使用系统的API

```
public String reversel(String str){
    return new StringBuilder(str).reverse().toString();
}
```

不实用系统的API
反转字符串的规律：两个反转字符串的下标之和为字符串长度-1.

1.逆序循环

```
//思路：
//将字符串转换为char数组
//遍历循环给char数组赋值
public static String reversel(String str){
    char[] chars = str.toCharArray();
    StringBuilder stringBuilder = new StringBuilder();
    for (int i = chars.length - 1; i >= 0; i--) {
        stringBuilder.append(chars[i]);
    }
    return stringBuilder.toString();
}

//上面的方法还可以继续优化
//只需要遍历一半就可以对所有位置对值进行赋值
public static String reversel(String str) {
    char[] chars = str.toCharArray();
    int length = str.length();
    for (int i = (chars.length - 1) / 2; i >= 0; i--) {
        chars[i] = str.charAt(length - 1 - i);
        chars[length - 1 - i] = str.charAt(i);
    }
    return new String(chars);
}
```

2.使用数组

```
//将字符串转换为char数组
//遍历循环给char数组赋值
public static String reversel(String str) {
    char[] chars = str.toCharArray();
    int length = str.length();

    for (int i = 0; i < length; i++) {
        chars[i] = str.charAt(length - 1 - i);
    }
```

```
    return  new String(chars);
}

//和逆序循环一样，也可以优化
public static String reversel(String str) {
    char[] chars = str.toCharArray();
    int length = str.length();

    for (int i = 0; i < length / 2; i++) {
            chars[i] = str.charAt(length - 1 - i);
        chars[length - 1 - i] = str.charAt(i);
    }
    return new String(chars);
}
```

3.使用栈

栈的特点：后进先出（LIFO）

```
//根据栈后进先出的特性来反转字符串
//1.先将字符串转换为char数组。
//2.将char数组中的字符依次压入栈中。
//3.将栈中的字符依次弹出。
public static String reversel(String str){
    char[] chars = str.toCharArray();
    Stack<Character> stack = new Stack<>();
    for (char aChar : chars) {
        stack.push(aChar);
    }

    int length = str.length();
    for (int i = 0; i < length; i++) {
        chars[i] = stack.pop();
    }
    return new String(chars);
}
```

4.使用位运算

二进制数据的处理往往是通过位运算来实现的。为操作有：与，或，非，异或。
使用异或操作能实现交换两个变量的值而不引入第三个变量。
异或操作：当两两数值相同为否，数值不同为真。相同为0，不同为1。
两个数异或的结果再与其中一个数异或的结果是另外一个数。

```

```

5.递归

后面两种方式暂时还没完全理解。理解透彻之后再上传。