1.什么是整数溢出?

Under Java, the ranges of the integer types do not depend on the machine on which you will be running the Java code. This alleviates a major pain for the programmer who wants to move software from one platform to another, or even between operating systems on the same platform. In contrast, C and C++ programs use the most efficient integer type for each processor. As a result, a C program

that runs well on a 32-bit processor may exhibit integer overflow on a 16-bit system. Since Java programs must run with the same results on all machines, the ranges for the various types are fixed.

2.这一段感觉乱乱的，都要记住吗?

Long integer numbers have a suffix `L` or `l` (for example, `4000000000L`). Hexadecimal numbers have a prefix `0x` or `0X` (for example, `0xCAFE`). Octal numbers have a prefix `0` (for example, `010` is 8)—naturally, this can be confusing, so we recommend against the use of octal constants.

Starting with Java SE 7, you can write numbers in binary, with a prefix `0b` or `0B`. For example, `0b1001` is 9. Also starting with Java SE 7, you can add underscores to number literals, such as `1_000_000` (or `0b1111_0100_0010_0100_0000`) to denote one million. The underscores are for human eyes only. The Java compiler simply removes them.

3.看不懂这个家伙在讲什么。。

3.3.4 Unicode and the char Type

To fully understand the `char` type, you have to know about the Unicode encoding scheme. Unicode was invented to overcome the limitations of traditional character encoding schemes. Before Unicode, there were many different standards: ASCII in the United States, ISO 8859-1 for Western European languages, KOI-8 for Russian, GB18030 and BIG-5 for Chinese, and so on. This caused two problems. A particular code value corresponds to different letters in the different encoding schemes. Moreover, the encodings for languages with large character sets have variable length: Some common characters are encoded as single bytes, others require two or more bytes.

Unicode was designed to solve these problems. When the unification effort started in the 1980s, a fixed 2-byte code was more than sufficient to encode all characters used in all languages in the world, with room to spare for future expansion—or so everyone thought at the time. In 1991, Unicode 1.0 was released, using slightly less than half of the available 65,536 code values. Java was designed from the ground up to use 16-bit Unicode characters, which was a major advance over other programming languages that used 8-bit characters.

Unfortunately, over time, the inevitable happened. Unicode grew beyond 65,536 characters, primarily due to the addition of a very large set of ideographs used for Chinese, Japanese, and Korean. Now, the 16-bit `char` type is insufficient to describe all Unicode characters.

We need a bit of terminology to explain how this problem is resolved in Java, beginning with Java SE 5.0. A *code point* is a code value that is associated with a character in an encoding scheme. In the Unicode standard, code points are written in hexadecimal and prefixed with `U+`, such as `U+0041` for the code point of the Latin letter A. Unicode has code points that are grouped into 17 *code planes*. The first code plane, called the *basic multilingual plane*, consists of the "classic" Unicode characters with code points `U+0000` to `U+FFFF`. Sixteen additional planes, with code points `U+10000` to `U+10FFFF`, hold the *supplementary characters*.

The UTF-16 encoding represents all Unicode code points in a variable-length code. The characters in the basic multilingual plane are represented as 16-bit values, called *code units*. The supplementary characters are encoded as consecutive pairs of code units. Each of the values in such an encoding pair falls into a range of 2048 unused values of the basic multilingual plane, called the *surrogates area* (`U+D800` to `U+DBFF` for the first code unit, `U+DC00` to `U+DFFF` for the second code unit). This is rather clever, because you can immediately tell whether a code unit encodes a single character or it is the first or second part of a supplementary character. For example, 𝕆 (the mathematical symbol for the set of octonions, `http://math.ucr.edu/home/baez/octonions`) has code point `U+1D546` and is encoded by the two code units `U+D835` and `U+DD46`. (See `http://en.wikipedia.org/wiki/UTF-16` for a description of the encoding algorithm.)

In Java, the `char` type describes a *code unit* in the UTF-16 encoding.

Our strong recommendation is not to use the `char` type in your programs unless you are actually manipulating UTF-16 code units. You are almost always better off treating strings (which we will discuss in Section 3.6, "Strings," on p. 65) as abstract data types.

4. System.out.println(xx);和public static void main(String[] args)是什么意思啊，我看书上好多代码都是以public开头，这有什么特殊含义吗

我只看完了3.1-3.4。。而且感觉只记下了一些字符表示的含义，在讲Unicode的定义之类的时候完全是懵逼的T-T