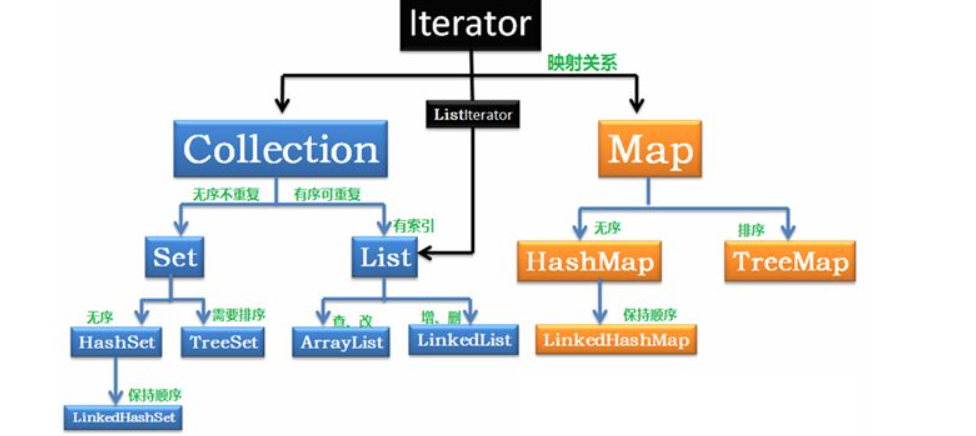
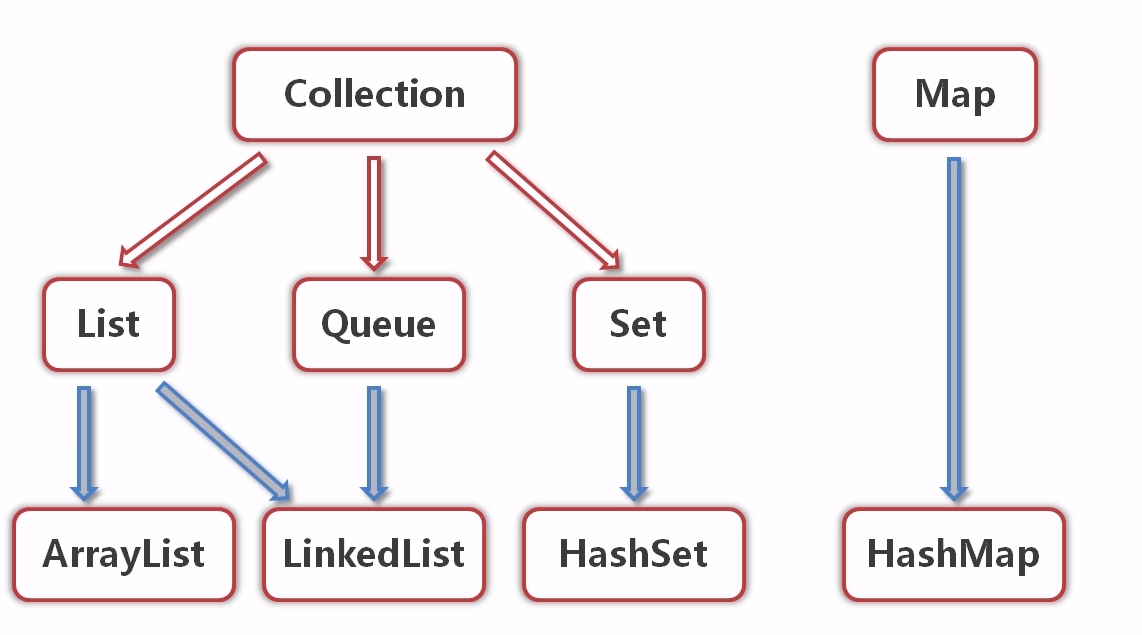
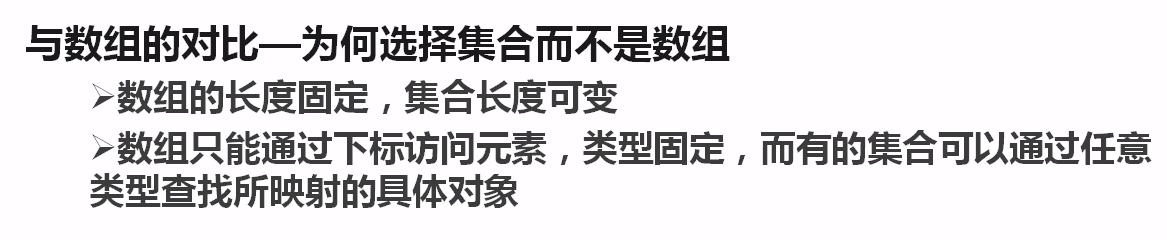
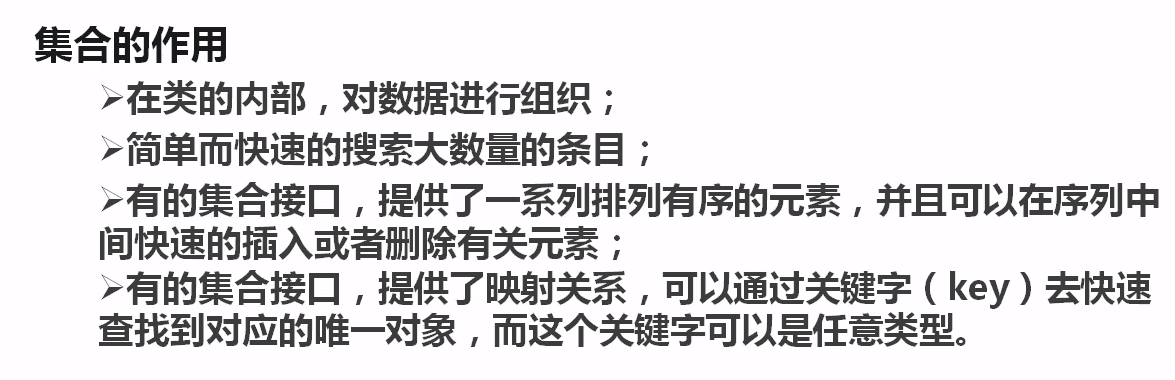
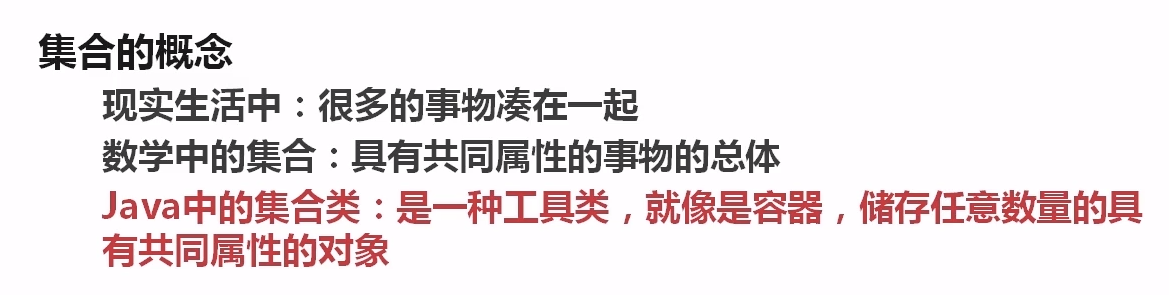


