D.P. VIPRA COLLEGE, BILASPUR (C.G.)

Department of Computer Science

Presented By :- Miss. Anjali Sona

CONCEPT OF JAVA IDE & GARBAGE COLLECTION

What is an IDE?

An integrated development environment (IDE) integrates popular developer tools into a single graphical user interface for developing applications (GUI).

An IDE typically consists of the following components:

Source code editor: A text editor can help write software code by offering features such as syntax highlighting with visual prompts, language-specific auto-completion, and bug testing as code is written.

Local build automation: Utilities that automate quick, repeatable tasks such as compiling computer source code into binary code, packaging binary code, and running automated tests to create a local build of the software for use by the developer.

Debugger: A program that can graphically represent the position of a bug in the original code and is used to evaluate other programs.

Advantages of using an IDE

Using an IDE will save you a lot of effort while writing Java programs. Some advantages include:

When writing a program, using an IDE will save you a lot of time and effort. Among the benefits are:

1. Saves time and effort: An IDE's whole aim is to make development faster and easier. Its tools and features are designed to help you manage your resources, avoid errors, and save time.

2. Enforce project or organization standards: A community of programmers can adhere to a common way of doing things simply by operating in the same production setting. If the IDE provides predefined models, or if code libraries are shared between different team members/teams working on the same project, standards can be applied even more.

3. Project management: There are two forms of project management. First, several IDEs have documentation tools that either automate developer comments or require developers to write comments in multiple areas. Second, providing a visual representation of resources can make it much easier to understand how an application is set out rather than searching the file system for arcane files.

Disadvantages of Using an IDE

Be mindful of some of the disadvantages of using an IDE, as it might not be sufficient for all or in every case.

1. Learning curve: IDEs are challenging to master. It will take time and patience to optimize their profit.

2. A powerful IDE might not be the best tool for new programmers: It can be complicated to learn how to program while still learning how to use an IDE. Furthermore, for skilled programmers, features and shortcuts frequently mask important yet tedious specifics of a language. When learning a foreign language, it's essential not to forget the little things. Using an IDE will make learning a new language more difficult.

3. Won't fix bad code, procedures, or design: You'll always need to be skilled and thorough. An IDE will not fix the application's efficiency or performance issues. Paintbrushes are analogous to IDEs. Your ability and decisions will determine whether you make a Van Gogh or a Velvet Elvis.

Now let's take a look at the top 10 most common Java IDEs.

Top 10 Java IDEs

First, we'll learn about Eclipse, the most widely used IDE.

Eclipse

Eclipse is an integrated development environment (IDE) for creating applications in Java and other programming languages such as C/C++, Python, PERL, Ruby, and others.

Eclipse is a cutting-edge, cross-platform, open-source, and freely distributed IDE for Web creation in the enterprise. In 1998, IBM Software Group envisioned an IDE that could succeed and lead in the competitive IDE industry. Because of this, Eclipse became known as one of the best Java IDEs for web creation.

The Eclipse platform, which serves as the basis for the Eclipse IDE, is made up of plug-ins and is intended to be extensible with new plug-ins. The Eclipse framework developed in Java can build rich client applications, integrated development environments, and other tools. Any programming language for which a plug-in is available can be used with Eclipse as an IDE.

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

It is a Java IDE that is known as one of the best Java IDEs. Both desktop and cloud versions of the common integrated development environment are available. Eclipse Che is the Eclipse cloud version, which helps programmers to create software using a web browser.

Eclipse, which is mainly written in Java, is an example of one of the best Java applications. It's designed to be a robust, feature-rich, commercial-grade platform for building modern web applications.

Significant features of Eclipse IDE

- Supports Java versions 8.0 and 9.0.
- Assists you with code refactoring, editing, gradual compiling, cross-referencing, and code recommendations.
- Static code review that is integrated.
- Offers intelligent code completion and fast fixes.
- Outstanding efficiency and usability.
- Windows, Linux, and Mac OS X support
- A PDE (Plugin Development Environment) is available for Java programmers who want to

create exceptional functionality for Eclipse.
• Eclipse has powerful charting, modeling, reporting, and testing tools to help Java developers speed up application development.

