**Eclipse:**

**What is Eclipse?**

In the context of computing, Eclipse is an integrated development environment (IDE) for developing applications using the Java programming language and other programming languages such as C/C++, Python, PERL, Ruby etc.

The Eclipse platform which provides the foundation for the Eclipse IDE is composed of plugins and is designed to be extensible using additional plug-ins. Developed using Java, the Eclipse platform can be used to develop rich client applications, integrated development environments, and other tools. Eclipse can be used as an IDE for any programming language for which a plug-in is available.

The Java Development Tools (JDT) project provides a plug-in that allows Eclipse to be used as a Java IDE, PyDev is a plugin that allows Eclipse to be used as a Python IDE, C/C++ Development Tools (CDT) is a plug-in that allows Eclipse to be used for developing application using C/C++, the Eclipse Scala plug-in allows Eclipse to be used an IDE to develop Scala applications and PHPeclipse is a plug-in to eclipse that provides complete development tool for PHP.

**Eclipse Origins**

Eclipse created by OT Eclipse created by OTI and IBM teams responsible I and IBM teams responsible for IDE products for IDE products – IBM VisualAgeVisualAge/Smalltalk (Smalltalk IDE) lltalk (Smalltalk IDE) – IBM VisualAgeVisualAge/Java (Jav /Java (Java IDE) – IBM VisualAgeVisualAge/Micro Edition.

1. Initially staffed with 40 full initially staffed with 40 full-time developers time developers.
2. Geographically dispersed development teams Geographically dispersed development teams – OTI Ottawa, OTI Minneapolis, OTI Zurich, IBM OTI Ottawa, OTI Minneapolis, OTI Zurich, IBM Toronto, OTI Ralei Toronto, OTI Raleigh, IBM RTP, IBM St. h, IBM RTP, IBM.
3. Effort transitioned into open source project Effort transitioned into open source project – IBM don IBM donated initial Eclipse code base ted initial Eclipse code base.
4. Platform, JDT, PDEEclipse is a universal platform for integrating development tools for integrating development tools.
5. Open, extensible architecture based on plug extensible architecture based on plug-ins.

Eclipse has emerged as a mainstream IDE for the Java programming language. The Eclipse Java debugger allows the developer to perform program animation (also referred to as stepping) by setting several types of breakpoints. Breakpoints enable pausing the execution when a specific location in the code is reached, when a specific condition is met, or when a specific variable is being accessed. In some applications, it is not feasible for the debugger to interrupt the program’s execution long enough for the developer to learn anything helpful about its behavior. If the program’s correctness depends on its real-time behavior, delays introduced by the debugger might cause the program to change its behavior, or perhaps fail, even when the code itself is correct. Examples of that arise when debugging application interfaces that involve user interaction. When a breakpoint defined in a method or on a variable that is involved in painting is hit, the program stops to respond to user interaction. Moreover, once the execution is resumed, the GUI needs to be repainted, which causes the breakpoint to be hit again. If the first hit does not reveal the bug, it would be difficult for the developer to reach the state that would reveal it.

The capabilities of each packaging of eclipse are different. Java developers typically use Eclipse Classic or Eclipse IDE for developing Java applications. The drop down box in the right corner of the download page allows you to set the operating system on which eclipse is to be installed. You can choose between Windows, Linux and Mac. Eclipse is packaged as a zip file.

Two features of the Eclipse debugger can be used to avoid unwanted suspension of execution. One feature is to add conditions to breakpoints. Another feature is to set the hit count for a breakpoint: the breakpoint will stop the execution only if it has been (silently) hit a specific number of times. To choose the appropriate hit count or condition, the developer should have considerable knowledge about the logic behind the source-code being debugged.

Another way to avoid unwanted suspension is to avoid using breakpoints at first. Instead, the developer can manually insert code snippets at the desired locations in the source code. These snippets print out the values that need to be checked to the console. After the code is executed, the developer can check the console to spot unexpected values. This mimics a debugging aid offered by some debuggers called tracepoints. In this work we propose a visual method for tracing Java programs in Eclipse. Our method fills a place in the middle between traditional stop-and-go debugging and full tracing.

Eclipse was created by OTI and IBM teams responsible for IDE products. Eclipse [3] is a platform that has been designed for building integrated web and application development tooling. The value of the platform is that it encourages rapid development of integrated features based on a plug­in model. Eclipse has a wide community of tool developers.