3-6：Java使用Math类型操作数据：

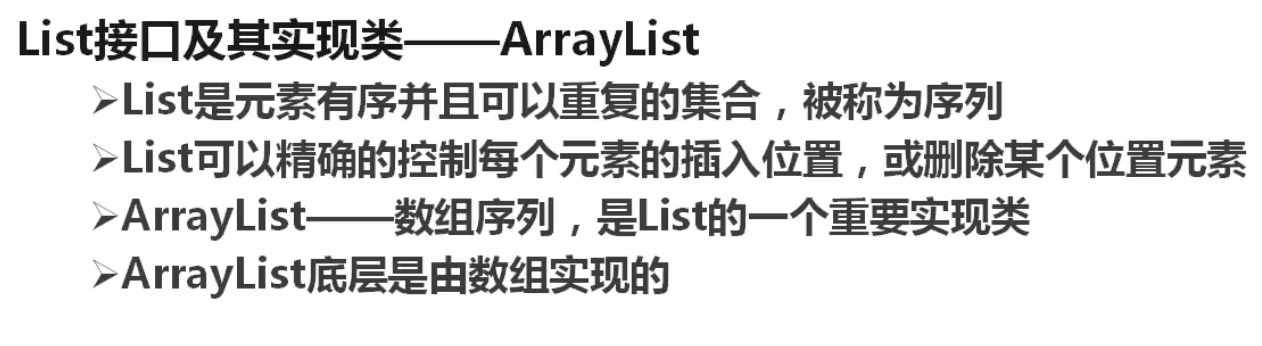


第四章：集合框架

4-1：



4-2：Collection接口 &list 接口



小案例：

**public class** Course {  
 **private** String **id**;  
 **private** String **name**;  
  
 **public** Course() {  
 }  
  
 **public** Course(String id, String name) {  
 **this**.**id** = id;  
 **this**.**name** = name;  
 }  
  
 **public** String getId() {  
 **return id**;  
 }  
  
 **public void** setId(String id) {  
 **this**.**id** = id;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Course{"** +  
 **"id='"** + **id** + **'\''** +  
 **", name='"** + **name** + **'\''** +  
 **'}'**;  
 }

**public class** Student {  
 **private** String **id**;  
 **private** String **name**;  
 **private** Set<Course> **courses**;  
  
 **public** Student() {  
  
 }  
  
 **public** Student(String id, String name) {  
 **this**.**id** = id;  
 **this**.**name** = name;  
 **this**.**courses** = **new** HashSet<>();  
 }  
  
 **public** Student(String id, String name, Set<Course> courses) {  
 **this**.**id** = id;  
 **this**.**name** = name;  
 **this**.**courses** = courses;  
 }  
  
