A

Seminar report

On

Java

Submitted in partial fulfillment of the requirement for the award of degree

Of CSE

SUBMITTED TO:

www.studymafia.org

SUBMITTED BY:

www.studymafia.org

Acknowledgement

I would like to thank respected Mr…….. and Mr. ……..for giving me such a wonderful opportunity to expand my knowledge for my own branch and giving me guidelines to present a seminar report. It helped me a lot to realize of what we study for.

Secondly, I would like to thank my parents who patiently helped me as i went through my work and helped to modify and eliminate some of the irrelevant or un-necessary stuf

Thirdly, I would like to thank my friends who helped me to make my work more organized and well-stacked till the end.

Next, I would thank Microsoft for developing such a wonderful tool like MS Word. It helped my work a lot to remain error-free.

Last but clearly not the least, I would thank The Almighty for giving me strength to complete my report on time.

Preface

I have made this report file on the topic **Java**; I have tried my best to elucidate all the relevant detail to the topic to be included in the report. While in the beginning I have tried to give a general view about this topic.

My efforts and wholehearted co-corporation of each and everyone has ended on a successful note. I express my sincere gratitude to ……..who assisting me throughout the preparation of this topic. I thank him for providing me the reinforcement, confidence and most importantly the track for the topic whenever I needed it.

CONTENT

INTRODUCTION OF JAVA

Java is a programming language and environment invented by James Gosling and others in 1994. Java was originaly named Oak and was developed as a part of the Green project at the Sun Company. Java is an object-oriented programming language with a built-in application programming interface (API) that can handle graphics and user interfaces and that can be used to create applications or applets. Because of its rich set of API's, similar to Macintosh and Windows, and its platform independence, Java can also be thought of as a platform in itself. Java also has standard libraries for doing mathematics.

Java is the same as C and C++. One major difference is that Java does not have pointers. However, the biggest difference is that you must write object oriented code in Java. Procedural pieces of code can only be embedded in objects. In the following we assume that the reader has some familiarity with a programming language.

In particular, some familiarity with the syntax of C/C++ is useful.In Java we distinguish between applications, which are programs that perform the same functions as those written in other programming languages, and applets, which are programs that can be embedded in a Web page and accessed over the Internet. Our initial focus will be on writing applications. When a program is compiled, a byte code is produced that can be read and executed by any platform that can run Java.

KEY FEATURES OF JAVA

1. Plateform Independence

   The concept of Write-once-run-anywhere (known as the Platform independent) is one of the important key feature of java language that makes java as the most powerful language. Not even a single language is idle to this feature but java is more closer to this feature. The programs written on one platform can run on any platform provided the platform must have the JVM.

2. Simple

   There are various features that makes the java as a simple language. Programs are easy to write and debug because java does not use the pointers explicitly. It is much harder to write the java programs that can crash the system but we can not say about the other programming languages. Java provides the bug free system due to the strong memory management. It also has the automatic memory allocation and deallocation system.

3. Object Oriented

   To be an Object Oriented language, any language must follow at least the four characteristics.

   - Inheritance: It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse the existing code and adding the additional features as needed.
   - Encapsulation: It is the mechanism of combining the information and providing the abstraction.
   - Polymorphism : As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the methods.

- Dynamic binding : Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

4. Robust

   Java has the strong memory allocation and automatic garbage collection mechanism. It provides the powerful exception handling and type checking mechanism as compare to other programming languages. Compiler checks the program whether there any error and interpreter checks any run time error and makes the system secure from crash. All of the above features makes the java language robust.

5. Distributed

   The widely used protocols like HTTP and FTP are developed in java. Internet programmers can call functions on these protocols and can get access the files from any remote machine on the internet rather than writing codes on their local system.

6. Portable

   The feature Write-once-run-anywhere makes the java language portable provided that the system must have interpreter for the JVM. Java also have the standard data size irrespective of operating system or the processor. These features makes the java as a portable language.

