工程框架：采用工厂模式，各种全排列生成算法均实现了接口PermutationGeneration并且继承了Permutation基类。

其中接口PermutationGeneration里面有三类方法，分别为void genPermutation()，用于根据给定的初始序列按某种顺序生成之后的所有排列；void genPermutation(int pos)则是用于根据给定的初始序列按某种顺序生成之后的第pos个排列；void genAllPermutation(int per\_size)用于按照某种顺序生成per\_size大小的所有排列。Permutation基类包含了排列列表和排列长度属性，以及相关的初始化和输出操作。工厂类为PermutationFactory用于根据用户的选择生成不同的方法实例。Util是工具类，里面静态实现了计算十进制数的阶乘，以及转化为递增进位制数和递减进位制数等操作。PermutationTest是测试类，用户首先输入选择的全排列生成算法的序号，接着输入生成排列的长度，控制台即会一次输出所有的全排列。各全排列算法思路如下：

1. 字典序法：

首先根据输入的排列长度，生成初始的数字排列， 接着计算排列的递增进位制表示的中介数(记录当前数字右边比当前数字小的数字个数)，之后分别计算出之后的中介数表示，并将其转化为对应的排列(从左往右依次填入，注意左边已经出现的比前数字小的数字)。

1. 递增进位制数法：

首先根据输入的排列长度，生成初始的数字排列， 接着计算排列的递增进位制表示的中介数(anan-1…a2)↑, *ai*：*i*的右边比*i*小的数字的个数)，之后分别计算出之后的中介数表示，并将其转化为对应的排列(从大到小求出n,n-1,…,2,1的位置,依次填入)。

1. 递减进位制数法：

首先根据输入的排列长度，生成初始的数字排列， 接着计算排列的递减进位制表示的中介数((a2a3…an-1an)↓，*ai*：*i*的右边比*i*小的数字的个数)，之后分别计算出之后的中介数表示，并将其转化为对应的排列(从大到小求出n,n-1,…,2,1的位置,依次填入)。

1. 邻位对换法法：

首先根据输入的排列长度，生成初始的数字排列， 接着计算排列的递减进位制表示的中介数以及各位数的方向(2的方向一定是向左, 对于每一个大于2的数字*i*，如果i为奇数，其方向性决定于bi-1的奇偶性，奇向右、偶向左。如果i为偶数，其方向性决定于bi-1+ bi-2的奇偶性，同样是奇向右、偶向左。当得到方向性后，bi的值就是背向i的方向直到排列边界这个区间里比i小的数字的个数), 之后分别计算出之后的中介数表示，并将其转化为对应的排列(从大到小求出n,n-1,…,2,1的位置以及方向,依次填入)。

下图各主要类的UML图：