 **public** String getId() {  
 **return id**;  
 }  
  
 **public void** setId(String id) {  
 **this**.**id** = id;  
 }  
  
 **public** String getName() {  
 **return name**;  
 }  
  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
  
 **public** Set<Course> getCourses() {  
 **return courses**;  
 }  
  
 **public void** setCourses(Set<Course> courses) {  
 **this**.**courses** = courses;  
 }  
}

**public class** ListTest {  
  
 **private** List **coursesToSelect**;  
  
 **public** ListTest() {  
 **this**.**coursesToSelect** = **coursesToSelect** = **new** ArrayList<Course>();  
 }  
  
 **public** List getCoursesToSelect() {  
 **return coursesToSelect**;  
 }  
  
 **public void** setCoursesToSelect(List coursesToSelect) {  
 **this**.**coursesToSelect** = coursesToSelect;  
 }  
 */\*\*  
 \* \*/* **public void** addCourses(){  
 String[] str = {**"数据结构"**,**"java"**,**"高等数学"**,**"线性代数"**,**"PHP"**,**"Oracle"**,**"Mysql"**,**"HTML"**,**"Jquery"**,**"StartBoot"**};  
  
 **for** (**int** i=0; i<str.**length**;i++){  
 **coursesToSelect**.add(**new** Course(String.*valueOf*(i+1),str[i]));  
 System.***out***.println(**"增加了课程："**+(i+1)+**" :"**+str[i]);  
 }  
  
 *//指定位置：添加元素* **coursesToSelect**.add(0,**new** Course(**"16"**,**"大学体育"**));  
 }

**/\*\*元素的修改\*/**

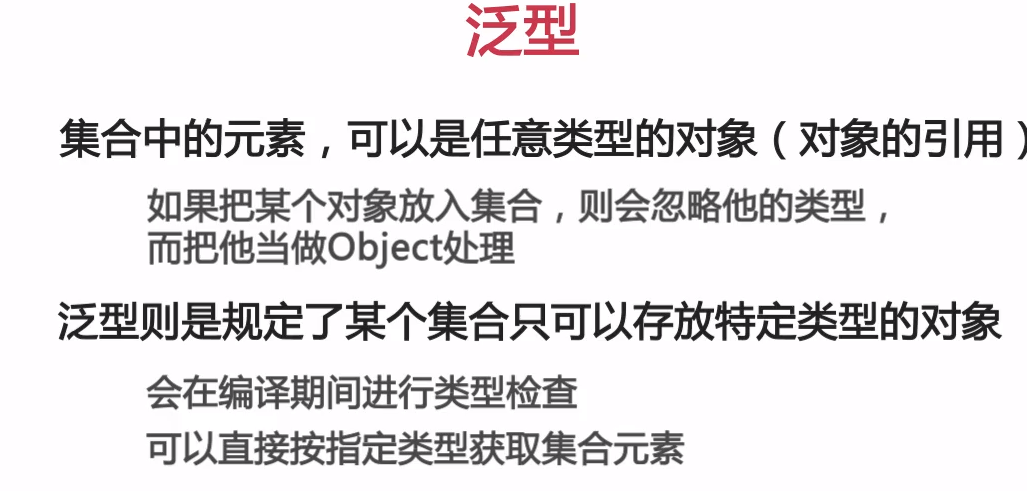
**public void** updateCourses(){  
 **coursesToSelect**.set(1,**new** Course(**"1"**,**"毛概"**));  
}

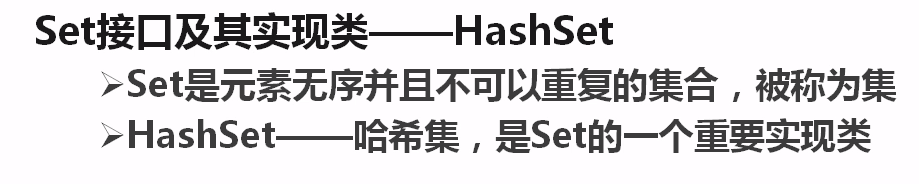
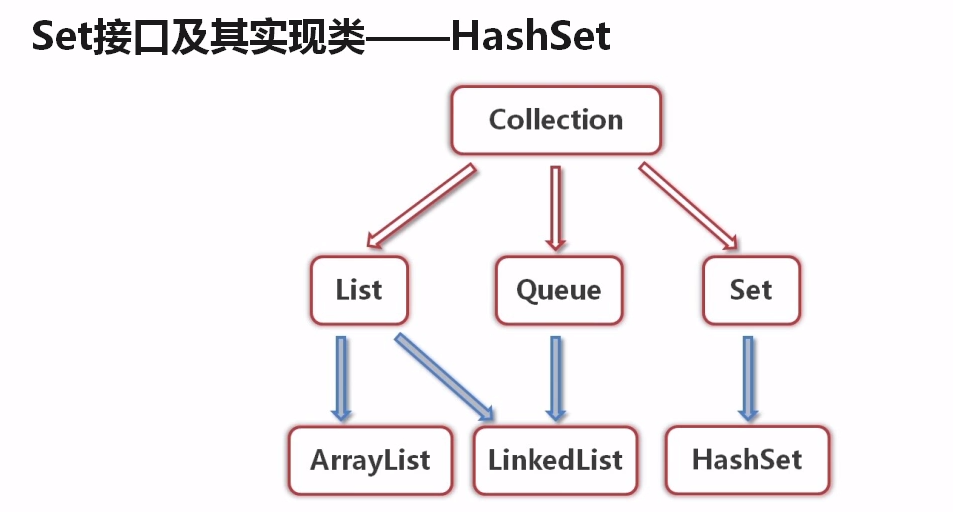
**public void** addCoursesByArray(){  
*// 数组方式添加到集合里面：* Course[] courses = {**new** Course(**"12"**,**"系统结构"**),**new** Course(**"11"**,**"网络原理"**)};  
 **coursesToSelect**.addAll(Arrays.*asList*(courses));  
 Course[] courses2 = {**new** Course(**"13"**,**"大学英语"**),**new** Course(**"14"**,**"大学物理"**)};  
 **coursesToSelect**.addAll(3,Arrays.*asList*(courses2));  
 }

