Java Essay, Research Paper

Final Term Paper

The Java Revolution

Java s new-age design is leading the object-oriented programming language design revolution; allowing even greater portability, state of the art security measures, and more extensive reusability. First, a brief description of what makes a programming language object-oriented so that its importance can be realized. Object oriented programming (OOP) emphasizes data, its storage, and its necessity, instead of algorithm usage to solve problems. Object oriented programming involves two closely related, yet separate parts: classes and objects. A class is a specification that describes a new object type, acting as templates defining what an object can store and how it will behave. An object is an instance of the class created by the parameters defined in the class specification. The object oriented programming approach to program design is to first design classes that will accurately represent those things with which the program deals, then use instances of those classes, objects, to complete the task at hand. An example may be designing a class Account, which contains an ID, password and account balance. The object would be a specific instance, for example, account 451. OOP has been fundamentally praised for its ability to create easily reusable code, whose functionality can grow through the inheritance, and derivation of new classes from old ones. In addition, polymorphism allows the programmer to create multiple definitions for operators and functions. For instance, polymorphism would allow a programmer to define the addition of two Account objects to represent the sum of their account balances. Object orientated programming s main innovated improvements in programming design are greatest accented by the Java programming language.

Java is a new programming language developed by Sun Microsystems that is well liked, and largely accepted world round. Although Java is widely recognized, few are fully aware of its potential, and even less are aware of its purpose. It certainly has the prospect of becoming a part of our everyday lives, existing in our mobile phones, televisions, Internet browsers, and all aspects of our future electronic communication devices. Still in its infancy, Java has yet to fulfill its intended purpose of being an embedded language for consumer electronics. Its reusability and portability has proved it to be an ideal interface for the Internet. In 1996, Netscape added Java support to its evolving web browser, Netscape Navigator. This allowed applets to be included in web page downloads, which lead to advancements from the world of static text. This directs us to the first topic proving Java to be the wave of the future.

Java s use of applets allows an independence to occur between the running of the Java application, and the actual machine that is running it. The applet is run in a Java-compatible browser, which creates a virtual machine interface to the local machine. An applet is similar to an application, but it will not run in a stand-alone environment. An applet must adhere to a certain set of conventions that enables it to run within a Java-compatible browser. The technology is still evolving and most Java applications are still relatively simple animations, or user interactions. The future of Java lies closely correlated to its original design purpose. It will lead the revolution of most electronics having embedded software that will all interact with each other. The portability of Java fills the gap that a lack of standard operating protocols has created. One of the biggest benefits of Java is the fact that it is machine independent. Java is compiled into a machine independent byte code. The byte code is then interpreted by what is referred to as a virtual machine running on the host computer, hence implementing a run-time environment. If Java were just another programming language, the industry would have greeted its introduction with a minimal amount of interest. What makes Java intriguing is that it is also a runtime environment embodied in a virtual machine. It sits, in essence, between the Java program and the machine it is running on, offering the program an abstract computer that executes the Java code and guarantees certain behaviors regardless of the underlying hardware or software platform. Java compilers thus turn Java programs not into a language for a particular machine but into a platform-neutral byte code that the machine-specific virtual machine interprets on the fly. A Java compiler can be best described as a translator that interprets a language to a common ground in a similar fashion to a United Nations translator translating all languages to English. Since English is known around the world, specific translators, like the virtual machine, are able to translate English into the specific language for their country. With so many different computers connected to the Internet, platform independence is necessary. If the java virtual machine can create the necessary interface between the operating systems, whether it is a coffee maker or a mainframe computer, all these electronic devices can easily communicate with one another, which can make collaboration available. Imagine if all that one had to do was to provide a meetings date, time, and location into Outlook Calendar (or a similar program), and could be confident that the following certain actions would occur. Automatically the alarm clock is set, directions to the meeting are located and printed out, coffee maker is turned on 10 minutes after the alarm clock, car is pre-heated 2 minutes prior to departure time, etc. Every electronic device can have a common gateway with the extension of Java s platform (or pre-defined classes). This universal portal is evident in Java s intended purpose by its design of having all classes inherit from a common parent, the Object class. This keeps every possible class creation connected and well related at the highest level. For now, the applet experience is not quite as exciting. Most contemporary users are unlikely to be impressed with scrolling text, or simple animations, especially if the price for these unnecessary displays warrants an increased download time. A Java applet is delivered along with the web page. The browser then downloads the applet and runs it on your machine. Enhancing the portability, the concept of being able to run applications on your system is significant for several reasons. If a user wants to see hear a music file, but doesn’t have an audio player, or file converter, they can be easily acquired all at once. This eliminates the problem of not having the corresponding assistant application or even worse, having to settle for faulty or sub-standard software. This ensures that applications are used in the way they were intended to. The ability to run applications on a local system has another significant advantage. Traditionally when something active is run on a web page, or interaction is ascertained, the processing is done on the remote computer. Both the input and output of the application must be transferred over constrained bandwidth, only to run on machines who s processing power is distributed over an unlimited number of requests. Also extra tracking and computation will be necessary if the remote server must make note of all currently active sessions. With applets all the necessary information is downloaded initially so the server can deal with and forget about the client in one atomic action. This has the affect of freeing Internet resources, allowing the work to take place on the client’s system instead of the server’s. If all applications were run on remote computers, the servers would be inundated with traffic, and these collectively could propagate to bring the entire Internet to a grinding halt. With a growing Internet population, the prospect of this becomes more of a reality. A solution does exist though. If it can be done safely, isn’t it quicker, not to mention more considerate, to get what you need, take it with you, and use it on your time? A good analogy is a long bank queue, with everyone waiting for the person in front of them to finish so they can get their business done. Some visitors will be quick, and while others may drag on depending on the scope of their duties. Imagine a different world where tellers, and not just their service, were issued at the bank. Instead of waiting for a teller to become free, you could simply request a copy of the teller you require, then complete your transaction on your own time. This independence would allow each bank client to get the desired services in a manner not heavily reliant on other clients demands . So if the person in front of you wants a thousand dollars in pennies, you don t have to wait for the teller to count them all. You would only have to wait in line to get a teller. If, like a web page, the bank got hundreds of new customers every day, can their be a different successful approach? The ability of the applet to be supported on a multitude of different machines shows the portability of a Java applet, but it also brings up another concern.

