1绪论

1.2计算机模拟生命起源的介绍

1.2.1C语言做模型的介绍

1.2.2Java语言做模型的介绍

基本类：核苷酸碱基类（NucleotideBase，NB）：包含属性ACGU0，其中有一个方法根据NB获取值。

4 讨论

4.1 java模型比C语言模型运行效率低的原因

Java语言和C语言作为高级语言，均需要进行编译才能运行，这个过程需要使用编译器，编译器的作用就是将源代码转换成计算机能够识别的二进制形式。

Java语言和C语言的对比实际上是JIT编译器与静态编译器的对比。

* **JIT编译器**，英文写作Just-In-Time Compiler，中文意思是即时编译器。JIT编译器能够将MSIL编译成为各种不同的机器代码，以适应对应的系统平台，最终使得程序在目标系统中得到顺利地运行。Java语言使用JIT编译器
* **静态编译**，就是编译器在编译可执行文件的时候，将可执行文件 需要调用的对应动态链接库(.so或.lib)中的部分提取出来，链接到可执行文件中去，使可执行文件在运行的时候不依赖于动态链接库。C语言使用静态编译

无论是C语言还是Java语言，最终编译之后被机器执行的都是本地机器码，那种语言性能更高，除了它们自身API库实现的好坏之外，其余的比较就成了编译器对比。

这两种语言各有何优势。

1. JIT编译器运行占用的是用户程序运行时间，而静态编译的时间成本在源码编译过程，而不是在用户运行过程。
2. Java语言是动态的类型安全语言，这意味着需要由JVM来确保程序不会违反语言语义或访问非结构化内存。在实现层面上看，这就意味着JVM必须频繁进行动态检查，如对象实例访问时检查空指针、数组元素访问时检查上下界范围等等。对于这类程序代码没有明确写出的检查行为，尽管编译器会努力进行优化，但是总体上仍然要消耗着不少的运行时间。
3. Java语言是可以动态扩展的语言，运行时加载新的类可能改变程序类型继承关系，使得很多全局的优化都难以进行，因为编译器无法看到程序的全貌，编译器不得不时刻注意并随着类型变化而在运行过程中进行一些优化。
4. Java语言中的对象内存分配都是在堆上进行，只有方法中的局部变量才在栈上分配。而C语言的对象则有多种内存分配方式，即可以在堆上分配，也可以在栈上分配。C语言中主要由用户程序代码来回收分配的内存，这就不存在无用对象筛选的过程，因此在效率（指运行效率，排除了开发效率）上也比垃圾收集机制要高。
5. C语言是编译型的，直接将源码编译成机器代码；Java是解释型的，源码被编译成二进制伪代码，由Java虚拟机（JVM）解释执行。但是，由于C是编译型的，它的可移植性差；而Java是解释执行，因此具有良好的移植性，可跨平台运行。

4.2 java模型比C语言模型开发效率高的佐证？原因？

二者的内存管理机制不同。C需要完全控制内存管理，Java主要依靠JVM的GC功能。

C语言在开发过程中需要开发人员对内存进行编程控制管理。C语言在语法限制以及变量的类型上比较宽松，这样就会影响到程序的安全性。

Java可以对内存中产生的垃圾进行自动收集，大幅度降低了程序的复杂程度。在开发过程中主要依靠JVM的GC功能进行内存管理，降低了开发人员手动编程控制管理。

C语言是面向过程的开发语言，而Java是面向对象的开发语言

面向过程就是分析出解决问题所需要的步骤，然后用函数把这些步骤一步一步实现，使用的时候一个一个依次调用执行。

面向过程编程的优缺点：

* 开发过程中对类调用需要实例化，开销比较大，比较消耗资源。
* 面向过程是符合机器的思想
* 不易维护、不易复用、不易扩展。
* 是具体化的，流程化的，解决一个问题，需要一步一步的分析，一步一步的实现。

面向对象（Object Oriented Programming ，简称OOP）是一种编程范式，它使用“对象”来设计软件和数据结构。在面向对象编程中，对象是包含数据和能够对数据进行操作的方法的实体。有时对象也称作类。这种方法强调了数据的封装、继承和多态性，使得软件设计更加模块化，易于理解和维护。将现实的事物抽象出来，注意抽象这个词是重点啊，把现实生活的事物以及关系，抽象成类，通过继承，实现，组合的方式把万事万物都给容纳了。实现了对现实世界的抽象和数学建模。这是一次飞跃性的进步。

面向对象编程的核心概念包括：

* 封装：隐藏对象的内部状态和实现细节，仅对外提供公共的访问方式，将变化隔离，便于使用，提高复用性和安全性。
* 继承：允许新创建的类（子类）继承现有类（父类）的属性和方法。提高代码复用性
* 多态：允许不同类的对象对同一消息做出响应，即同一操作可以作用于不同的对象上，产生不同的结果。

面向对象编程的优点是：

* 提高了代码的复用性、可维护性和可扩展性。特别是在需要大量代码重用和模块化设计的情况。
* 可以设计出低耦合的系统，使系统更加灵活、更加易于维护
* 抽象会使复杂的问题更加简单化，它支持更高级的抽象和更容易的代码管理。
* 面向对象更符合人类的思维。
* 是模型化的，只需要抽象出一个类，像一个封闭的盒子，在这里你拥有数据也拥有解决问题的方法。需要什么功能直接使用旧可以了，不必去一步一步的实现，至于这个功能是如何实现的，不需要去管，会用就可以了。

你可以理解为面向对象编程最终的目的是对要解决的问题进行问题的抽象和数学建模。利用模型去解决问题。

或者可以理解为面向对象的底层其实就是面向对象，把面向对象抽象成类，然后封装，方便我们使用，这就是面向对象了。

多态和继承是相辅相成的。举例说明：

打印机，包含黑白打印机、彩色打印机。

此时，打印机它属于一个父类，黑白打印机属于子类，彩色打印机也属于子类

打印机的关系

1. 继承：黑白打印机和彩色打印机都继承自打印机这个父类。例如打印机父类有一个自动翻页打印功能，那么黑白打印机和彩色打印机就自动拥有了自动翻页打印功能。
2. 重写：使用打印机进行打印（此时打印机有一个打印方法），如果打印机为彩色打印机，则打印出来的就是彩色的（此时彩色打印机需要重写打印机的打印方法，让其打印效果为彩色）；如打印机为黑白打印机，则打印出来的就是黑白的（此时黑白打印机需要重写打印机的打印方法，让其打印效果为黑白的）。
3. 引用：new一个打印机对象时，父类打印机的引用可以指向具体的子类对象（彩色打印机或黑白打印机）。Print parent = new ColorPrint();父类打印机的引用指向子类彩色打印机。

资料：

* <https://www.educative.io/blog/compiled-vs-interpreted-language>
* <https://www.baeldung.com/java-compiled-interpreted>
* <https://www.baeldung.com/cs/compiled-vs-interpreted-languages>
* <https://www.baeldung.com/cs/how-compilers-work>
* https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/