/\*\*\*迭代器方式遍历\*\*/  
**public void** getByIterator(){  
 Iterator it = **coursesToSelect**.iterator();  
 **while** (it.hasNext()){  
  
 System.***out***.println(it.next().toString());  
 }  
}

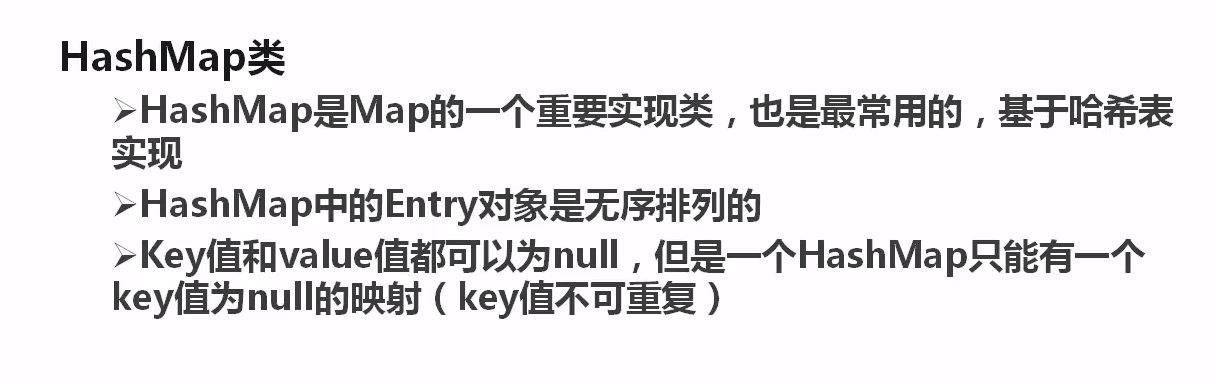
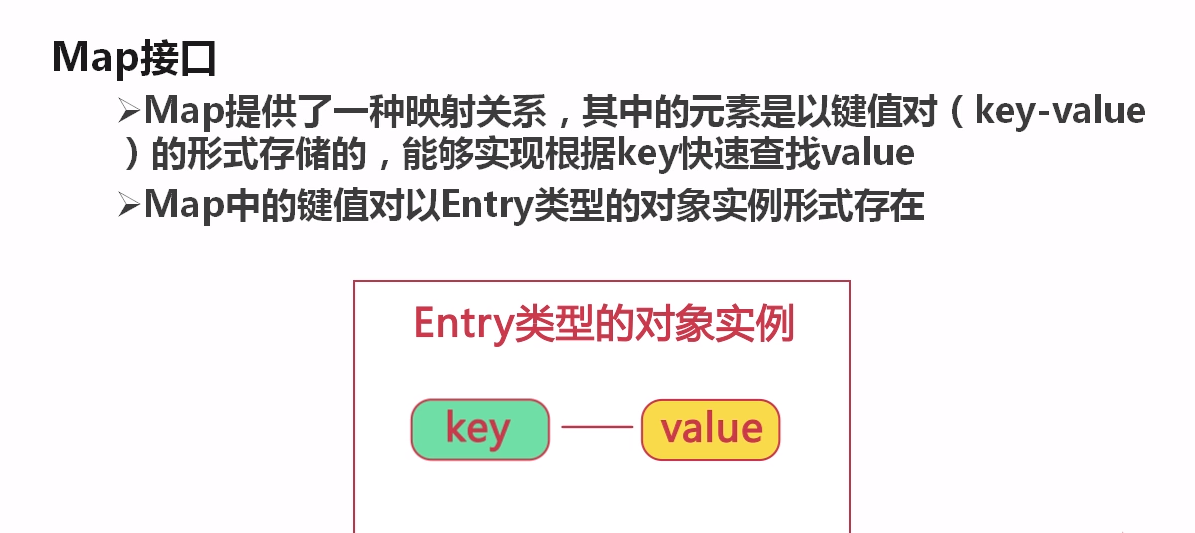
**public static void** main(String[] args) {  
 ListTest lt = **new** ListTest();  
 lt.addCourses();  
 lt.addCoursesByArray();  
 List<Course> courses = lt.getCoursesToSelect();  
  
