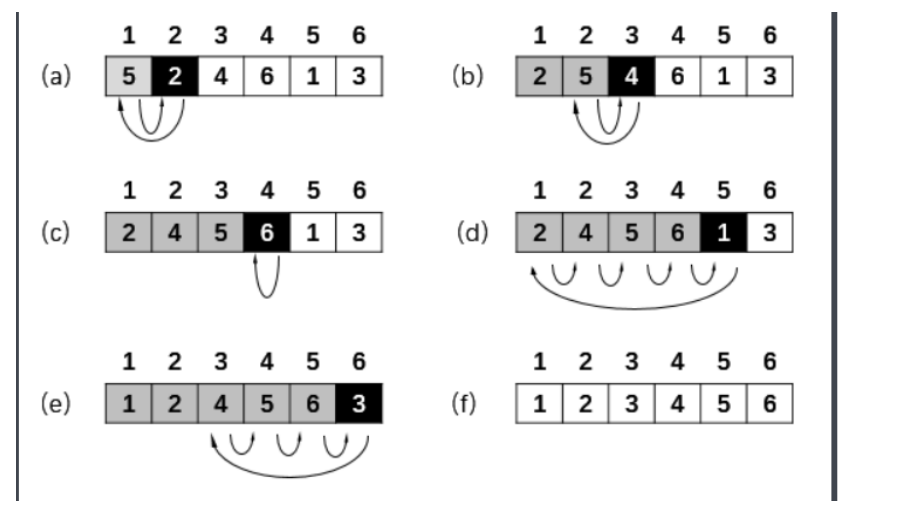
**题目1：插入排序过程如下，请编码实现插入排序**



**答案：**

|  |
| --- |
| import java.util.Arrays;  public class Test07\_InsertSort {  public static void main(String[] args) {  //准备一个int数组  int[] array = {5, 2, 6, 5, 9, 0, 3};  System.out.println("排序前: "+ Arrays.toString(array));  //插入排序  insertionSort(array);  //输出排序结果  System.out.println("排序后: "+ Arrays.toString(array));  }  public static void insertionSort(int[] arr) {  int len = arr.length;  if(len <= 1) {  return;  }  //外层循环控制 总体循环次数  for(int i = 1; i < len; i++) {  //内层循环做的事情：将无序列表中第一个元素插入到有序列表中  合适位置  int value = arr[i];  //获取有序列表中最后一个元素下标  int j = i - 1;  for(; j >= 0; j--) {  if(value < arr[j]) {  arr[j+1] = arr[j];  }else {  break;  }  }  //将需要插入的元素 放置到合适位置  arr[j+1] = value;  //一次排序完成后，输出 方便 观察  System.out.println(Arrays.toString(arr));  }  }  } |
|  |

**题目2：观察下面代码，写出程序输出结果，并分析为什么？**

|  |
| --- |
| public static void main(String[] args) {  Integer i1 = new Integer(127);  Integer i2 = new Integer(127);  System.out.println("i1 == i2: "+ (i1 == i2)); //?  Integer i3 = 127;  Integer i4 = 127;  System.out.println("i3 == i4: "+ (i3 == i4)); //?  Integer i5 = 128;  Integer i6 = 128;  System.out.println("i5 == i6: "+ (i5 == i6)); //?  } |

**答案：**

|  |
| --- |
| i1 == i2: false  i3 == i4: true  i5 == i6: false |

**题目3：static关键字都可以修饰什么，分别有什么作用？**

**答案：**

|  |
| --- |
| 修饰局部变量      static修饰局部变量时，使得被修饰的变量成为静态变量，存储在静态区。存储在静态区的数据生命周期与程序相同，在main函数之前初始化，在程序退出时销毁。（无论是局部静态还是全局静态    修饰全局变量  全局变量本来就存储在静态区，因此static并不能改变其存储位置。但是，static限制了其链接属性。被static修饰的全局变量只能被该包含该定义的文件访问（即改变了作用域）。    修饰函数       static修饰函数使得函数只能在包含该函数定义的文件中被调用。对于静态函数，声明和定义需要放在同一个文件夹中。    修饰成员变量        用static修饰类的数据成员使其成为类的全局变量，会被类的所有对象共享，包括派生类的对象，所有的对象都只维持同一个实例。 因此，static成员必须在类外进行初始化(初始化格式：int base::var=10;)，而不能在构造函数内进行初始化，不过也可以用const修饰static数据成员在类内初始化。    修饰成员函数       用static修饰成员函数，使这个类只存在这一份函数，所有对象共享该函数，不含this指针，因而只能访问类的static成员变量。静态成员是可以独立访问的，也就是说，无须创建任何对象实例就可以访问。例如可以封装某些算法，比如数学函数，如ln，sin，tan等等，这些函数本就没必要属于任何一个对象，所以从类上调用感觉更好，比如定义一个数学函数类Math，调用Math::sin(3.14);还可以实现某些特殊的设计模式 |