**Differences between applets and applications**

In Java, **applets** and **applications** are two types of programs, each serving different purposes and running in different environments. Here's a breakdown of the key differences between the two:

**1. Definition and Purpose**

* **Java Applications**: A Java application is a stand-alone program that runs directly on the Java Virtual Machine (JVM). It is typically executed from the command line or by double-clicking the executable file. Applications can be graphical (GUI) or console-based and have full access to the system resources.
* **Java Applets**: An applet is a small, browser-based program that runs within a web browser or an applet viewer. Applets are embedded in web pages and are used to provide dynamic content on websites. However, Java applets are largely obsolete today due to security concerns and the decline in browser support.

**2. Execution Environment**

* **Java Applications**: Run directly on the **JVM** on the user's system. The JVM can be launched through a command line, an IDE, or an executable JAR file. Applications don't rely on a web browser to run.
* **Java Applets**: Run inside a **web browser** or an applet viewer like the Java Plug-in. Applets are typically embedded in HTML pages using the <applet> or <object> tags. They depend on the browser's JVM plugin to execute.

**3. Lifecycle**

* **Java Applications**: The lifecycle of a Java application is controlled by the main() method. The program starts executing when the JVM invokes the main() method and terminates when the program finishes executing.

java

Copy code

public class MyApplication {

public static void main(String[] args) {

System.out.println("This is a Java application.");

}

}

* **Java Applets**: Applets have a well-defined lifecycle managed by the browser or the applet viewer. An applet typically has the following lifecycle methods:
  + init(): Initializes the applet.
  + start(): Starts or resumes the execution.
  + stop(): Pauses the applet.
  + destroy(): Cleans up resources when the applet is no longer needed.

Example of an applet class:

import java.applet.Applet;

import java.awt.Graphics;

public class MyApplet extends Applet {

public void init() {

System.out.println("Applet initialized.");

}

public void paint(Graphics g) {

g.drawString("Hello, Applet!", 50, 50);

}

}

**4. Security Restrictions**

* **Java Applications**: Applications have full access to the system's resources (such as file system, network, and local storage). They can read/write files, open network connections, and perform operations that require permissions from the OS.
* **Java Applets**: Applets are **sandboxed** for security reasons. They have restricted access to the local system. For instance:
  + Applets cannot read/write files from the local file system.
  + Applets can only make network connections to the host from which they were downloaded.
  + These restrictions are meant to prevent malicious applets from damaging the user's system.

**5. User Interface**

* **Java Applications**: Applications can have rich, fully-featured user interfaces using libraries like **Swing**, **AWT**, or **JavaFX**. They can run as command-line interfaces (CLI) or graphical user interfaces (GUI).
* **Java Applets**: Applets generally have simpler UIs that run inside the browser's space. They can also use **AWT** and **Swing** for their graphical interfaces, but their UI is constrained to the small, embedded region on a web page.

**6. Deployment**

* **Java Applications**: Java applications are deployed by packaging the .class files or a .jar (Java Archive) file and running it on any system with a JVM installed. They can also be bundled as executables or run on servers in the case of enterprise applications.
* **Java Applets**: Applets are deployed via web browsers. An HTML file includes the applet code (using the <applet> or <object> tag), and the applet's bytecode is downloaded and run on the client's browser. Applets can be stored on web servers and downloaded dynamically as needed by the browser.

**7. Usage**

* **Java Applications**: Used for general-purpose software, such as desktop programs, server-side applications, utilities, and more. Examples include text editors, games, enterprise applications, and database-driven systems.
* **Java Applets**: Used to provide dynamic content on web pages, such as small games, interactive charts, or animations. However, applets are now obsolete, replaced by more modern web technologies like **JavaScript**, **HTML5**, and **CSS**.

**Key Differences Summary**

| **Feature** | **Java Applications** | **Java Applets** |
| --- | --- | --- |
| **Execution Environment** | Standalone, runs on JVM. | Runs within a web browser (or applet viewer). |
| **Lifecycle** | Starts with the main() method. | Managed by init(), start(), stop(), destroy(). |
| **Security** | Full access to system resources. | Restricted access (sandboxed). |
| **User Interface** | Rich UIs via Swing, JavaFX, or CLI. | Simple UIs, constrained to browser window. |
| **Deployment** | Packaged as .jar or .class files; standalone. | Embedded in web pages and downloaded dynamically. |
| **Purpose** | General-purpose software. | Web-based interactive content. |
| **Current Usage** | Widely used in various domains. | Largely obsolete, deprecated since Java 9. |

**Conclusion:**

* **Java Applications** are widely used and are designed to be standalone programs with full access to system resources.
* **Java Applets**, though once popular for embedding dynamic content into web pages, are now considered obsolete due to security concerns and the rise of modern web technologies like JavaScript and HTML5. Most browsers no longer support applets, and the technology is officially deprecated.