*// for(Course:courses){  
// System.out.println(courses.toString());  
// }* **int** i = 0;  
 **for** (Course c:courses) {  
 System.***out***.println(i+**"--"**+c.toString());  
 i++;  
 }  
  
  
 }







第五章：map接口：hashMap

package com.imooc.collection;

import java.util.HashMap;

import java.util.Map;

import java.util.Map.Entry;

import java.util.Scanner;

import java.util.Set;

public class MapTest {

/\*\*

\* 用来承装学生类型对象

\*/

public Map<String, Student> students;

/\*\*

\* 在构造器中初始化students属性

\*/

public MapTest() {

this.students = new HashMap<String, Student>();

}

/\*\*

\* 测试添加：输入学生ID，判断是否被占用 若未被占用，则输入姓名，创建新学生对象，并且 添加到students中

\*/

public void testPut() {

// 创建一个Scanner对象，用来获取输入的学生ID和姓名

Scanner console = new Scanner(System.in);

int i = 0;

while (i < 3) {

System.out.println("请输入学生ID：");

String ID = console.next();

// 判断该ID是否被占用

Student st = students.get(ID);

if (st == null) {

// 提示输入学生姓名

System.out.println("请输入学生姓名：");

String name = console.next();

// 创建新的学生对象

Student newStudent = new Student(ID, name);

// 通过调用students的put方法，添加ID-学生映射

students.put(ID, newStudent);

System.out.println("成功添加学生：" + students.get(ID).name);

i++;

} else {

System.out.println("该学生ID已被占用！");

continue;

}

}

}

/\*\*

\* 测试Map的keySet方法

\*/

public void testKeySet() {

// 通过keySet方法，返回Map中的所有“键”的Set集合

Set<String> keySet = students.keySet();

// 取得students的容量

System.out.println("总共有：" + students.size() + "个学生！");

// 遍历keySet，取得每一个键，再调用get方法取得每个键对应的value

for (String stuId : keySet) {

Student st = students.get(stuId);

if (st != null)

System.out.println("学生：" + st.name);

}

}

/\*\*

\* 测试删除Map中的映射

\*/

public void testRemove() {

// 获取从键盘输入的待删除学生ID字符串

Scanner console = new Scanner(System.in);

while (true) {

// 提示输入待删除的学生的ID

System.out.println("请输入要删除的学生ID！");

String ID = console.next();

// 判断该ID是否有对应的学生对象

Student st = students.get(ID);

if (st == null) {

// 提示输入的ID并不存在

System.out.println("该ID不存在！");

continue;

}

students.remove(ID);

System.out.println("成功删除学生：" + st.name);

break;

}

}

/\*\*

\* 通过entrySet方法来遍历Map

\*/

public void testEntrySet() {

// 通过entrySet方法，返回Map中的所有键值对

Set<Entry<String, Student>> entrySet = students.entrySet();

for (Entry<String, Student> entry : entrySet) {

System.out.println("取得键：" + entry.getKey());

System.out.println("对应的值为：" + entry.getValue().name);

}

}

/\*\*

\* 利用put方法修改Map中的已有映射

\*/

public void testModify() {

// 提示输入要修改的学生ID

System.out.println("请输入要修改的学生ID：");

// 创建一个Scanner对象，去获取从键盘上输入的学生ID字符串

Scanner console = new Scanner(System.in);

while (true) {

// 取得从键盘输入的学生ID

String stuID = console.next();

// 从students中查找该学生ID对应的学生对象

Student student = students.get(stuID);

if (student == null) {

System.out.println("该ID不存在！请重新输入！");

continue;

}

// 提示当前对应的学生对象的姓名

System.out.println("当前该学生ID，所对应的学生为：" + student.name);

// 提示输入新的学生姓名，来修改已有的映射

System.out.println("请输入新的学生姓名：");

String name = console.next();

Student newStudent = new Student(stuID, name);

students.put(stuID, newStudent);

System.out.println("修改成功！");

break;

}

}

/\*\*

\* @param args

\*/

public static void main(String[] args) {

MapTest mt = new MapTest();

mt.testPut();

mt.testKeySet();

// mt.testRemove();

// mt.testEntrySet();