7. Performance

   Java uses native code usage, and lightweight process called threads. In the beginning interpretation of bytecode resulted the performance slow but the advance version of JVM uses the adaptive and just in time compilation technique that improves the performance.

8. Multithreaded

   Java is a Multithreaded programming language. Multithreading means a single program having different threads executing independently at the same time. Multiple

threads execute instructions according to the program code in a process or a program. Multithreading works the similar way as multiple processes run on one computer. Multithreading programming is a very interesting concept in Java. In multithreaded programs not even a single thread disturbs the execution of other thread. Threads are obtained from the pool of available ready to run threads and they run on the system CPUs.

9. Interpreted & Compiler

   We all know that Java is an interpreted language as well. With an interpreted language such as Java, programs run directly from the source code. The interpreter program reads the source code and translates it on the fly into computations. Thus, Java as an interpreted language depends on an interpreter program.The versatility of being platform independent makes Java to outshine from other languages. The source code to be written and distributed is platform independent. Another advantage of Java as an interpreted language is its error debugging quality. Due to this any error occurring in the program gets traced. This is how it is different to work with Java.

10. Architecture Neutral

    The term architectural neutral seems to be weird, but yes Java is an architectural neutral language as well. The growing popularity of networks makes developers think distributed. In the world of network it is essential that the applications must be able to migrate easily to different computer systems. Not only to computer systems but to a wide variety of hardware architecture and Operating system architectures as well. The Java compiler does this by generating byte code instructions, to be easily interpreted on any machine and to be easily translated into native machine code on the fly.

    PACKAGE

    A *package* is a grouping of related types providing access protection and name space management. Note that *types* refers to classes, interfaces, enumerations, and

annotation types. Enumerations and annotation types are special kinds of classes and interfaces, respectively, so *types* are often referred to in this lesson simply as *classes and interfaces*. The types that are part of the Java platform are members of various packages that bundle classes by function: fundamental classes are in java.lang, classes for reading and writing (input and output) are in java.io, and so on.If no package is specified, the classes in the file go into a special unnamed package (the same unnamed package for all files). If package *name* is specified, the file must be in a subdirectory called *name* (i.e., the directory name must match the package name). The Java Package name consists of words separated by periods.

The first part of the name represents the organization which created the package. The remaining words of the Java Package name reflect the contents of the package. The Java Package name also reflects its directory structure. important reason for using packages is that it provides programmers with greater control over their source code. It is typical to have a few thousand source files in medium to large scale applications, and trying to maintain them would be difficult at best, if not impossible. However, separating these source files into packages makes it much easier to manage the source code. What usually occurs is that related classes are grouped into a single package, for example, all the user interface classes of an application will be grouped into a package.

FEATURES OF PACKAGES

1. Packages are useful for the following purposes: Packages allow you to organize your classes into smaller units(such as folders) and make it easy to locate and use the appropriate class file.
2. It helps to avoid naming conflicts. When you are working with a number of classes, it becomes difficult to decide on names of the classes & methods.
3. At times you would want to use the same name, which belongs to an another class. Package, basically hides the classes and avoids conflicts in names.
4. Packages allow you to protect your classes, data and methods in a larger way than on a class-to-class basis.
5. Package names can be used to identify your classes.
6. Reusability of code is one of the most important requirements in the software industry.
7. Reusability saves time, effort and also ensures consistency.
8. In Java, the codes which can be reused by other programs is put into a “Package”.
9. A Package is a collection of classes, interfaces and/or other packages.
10. Packages are essentially a means of organizing classes together as groups.