At the core of Eclipse there is an architecture for dynamic discovery, loading, and running of plugins. The platform handles the logistics for finding and running the right code. The platform UI provides a standard user navigation model. Each plug­in can then focus on a small number of well-defined tasks.



**Figure 1.1** Eclipse IDE

**Open architecture**

The platform itself is built in layers of plugins, each one defining extensions to the extension points of lower level plug­ins, and in turn defining their own extension points for further customization. This extension model allows plug­in developers to add a variety of function to the basic tooling platform.

The platform manages the complexity of different runtime environments. Plug­in developers can focus on their specific task instead of worrying about these integration issues. Eclipse was created as a platform for plug-in tools that extend the IDE’s capabilities so that it can work with numerous programming languages and applications, as Figure 1 shows. Anyone can write plug-ins for Eclipse and have them work directly with any other plug-ins for the platform. Some other IDEs limit plug-in creation to company partners.

There is thus a “huge number” of interoperable third-party plug-ins, which has made Eclipse very popular, said John Andrews, Evans Data’s chief operating officer. IBM’s release of Eclipse to the Eclipse Foundation made the technology independent of any company, which fuelled its broader adoption by businesses that don’t want to be tied to a specific vendor, noted Rob Cheng, Borland’s director of product marketing. “The more independent Eclipse is, the more comfortable companies and developers feel using it,” he explained.

For example, Oracle is working to ensure that any developer using Eclipse can build applications for the vendor’s application server and database, explained Ted Farrell, chief architect in the company’s Application Development Tools Division.

**Lower costs**

The entire Eclipse development platform is free. Proprietary IDE systems such as JBuilder, JDeveloper, and JetBrains’ IntelliJ IDEA, on the other hand, can cost up to $3,500 each.

Users seeking to add plug-ins that aren’t part of Eclipse can get some tools for free and pay for others. Either way, it can be less expensive than buying an entire proprietary development platform.Eclipse’s popularity has led many tool developers to make their products compatible with it. This competition has reduced the price of the plug-ins that aren’t free, according to Evans Data’s Andrews.

**Fast-moving innovation and development**

Because Eclipse is open source, Borland’s Cheng said, developers have ready access to the source code and can modify it and innovate quickly to meet users’ needs And, Cheng added, companies like the technology’s open development process. “It is a very transparent process. Most of the communications, milestones and plans are public, and the builds are available for public download. Interim builds come out every couple weeks or every month so that people can try it out and give feedback quickly. There is a lot of community involvement,” he said.

**Elegant architecture**

According to Cheng, Eclipse is a small, modular IDE with an elegant architecture that starts from a basic butpowerful foundation. “There is a layer that lets you integrate applications without worrying about drawing dialog boxes, buttons, and widgets or property pages and project trees,” he explained.

Thus, he elaborated, developers can hand code the new elements they need or want and disregard the elements that stay the same from program to program, such as dialog boxes.

**Not a total eclipse**

Although it appears to be the Java IDE of choice, Eclipse still faces competition from alternatives such as JBuilder, Visual J#, JDeveloper, and NetBeans. “Eclipse is certainly a very popular IDE and very successful,” said Tim Cramer, software engineering director for Sun’s NetBeans. “However, NetBeans is seeing a resurgence with the developer community. Eclipse has been great competition, and because of this, we’re all going to improve.” “With NetBeans 4.1,” he added, “we now have a number of features that add value above and beyond what Eclipse might have: visual development ofJ2ME (Java 2 Platform, Micro Edition) applications, debugging on a live cell phone, and an advanced profiler.” Thus, he said, “Our active users have gone up dramatically according to our internal measures, and we’re also seeing a surge in traffic to our Web site.”

Professional developers frequently work in teams, and Eclipse is up to the task. Eclipse supports the Concurrent Versions System (CVS) for this purpose. If you’re working in a team, you have to coordinate your development work with others to avoid conflicts. You’re all sharing the same code, which means your work of genius might be destroyed unintentionally by someone else’s thoughtless efforts. Source control precludes those kinds of problems because it controls access to shared code in a well-defined way. Besides controlling access to code, source control maintains a history of changes so that you can restore the code from earlier versions. Because it maintains a history of your code, not only can you restore code against earlier versions, but you can also compare the current code to earlier versions to see the differences at a glance. Like much else in the Java world, CVS is an open source project. CVS first appeared in 1986, when it was a set of UNIX shell scripts; it wasn’t until 1989 that dedicated CVS software first appeared. Today, CVS is available on many operating systems across the board, from UNIX and Linux to Windows.