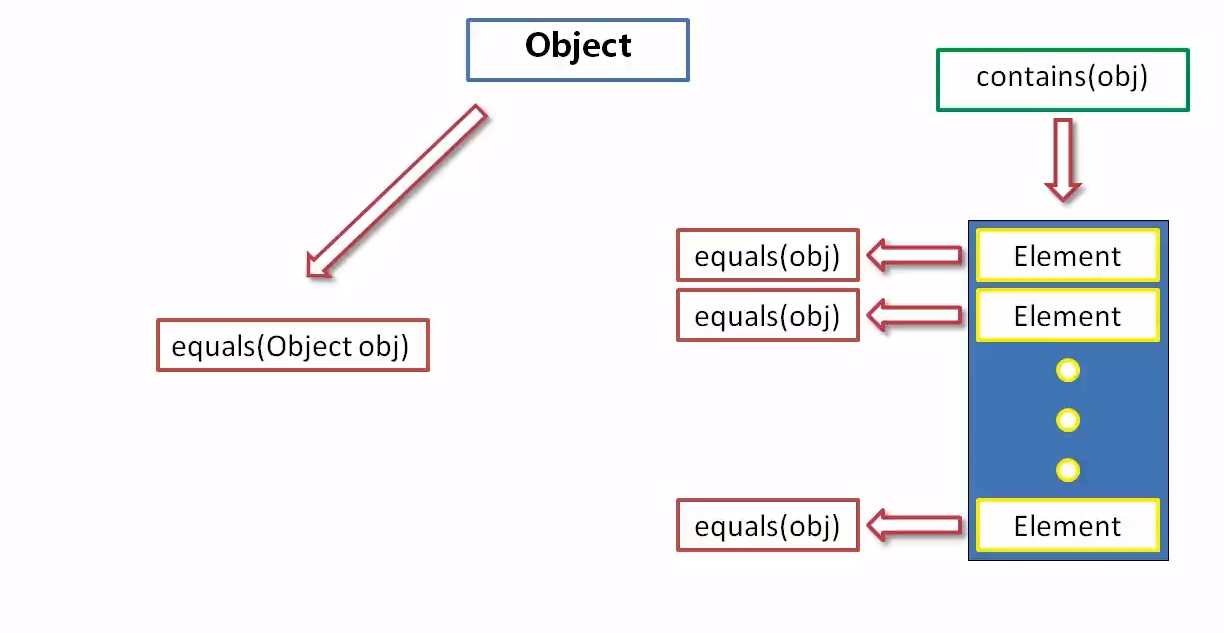
// mt.testModify();

// mt.testEntrySet();

}

}

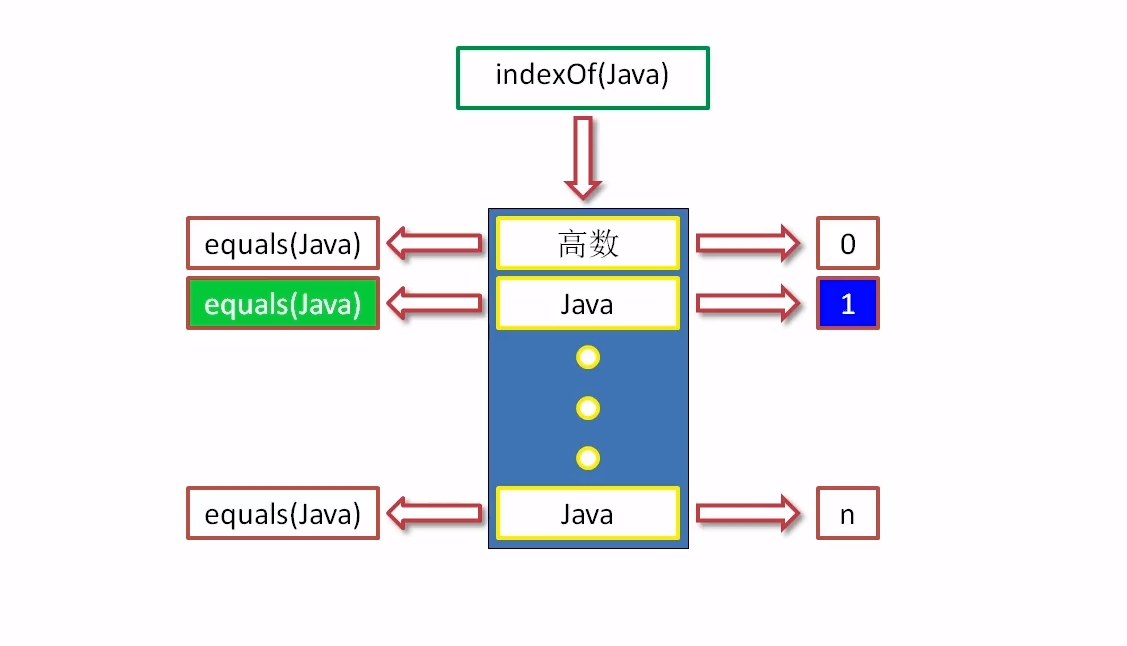
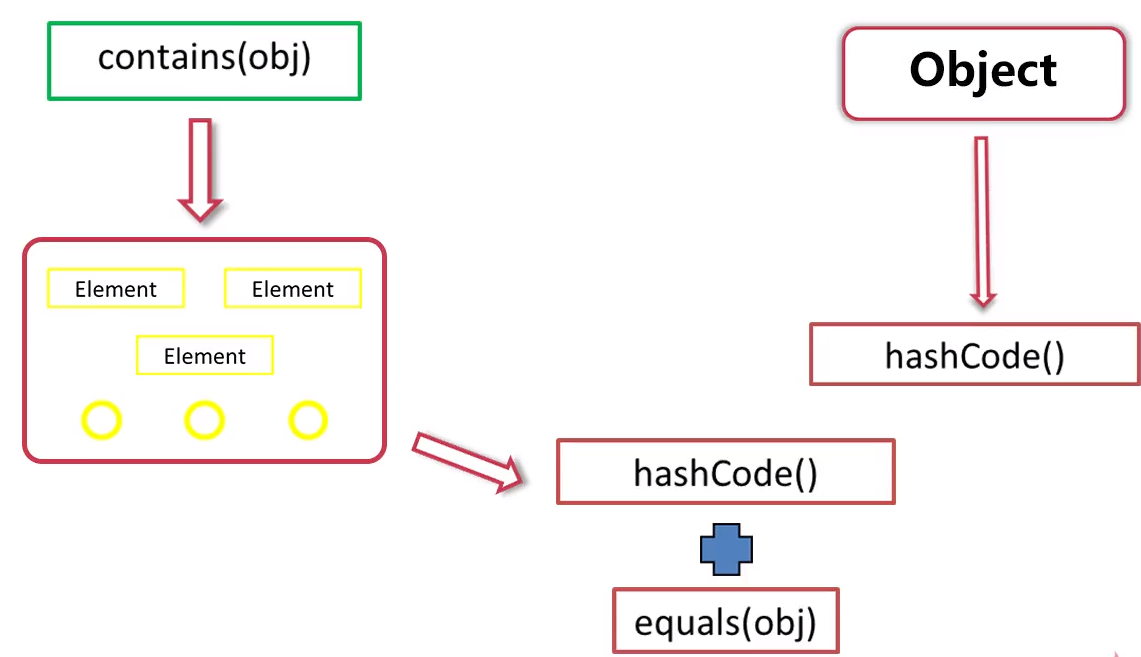
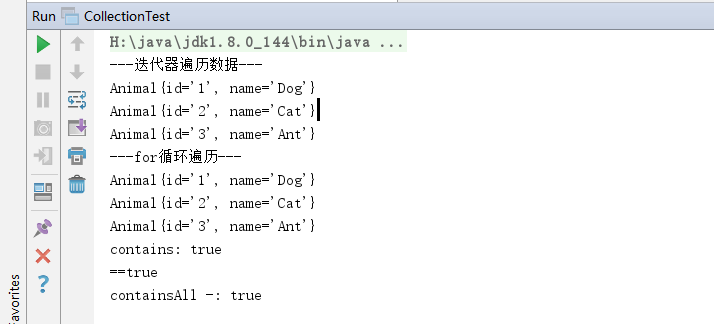
包含问题：