IntelliJ

IntelliJ Concept is a Java-based integrated development environment (IDE) for developing applications. IntelliJ Concept is often known as one of the best Java programming environments. JetBrains (formerly IntelliJ) developed it, and it is available in two versions: an Apache 2 Licensed community edition and a proprietary commercial edition, a paid version. Both can be used to grow a company. IntelliJ Concept includes cross-language refactoring and data flow analysis to help developers dig deeper into Java code.

The community edition of IntelliJ

The Community Edition is a free, open-source IDE for Java Virtual Machine (JVM) and Android development.

It supports Java, Kotlin, Groovy, Scala, Maven, Gradle, SBT, Git, SVN, Mercurial, CVS, TFS, and Maven, Gradle, SBT, Git, SVN, Mercurial, CVS, and TFS.

The ultimate version

The Ultimate edition supports Perforce, and other version control systems and is optimized for web and business development. JavaScript and TypeScript are supported, and Java EE, Spring, GWT, Vaadin, Play, Grails, and other frameworks. SQL and database tools are also provided.

NetBeans

Another essential Java IDE is NetBeans. The official IDE for Java 8 is NetBeans. It enables the creation of applications from a series of modular software components known as modules. It's compatible with Windows, Mac OS X, Linux, and Solaris. Identical to the other IDEs.

Features:

• It's an open-source IDE that's simple to set up, runs on various platforms, and is simple to use.
• Its adaptability extends to mobile growth, making it a standard IDE among mobile developers.
• It can also be expanded for plug-ins by a third-party Java development team at a later date.
• The Java editor in each new version of NetBeans has been enhanced and reworked.
• By highlighting Java code both syntactically and semantically, the Java editor makes it easier for programmers to create custom software applications.
• NetBeans' tools also assist developers in refactoring and writing bug-free code.

JDeveloper

Oracle Corporation offers JDeveloper as a freeware IDE. It supports Java, XML, SQL, PL/SQL, HTML, JavaScript, BPEL, and PHP development. JDeveloper handles the entire development process, from design to coding, debugging, optimization, profiling, and deployment.

Oracle's mission with JDeveloper is to make application development simpler by focusing on providing a visual and declarative approach to application development and a robust coding environment. Oracle JDeveloper works with the Oracle Application Development Framework (Oracle ADF), a Java EE-based end-to-end framework that makes application development much easier.

Features:

• It speeds up the development of Java-based applications by approaching each stage of the development process.

• It primarily provides an integrated development environment with a variety of features and many visual development tools.
• This Oracle JDeveloper can be used in combination with the Oracle Application Development Framework (Oracle ADF) to make application development even more accessible.
• JDeveloper can be used to create HTML, JavaScript, PHP, SQL, and XML in addition to Java.
• It also includes coding, designing, debugging, optimization, profiling, and deployment as part of the development lifecycle.

MyEclipse

MyEclipse is a commercial Java EE IDE developed and maintained by Genuitec, an Eclipse Foundation founding member.
MyEclipse is a programming environment based on the Eclipse platform that incorporates both proprietary and open-source code.
Aside from the Blue Edition, Spring Edition, and Bling Edition listed below, MyEclipse has two previous versions: Professional and Regular. The Standard edition adds database tools, a visual web builder, persistence tools, Spring tools, Struts and JSF tooling, and several other features to the basic Eclipse Java Developer profile. While it competes with the Web Tools Project, which is part of Eclipse, MyEclipse is a separate project with a different feature set.
MyEclipse runs on a variety of platforms, including Windows, Linux, and Mac, and includes industry-leading features like:
• Advanced JavaScript features in Ajax and Web 2.0 tooling.
• Spring and Hibernate incorporation
• Maven configurations are provided.
• Support for Swing GUI style.
• Tools for advanced reporting.
• Java Persistence tooling that is industry-leading, as well as a lot more.

BlueJ

BlueJ is an IDE for the Java programming language explicitly produced for educational purposes and is suitable for small-scale software development. JDK is used to make it work (Java Development Kit).
BlueJ was designed to assist in the learning and teaching of object-oriented programming, and as a result, its architecture is distinct from that of other development environments. The main screen graphically portrays the class structure under construction, and objects can be generated and evaluated interactively. This interaction capability, combined with a clean, simple user interface, makes it simple to play with evolving objects. Object-oriented concepts (classes, objects, and communication through method calls) are visually represented in the GUI and interaction design.