The issue of security still has to be carefully addressed. Away from the work environment, how could you trust the teller not to steal from your account, or even from your house. The bank manager wouldn’t be around to oversee her, unless you had a copy of him too. But then who would watch that he doesn t steal from the cookie jar, and on and on. A more appropriate solution would be to restrict the teller to a certain area, say a sandbox . A sandbox is the name given to the concept of setting the boundaries in which a Java applet can play , or do its duty/function. A Java applet cannot look at arbitrary files on the machine it’s running on, or have unauthorized access to system resources. It can’t introduce a virus or other malicious logic, delete critical files, or gain access to your passwords unless you allow it to. It is security measures like the sandbox that make Java language of the future. Java has been thrust into the spotlight with its new language built on the core values of security and platform independence. Sun Microsystems has promised a “Write once, run anywhere” language suited to an Internet community comprised of a different operating systems of different computer, hardware and software, configurations. The power to write programs that run on most everyone’s computer is revolutionary. For Internet applications, though, the value of platform independence degrades exponentially without strict, built-in security. Java provides this security, and has the power to change the way we compute.

The third aspect of Java that will aid in its evolution, is the large extent and dependency of its code s reusability. With the rise of the Internet, sharing information and code has become even easier than ever. This will only promote more classes to be re-used and customized than before, because of the apparent ease to find and access the desired information. As of now, an openly distributed, well-documented description of Java s platform is available on the web. As described before Sun has produced a tree of classes that continue to grow in the developers community as needs for customization arises. As long as these new classes are easily accessible, other people can reuse them. Post boards already exist, and multiple Internet pages donate their space to building these developer communities. It is a means of sharing code for the common programming good, which has proven to be successful with the rise of Lynx. Hence, the language builds upon itself at an exponential rate. The huge appeal of free software is another reason why I believe Java will develop at a quicker pace then its counterparts.

When the future of technology is pondered, only hypotheses can be made. Who would have imagined that the transfer of video and voice (multimedia) would first become possible through computers, and not through a combination of TV and phone products. Apple had plenty of resources and investments strictly attempting to figure out what the trend of the future would be so that their business strategy could be aligned in that direction. They decided that hardware was where the money was, hence where patents were necessary. Bill Gates had another idea and founded the most successful company on an opposite premise, the value of software. So, without the vast resources of a large company, I have speculated that because of Java s enhanced portability, security assurance, and reusability it will be the language to pave the way for object orientated design to revolutionize the computer industry.