STANDARD JAVA PACKAGES

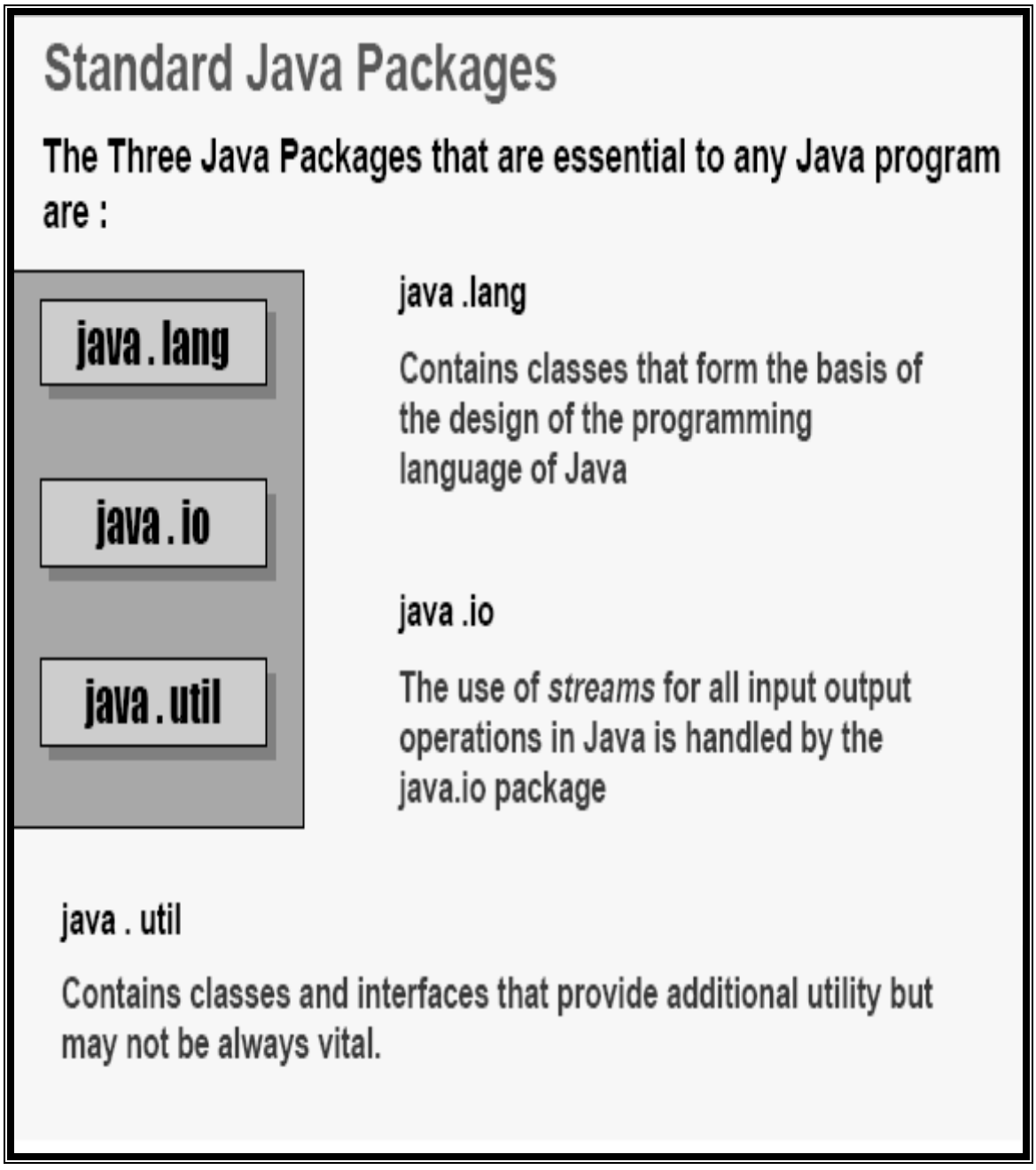

Figure 1

JAVA.LANG PACKAGE

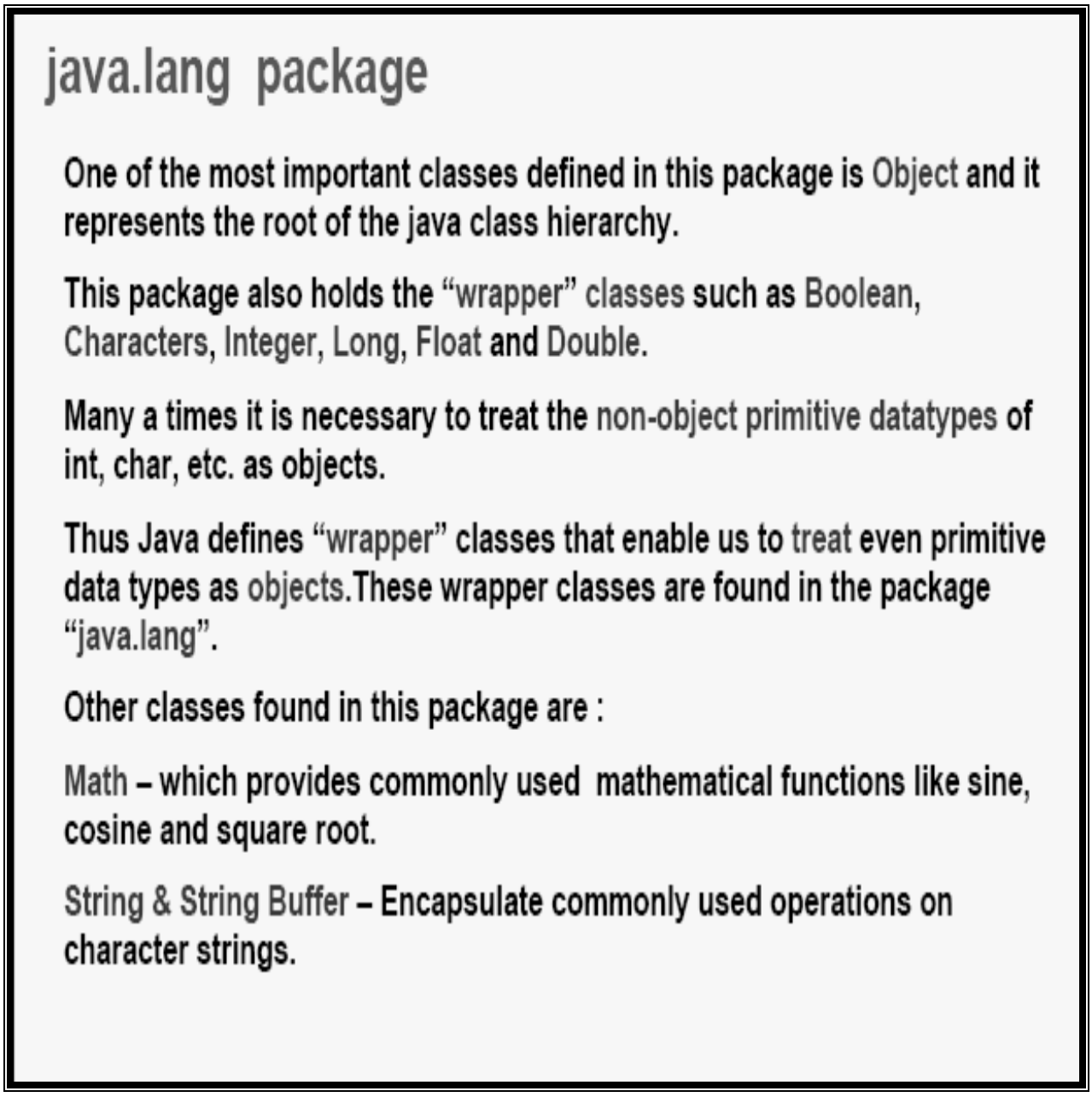

Figure 2

JAVA.IO PACKAGE

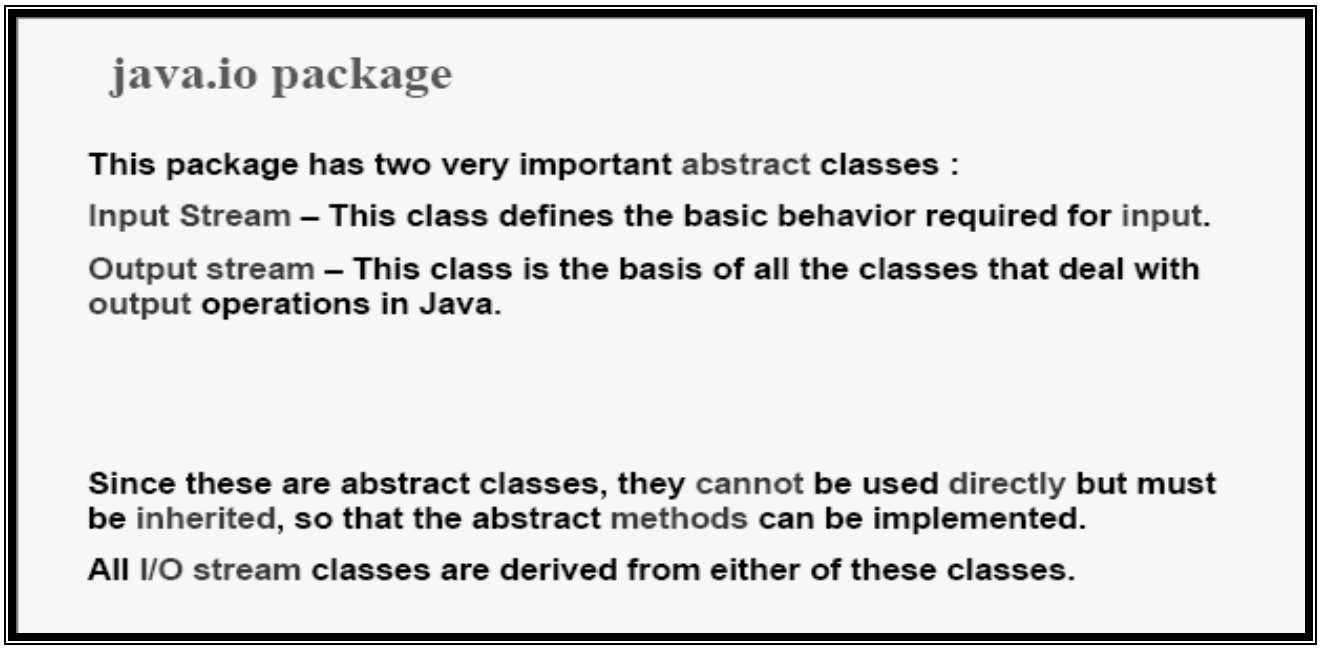

Figure 3

- The classes derived in Inputstream and Outputstream can only read from or write to the respective files.
- We cannot use the same class for both reading and writing operations.
- An exception to this rule is the class RandomAccessFile".
- This is the class used to handle files that allow random access and is capable of mixed reading and writing operations on a file.
- There are two additional interface to this package :
    - Data input

- Data output

- These classes are used to transfer data other than bytes or characters.

JAVA.UTIL PACKAGE

- One of the most important package in this package is the class “Date”, which can be used to represent or manipulate date and time information.
- In addition, the class also enable us to account for time zones .
- Java helps us to change the size of an array which is usually fixed, by making use of the class “Vector”. This class also enable us to add, remove and search for items in the array.

IMPORTANT PACKAGES

- java.applet
  - This package consists of classes that you need, to execute an applet in the browser or an appletviewer.
- java.awt
  - This package is useful to create GUI applications.
- java.net
  - This package provides classes and interfaces for TCP/IP network programming.