Map

**package** com.collection;  
  
**import** java.util.ArrayList;  
**import** java.util.Iterator;  
**import** java.util.List;  
  
**class** Animal{  
 **private** String **id**;  
 **private** String **name**;  
  
 **public** Animal(){}  
 **public** Animal(String id, String name) {  
 **this**.**id** = id;  
 **this**.**name** = name;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Animal{"** +  
 **"id='"** + **id** + **'\''** +  
 **", name='"** + **name** + **'\''** +  
 **'}'**;  
 }  
  
 @Override  
 **public boolean** equals(Object o) {  
 **if** (**this** == o) **return true**;  
 **if** (o == **null** || getClass() != o.getClass()) **return false**;  
  
 Animal animal = (Animal) o;  
  
 **if** (**id** != **null** ? !**id**.equals(animal.**id**) : animal.**id** != **null**) **return false**;  
 **return name** != **null** ? **name**.equals(animal.**name**) : animal.**name** == **null**;  
 }  
  
 @Override  
 **public int** hashCode() {  
 **int** result = **id** != **null** ? **id**.hashCode() : 0;  
 result = 31 \* result + (**name** != **null** ? **name**.hashCode() : 0);  
 **return** result;  
 }  
}  
**public class** CollectionTest {  
  
 **private** List<Animal> **animals** ;  
  
 */\*\*\*  
 \* 构造方法初始化 list  
 \* \*\*/* **public** CollectionTest(){  
 **animals** = **new** ArrayList<>();  
 }  
  
 */\*\*添加\*\*\*/* **public void** addAnimal(){  
 Animal[] animals = {**new** Animal(**"1"**,**"Dog"**),**new** Animal(**"2"**,**"Cat"**),**new** Animal(**"3"**,**"Ant"**)};  
 **for** (**int** i=0;i<animals.**length**;i++){  
 **this**.**animals**.add(animals[i]);  
 }  
 }  
*/\*\*  
 \* 迭代器遍历集合  
 \*  
 \* \*/* **public void** getByIterator(){  
 System.***out***.println(**"---迭代器遍历数据---"**);  
 Iterator it = **animals**.iterator();  
 **while** (it.hasNext()){  
 System.***out***.println(it.next().toString());  
 }  
 }  
  