Features:

• It is still commonly used by Java programmers all over the world.
• Has a clutter-free interactive GUI that is quick to use and test.
• It’s also an excellent place to start for a beginner.
• Classes are represented by boxes in this GUI.
• The Java IDE is a cross-platform IDE that allows programmers to communicate with its objects in a fluid manner.
• It’s simple to look at object values, call object methods, and transfer objects as parameters.
• BlueJ accelerates the development of Java applications by offering a range of powerful

features.
• It includes an editor that allows developers to visually search the code, assist in creating dynamic objects, and inspect them.
• You can also use the Java code without having to compile it by simply typing it in.

JCreator

JCreator is a Java Integrated Development Environment (IDE) developed by Xinox Software. It has a similar user interface to Microsoft's Visual Studio. Xinox Software claims that JCreator is faster than competing Java-based Java IDEs since it is entirely written in C++ (except for the first version (0.1), which was Java-based).

Features:

• It comes in three different editions: Lite Edition, Pro Edition, and Life-Pro Edition.
• JCreator's paid edition includes Ant support, code wizards, and a debugger.
• JCreator lacks advanced functionality as compared to other Java IDEs. Extensibility via third-party plugins is also not possible.
• On the plus side, it's small and fast, making it perfect for beginners learning Java.
• JCreator is written entirely in C++, even though there are other standard Java IDEs. Furthermore, it does not require the use of a JRE to run Java code.

The developer group claims that JCreator is faster than most traditional Java-based IDEs for this purpose.

DrJava

DrJava is a Java integrated development environment (IDE). The JavaPLT community at Rice University actively develops and maintains the GUI, mainly for beginners, and uses Sun Microsystems' Swing toolkit. It has a consistent look across all platforms. DrJava will test Java code from a console interactively and show the results to the same console. It also has many other features that are geared for advanced users. JUnit testing is available in DrJava.

Features:

• DrJava includes the ability to interactively test Java code from a console and display the results in the same console.
• It has features like go to line and find/replace that support programmer.
• It includes auto-completion, automatic indentation, brace matching, commenting, and syntax coloring for experienced programmers.
• DrJava can also be inserted into Eclipse using a plugin.
• Unlike other Java IDEs, DrJava retains a consistent appearance across platforms. It's because Sun Microsystems' Swing toolkit was used to build it.

jGRASP

jGRASP is a programming environment that allows you to create software visualizations automatically. It generates both static and runtime visualizations of source code structure and data structures.

jGRASP is written in Java and can be used on any Java Virtual Machine platform (Java version 1.8 or higher). GRASP (Linux, UNIX) and pcGRASP (Windows) are C/C++ programs, while jGRASP is a Java program (the "j" in jGRASP means it runs on the JVM). Downloads for Windows, Mac OS, and a generic ZIP file suitable for Linux and other systems are available on the jGRASP website.

jGRASP is a source code editor for languages other than Java. It can be set up to work with any programming language and the most accessible and commercial compilers.

Features:

• It is explicitly designed to create software visualizations automatically, which increases the overall comprehensibility of any software.
• During runtime, the lightweight Java IDE will generate static visualizations of source code structure and visualizations of data structures.
• jGRASP can generate CSDs (Control Structure Diagrams) for other programming languages, despite being written in Java.
• ADA, C, C++, Objective-C, and Python are all on the list.
• It can also be used as a source code editor for other languages. Many commercial compilers for various programming languages can be configured to operate with the free IDE.

JSource

JSource is a small Java IDE that is entirely made up of Swing components. It comes with a simple but powerful editor that allows you to create, edit, compile and run Java files. Syntax highlighting is also supported for other programming languages.

Features:

• JSource is released under the GNU General Public License, version 2.0. (GPLv2).
• It helps in the creation of cross-platform applications for a variety of domains.
• It's exceptionally light. JSource helps you to execute, compile, modify, and build Java files.
• Syntax highlighting for various languages and Java Swing components is one of the key features.
• You can use jEdit syntax packages in version 2.0 of JSource, as well as other open-source Java software for rapid development.
• Several methods have been tweaked to work with the JSource framework.

Garbage Collection in Java

When you create an object by instantiating a class, the object uses some memory. After an object in memory has been used and is no longer needed, it is sensible to free memory from that object. Unlike some object-oriented languages where you have to explicitly destroy the objects that are no more references to that object. References that held in a variable naturally dropped when the variable goes out of scope. Alternatively, we can explicitly drop an object reference by setting the variable to null. which is a tedious and error-prone task, Java follows a different approach.