CREATING A PACKAGE

Java supports a keyword called “package” for creating user-defined packages. The package statement must be the first statement in a Java source file (except comments and white spaces) followed by one or more classes.

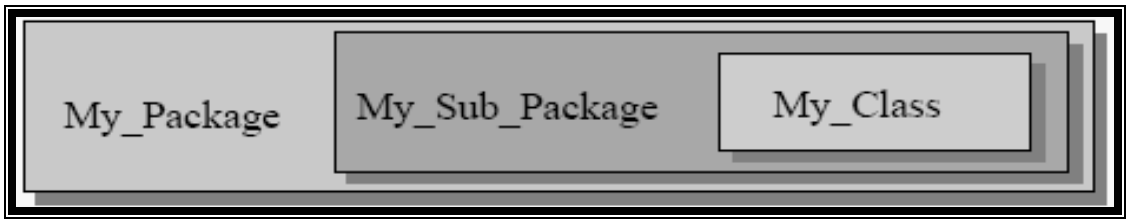

Figure 4

```
import My_Package . MySub_Package . My_Class ;
```

Package name is “myPackage” and classes are considred as part of this package; The code is saved in a file called “calculater.java” and located in a directory called “myPackage”.

Advantages of Java

1.Java is easy to learn.

Java was designed to be easy to use and is therefore easy to write, compile, debug, and learn than other programming languages.

2.Java is object-oriented.

This allows you to create modular programs and reusable code.

3.Java is platform-independent.

One of the most significant advantages of Java is its ability to move easily from one computer system to another. The ability to run the same program on many different systems is crucial to World Wide Web software, and Java succeeds at this by being platform-independent at both the source and binary levels.

4.Java is distributed.

Java is designed to make distributed computing easy with the networking capability that is inherently integrated into it. Writing network programs in Java is like sending and receiving data to and from a file.

5.Java is secure.

Java considers security as part of its design. The Java language, compiler, interpreter, and runtime environment were each developed with security in mind.

6.Java is robust.

Robust means reliability. Java puts a lot of emphasis on early checking for possible errors, as Java compilers are able to detect many problems that would first show up during execution time in other languages.

7.Java is multithreaded

Multithreaded is the capability for a program to perform several tasks simultaneously within a program. In Java, multithreaded programming has been smoothly integrated into it, while in other languages, operating system-specific procedures have to be called in order to enable multithreading.

Disadvantages of Java

- Slow
- Environment limitations
- Applet limitations imposed due to security model
- Uncertain status of security
- Programming language limitations
- Platform limitations
- General stability concerns

Conclusions

Java offers the real possibility that most programs can be written in a type-safe language. However, for Java to be broadly useful, it needs to have more expressive power than it does at present.

This paper addresses one of the areas where more power is needed. It extends Java with a mechanism for parametric polymorphism, which allows the definition and implementation of generic abstractions. The paper gives a complete design for the extended language. The proposed extension is small and conservative and the paper discusses the rationale for many of our decisions. The extension does have some impact on other parts of Java, especially Java arrays, and the Java class library.

The paper also explains how to implement the extensions. We first sketched two designs that do not change the JVM, but sacrifice some space or time performance. Our implementation avoids these performance problems. We had three main goals: to allow all instantiations to share the same bytecodes (avoiding code blowup), to have good performance when using parameterized code, and to have little impact on the performance of code that does not use parameterization.

The implementation discussed in Section 3 meets these goals. In that section, we described some small extensions to the virtual machine specification that are needed to support parameterized abstractions; we also described the designs of the bytecode verifier and interpreter, and the runtime structures they rely on.

Preliminary performance results from our implementation of the extended bytecode interpreter show roughly a 2% penalty for the presence of parameterized code, but a speedup for parameterized code of 17%, by eliminating runtime checks. We expect that some simple performance tuning can improve these results.

References

- www.google.com
- www.wikipedia.com
- www.studymafia.org