 **public void** getByfor(){  
 System.***out***.println(**"---for循环遍历---"**);  
 **for** (**int** i=0;i<**animals**.size();i++){  
 System.***out***.println(**animals**.get(i).toString());  
 }  
 }  
 */\*\*  
 \* 如果此 collection 包含指定的元素，则返回 true。  
 \* \*/* **public void** testContains(){  
 Animal dog = **new** Animal(**"1"**,**"Dog"**);  
*// 需要重写animal de equals方法和 hashCode方法，这样才好比较是true* **boolean** ishas = **animals**.contains(dog);  
 System.***out***.println(**"contains: "**+ishas);  
*//* Animal dogs = **animals**.get(0);  
 System.***out***.println(**"=="**+**animals**.contains(dogs));  
  
 }  
  
 **public void** testContainsAll(){  
 List<Animal> list = **new** ArrayList<>();  
 list.add(**new** Animal(**"2"**,**"Cat"**) );  
 list.add(**new** Animal(**"3"**,**"Ant"**) );  
  
 System.***out***.println(**"containsAll -: "**+**animals**.containsAll(list));  
 }  
  
 **public static void** main(String[] args) {  
  
 CollectionTest ct = **new** CollectionTest();  
 ct.addAnimal();  
 ct.getByIterator();  
 ct.getByfor();  
 ct.testContains();  
 ct.testContainsAll();  
 }  
}

**package** com.collection;  
  
**import** java.util.ArrayList;  
**import** java.util.Iterator;  
**import** java.util.List;  
  
**class** Animal{  
 **private** String **id**;  
 **private** String **name**;  
  
 **public** Animal(){}  
 **public** Animal(String id, String name) {  
 **this**.**id** = id;  
 **this**.**name** = name;  
 }  
  
 @Override  
 **public** String toString() {  
 **return "Animal{"** +  
 **"id='"** + **id** + **'\''** +  
 **", name='"** + **name** + **'\''** +  
 **'}'**;  
 }  
  
 @Override  
 **public boolean** equals(Object o) {  
 **if** (**this** == o) **return true**;  
 **if** (o == **null** || getClass() != o.getClass()) **return false**;  
  
 Animal animal = (Animal) o;  
  
 **if** (**id** != **null** ? !**id**.equals(animal.**id**) : animal.**id** != **null**) **return false**;  
 **return name** != **null** ? **name**.equals(animal.**name**) : animal.**name** == **null**;  
 }  
  
 @Override  
 **public int** hashCode() {  
 **int** result = **id** != **null** ? **id**.hashCode() : 0;  
 result = 31 \* result + (**name** != **null** ? **name**.hashCode() : 0);  
 **return** result;  
 }  
}  
**public class** CollectionTest {  
  
 **private** List<Animal> **animals** ;  
  
 */\*\*\*  
 \* 构造方法初始化 list  
 \* \*\*/* **public** CollectionTest(){  
 **animals** = **new** ArrayList<>();  
 }  
  
 */\*\*添加\*\*\*/* **public void** addAnimal(){  
 Animal[] animals = {**new** Animal(**"1"**,**"Dog"**),**new** Animal(**"2"**,**"Cat"**),**new** Animal(**"3"**,**"Ant"**),**new** Animal(**"1"**,**"Dog"**)};  
 **for** (**int** i=0;i<animals.**length**;i++){  
 **this**.**animals**.add(animals[i]);  
 }  
 }  
*/\*\*  
 \* 迭代器遍历集合  
 \*  
 \* \*/* **public void** getByIterator(){  
 System.***out***.println(**"---迭代器遍历数据---"**);  
 Iterator it = **animals**.iterator();  
 **while** (it.hasNext()){  
 System.***out***.println(it.next().toString());  
 }  
 }  
  
 **public void** getByfor(){  
 System.***out***.println(**"---for循环遍历---"**);  
 **for** (**int** i=0;i<**animals**.size();i++){  
 System.***out***.println(**animals**.get(i).toString());  
 }  
 }  
 */\*\*  
 \* 如果此 collection 包含指定的元素，则返回 true。  
 \* \*/* **public void** testContains(){  
 Animal dog = **new** Animal(**"1"**,**"Dog"**);  
*// 需要重写animal de equals方法和 hashCode方法，这样才好比较是true* **boolean** ishas = **animals**.contains(dog);  
 System.***out***.println(**"contains: "**+ishas);  
 System.***out***.println(**"indexOf: "**+**animals**.indexOf(dog));  
 System.***out***.println(**"lastindexOf: "**+**animals**.lastIndexOf(dog));  
*//* Animal dogs = **animals**.get(0);  
 System.***out***.println(**"=="**+**animals**.contains(dogs));  
  
 }  
  
 **public void** testContainsAll(){  
 List<Animal> list = **new** ArrayList<>();  
 list.add(**new** Animal(**"2"**,**"Cat"**) );  
 list.add(**new** Animal(**"3"**,**"Ant"**) );  
  
 System.***out***.println(**"containsAll -: "**+**animals**.containsAll(list));  
 }  
  
 **public static void** main(String[] args) {  
  
 CollectionTest ct = **new** CollectionTest();  
 ct.addAnimal();  
 ct.getByIterator();  
 ct.getByfor();  
 ct.testContains();  
 ct.testContainsAll();  
  
 }  
}



比较需要实现这2个中的一个接口咯；

案例：

