Java不仅仅是一门编程语言，它还是一个由一系列计算机软件和规范组成的技术体系，

在虚拟机层面隐藏了底层技

术的复杂性以及机器与操作系统的差异性。运行程序的物理机千差万别，而Java虚拟机则在千差万别

的物理机上面建立了统一的运行平台，实现了在任意一台Java虚拟机上编译的程序，都能在任何其他

Java虚拟机上正常运行。这一极大的优势使得Java应用的开发比传统C/C++应用的开发更高效快捷，程

序员可以把主要精力放在具体业务逻辑，而不是放在保障物理硬件的兼容性上。

为了达到“所有硬件提供一致的虚拟平台”的目的，牺牲了一些硬件相关的 性能特性。

Java EE 以javax.\*作为包名，而以java.\*为包名的包都是Java SE API的核心包

1995年5月23日，Oak语言改名为Java，并且在SunWorld大会上正式发布Java 1.0版本。Java语言第

一次提出了“Write Once，Run Anywhere”的口号。

JIT即时编译器

2002年前后还发生了一件与Java没有直接关系，但事实上对Java的发展进程影响很大的事件，就是

微软的.NET Framework发布。这个无论是技术实现还是目标用户上都与Java有很多相近之处的技术平

台给Java带来了很多讨论、比较与竞争，.NET平台和Java平台之间声势浩大的孰优孰劣的论战到今天

为止都仍然没有完全平息。

提前编译（Ahead of Time Compilation，AOT）

.net CLR 和JIT理解<https://www.cnblogs.com/tk091/archive/2012/09/01/2666810.html>

更重要的是指近几年在从大型单

体应用架构向小型微服务应用架构发展的技术潮流下，Java表现出来的不适应。

提前编译是相对于即时编译的概念，提前编译能带来的最大好处是Java虚拟机加载这些已经预编

译成二进制库之后就能够直接调用，而无须再等待即时编译器在运行时将其编译成二进制机器码。理

论上，提前编译可以减少即时编译带来的预热时间，减少Java应用长期给人带来的“第一次运行慢”的

不良体验，可以放心地进行很多全程序的分析行为，可以使用时间压力更大的优化措施

但是提前编译的坏处也很明显，它破坏了Java“一次编写，到处运行”的承诺，必须为每个不同的

硬件、操作系统去编译对应的发行包；也显著降低了Java链接过程的动态性，必须要求加载的代码在

编译期就是全部已知的，而不能在运行期才确定，否则就只能舍弃掉已经提前编译好的版本，退回到

原来的即时编译执行状态。

Configuration summary:

\* Debug level: release

\* HS debug level: product

\* JVM variants: server

\* JVM features: server: 'aot cds cmsgc compiler1 compiler2 epsilongc g1gc graal jfr jni-check jvmci jvmti management nmt parallelgc serialgc services shenandoahgc vm-structs zgc'

\* OpenJDK target: OS: linux, CPU architecture: x86, address length: 64

\* Version string: 12-internal+0-adhoc.miku.jdk12-06222165c35f (12-internal)

Tools summary:

\* Boot JDK: openjdk version "11.0.10" 2021-01-19 OpenJDK Runtime Environment (build 11.0.10+9-Ubuntu-0ubuntu1.20.04) OpenJDK 64-Bit Server VM (build 11.0.10+9-Ubuntu-0ubuntu1.20.04, mixed mode, sharing) (at /usr/lib/jvm/java-11-openjdk-amd64)

\* Toolchain: gcc (GNU Compiler Collection)

\* C Compiler: Version 9.3.0 (at /usr/bin/gcc)

\* C++ Compiler: Version 9.3.0 (at /usr/bin/g++)

Build performance summary:

\* Cores to use: 3

\* Memory limit: 3907 MB

程序1 在idea中研究调试Java虚拟机参数。实现了Java 堆内存溢出异常测试

import java.util.ArrayList;

import java.util.List;

public class HeapOOM {

static class OOMObject {

}

public static void main(String[] args) {

List<OOMObject> list = new ArrayList<OOMObject>();

while (true) {

list.add(new OOMObject());

}

}

}