Java allows you to create as many objects as you want without any worry about destroying them. The Java Runtime Environment (JRE) deletes objects automatically when it determines that they no longer used. The objects which no longer used termed as garbage, and the process of deleting these objects automatically is known as **garbage collection**.

The Java garbage collector is a **mark-sweep garbage collector**. It scans active memory areas for objects and marks those that are referenced and then all the unmarked objects Le. Unreferenced objects which considered as garbage are collected.

A separate garbage collector thread always runs at low priority. It can run both synchronously and asynchronously. It runs synchronously when the system resources are low or when it has been forced to run by a call to the collector.

This garbage collector thread checks the dynamic memory area and marks all those areas that are referenced. After sweeping through the area, it cleans up all the areas that were not referenced. It checks all possible areas in which the objects could have been referenced. The garbage collector normally calls the object's finalize method if one was written, before it works on that object. This allows the object itself to clean up all the system resources that it allocated.

The Java runtime environment runs the garbage collector automatically. But if your program requires that the garbage collector is run at a particular convenient time, you can force it using the System.gc() method.

An object is eligible for garbage collection when

• You drop an object reference by setting the variable to a special value null.
• As soon as the variable goes out of scope, the references in the variable are automatically dropped.

In Java, the memory is allocated to the objects using 'new' operator. In languages like C++, the memory is deallocated using 'delete' operator. This process of deallocating memory is called *garbage collection*. Java deallocates the memory itself. When no reference to an object is found, the object is assumed to be not needed any more. So, the memory allocated to that object can be reclaimed. There is no need to explicitly destroy objects.

Garbage collection is done automatically using a garbage collector. The JVM executes its garbage collector for retrieving the memory of objects that no longer used so that the memory can use for other objects. Therefore, memory leaks which are common in C++ due to inability to perform automatic garbage collection are less likely in Java.

In other languages like C++, the programmer needs to ensure that an object's memory is de-allocated. If the C++ programmer is not careful about these de-allocations, the program may end up with memory "leaks." The ill effects of these are all too familiar-program crashes, more paging, swapping, slower system response time, and the like.

Garbage collection takes time, so the Java run-time system does it only when necessary. It does not occur just because one or more objects exist that no longer used. For efficiency, the garbage collector usually runs when there are objects to recycle, and there is a need to recycle them. You can't know precisely when garbage collection takes place.

There may occur a situation when the object that destroyed the need to perform some action. To handle such situations, Java provides 'a mechanism called finalization. So, we need to add a finalize () method to our class.

The finalize() method of an object is the opposite of its constructor. The finalize() method is where resources that are allocated by the object can free.

In Java, when an object no longer referenced, the Java runtime environment runs the object's finalize() method before reclaiming the object's memory. The programmer can also force this process synchronously by calling the following method:
System.runFinalization();

Every object has a default finalize() method associated with it. You can override the default behavior by writing code in an object's finalize() method. Reasons to override default behavior include de-allocation of other resources that an object may have allocated, such as files, streams, and so on.
This default method is declared as follows:

protected void finalize() throws Throwable

NOTE. It is usually a good practice to call the finalize method of the superclass if the superclass has a finalize method. It enables the resources used by the superclass to be cleared up.

You can also call the finalize() method of an object in your program, as shown just below. It does not, however, guarantee that the object's memory reclaimed immediately. That is a function of the garbage collector thread.

```
protected void finalize() throws Throwable {
   // clean up code for this class here
   super.finalize();
}
```

Finalize methods always have a return type of void and override a default destructor in java.object.Object.

Usually the finalize methods are used to free up all the system resources, closing up the files, sockets, and so on. Here a PrintStream and DataInput-Stream which opened earlier are closed in its finalize method.

We need not worry about the use of finalize () in our programs since it used in sporadic cases where the object is not under the control of the garbage collector.

A program can call finalize directly just as it would any other method. However, calling finalize not initiate any garbage collection. It treated like any other method called directly. When Java does garbage collection, finalize is still called even if it has already been called directly by the program.

The finalize method can be overloaded also. If Java finds a finalize method with arguments at garbage-collection time, it looks for a finalize method with no arguments. If it does not find one, Java uses the default finalize method instead.

The system only calls finalize when it is ready to reclaim the memory associated with the object and not immediately after an object no longer referenced. That is, finalize() is called only when the system is running short of memory so that it may be called long after the application finishes.