**Java 11 introduced several new features and improvements over Java 8. Here are some of the major differences:**

**Local Variable Type Inference:** Java 11 introduced local variable type inference, which allows you to declare a variable without explicitly specifying its type. Instead, the type is inferred from the value on the right-hand side of the assignment operator.

Local Variable Type Inference (also known as "var") is a feature introduced in Java 10 and improved in Java 11 that allows you to declare a local variable without specifying its type explicitly. Instead, the type of the variable is inferred from the expression on the right-hand side of the assignment operator.

Here's an example:

var message = "Hello, World!"; // inferred type: String

In this example, the type of the variable "message" is inferred to be a String because the expression on the right-hand side of the assignment operator is a String literal.

Local Variable Type Inference can make your code more concise and readable, especially when dealing with complex types or generic collections. However, it's important to use it judiciously and not rely on it too heavily, as it can make your code less clear if the inferred type is not obvious from the context.

**HTTP Client API**: Java 11 includes a new HTTP Client API that allows developers to send HTTP/1.1 and HTTP/2 requests directly from Java code. This API replaces the old HttpURLConnection API, which was difficult to use and lacked many modern features.

a new HTTP Client API that provides a more modern and flexible way to send HTTP requests and receive responses from web servers. Here are some key features of the HTTP Client API:

1. Asynchronous and Reactive: The HTTP Client API supports both asynchronous and reactive programming paradigms, which can help improve the scalability and performance of your applications.
2. Non-blocking I/O: The API uses non-blocking I/O, which allows multiple requests to be sent and received in parallel without blocking the application's main thread.
3. HTTP/1.1 and HTTP/2 Support: The API supports both HTTP/1.1 and HTTP/2 protocols, which can provide faster and more efficient data transfer between clients and servers.
4. **WebSocket Support**: The API also supports the WebSocket protocol, which enables **bidirectional communication** between clients and servers **over a single TCP connection.**
5. **Authentication and Authorization**: The API provides built-in support for various authentication and authorization mechanisms, such as Basic Authentication and OAuth 2.0.
6. **SSL/TLS Support**: The API supports SSL/TLS encryption for secure communication between clients and servers.
7. **Response Body Processing**: The API provides various ways to process and handle the response body, including streaming, **asynchronous** processing, and reactive programming.
8. **Customization and Extension**: The API is highly customizable and extensible, allowing developers to implement their own HTTP request/response filters and interceptors.
9. Overall, the HTTP Client API in Java 11 provides a more flexible and modern way to communicate with web servers, with support for multiple protocols, authentication mechanisms, and response processing options.

**Improved Security**: Java 11 comes with improved security features, including better TLS support and stronger encryption algorithms.

1. **TLS 1.3 Support:** Java 11 added support for the latest version of the Transport Layer Security (TLS) protocol, TLS 1.3, which provides improved security and performance over earlier versions.
2. **Stronger Cryptographic Algorithms**: Java 11 increased the minimum key size for certain cryptographic algorithms, such as RSA and DSA, to 2048 bits by default. This helps to mitigate the risk of attacks that can exploit weaknesses in shorter key sizes.
3. **Root CA Certificate Changes**: Java 11 updated its list of trusted root Certificate Authorities (CAs), removing some older or untrusted CAs and adding new ones.
4. **Enhanced JCE Provider**: Java 11 includes an enhanced version of the Java Cryptography Extension (JCE) provider, which supports additional cryptographic algorithms and provides improved performance.
5. **Secure Random Number Generation**: Java 11 improved the security of its random number generation by using a more secure default algorithm (SHA-512) and by periodically reseeding its internal entropy pool.
6. **Stronger Password-Based Encryption**: Java 11 introduced a new Password-Based Encryption (PBE) algorithm, PBKDF2WithHmacSHA512, which provides stronger security than the previous default algorithm, PBKDF2WithHmacSHA1.
7. Overall, these security improvements in Java 11 help to ensure that Java applications are **more secure and resistant to various forms of attacks**, such as man-in-the-middle attacks, data tampering, and unauthorized access to sensitive data.

**Flight Recorder**: Java 11 introduced Flight Recorder, a low-overhead data collection framework that can be used to diagnose and troubleshoot performance and stability issues in Java applications.

**Performance Improvements**: Java 11 includes several performance improvements, including faster startup times, reduced memory usage, and better garbage collection.

**Unicode 10**: Java 11 adds support for Unicode 10, which includes over 8,500 new characters, including emoji.

Java 11 introduced support for Unicode 10, which includes several new characters, scripts, and emoji characters. Here are some key features of Unicode 10 support in Java 11:

1. **New Scripts**: Unicode 10 adds support for several new scripts, including Masaram Gondi, Nushu, and Wancho. These scripts can now be used in Java applications and displayed correctly in user interfaces.
2. **New Characters**: Unicode 10 includes several new characters, such as new currency symbols, historic scripts, and emoji characters. These characters can now be used in Java applications and displayed correctly in user interfaces.
3. **Emoji 5.0 Support**: Unicode 10 also includes Emoji 5.0, which adds support for new emoji characters, such as the vomiting face, the hijab, and the exploding head. These emoji characters can now be used in Java applications and displayed correctly in user interfaces.
4. **Text Segmentation**: Java 11 also includes updated algorithms for text segmentation, which can help improve the accuracy and consistency of text rendering and processing in various languages and scripts.
5. Overall, Unicode 10 support in Java 11 provides improved support for internationalization and localization, with support for new scripts, characters, and emoji. This can help Java applications to better handle diverse text input and output, and to provide a more inclusive and accessible user experience

**Dynamic Class-File Constants**: Java 11 allows class-file constants to be dynamic, meaning they can be created at runtime rather than being hard-coded into the class file.

**Epsilon Garbage Collector:** Java 11 introduces a new experimental garbage collector called the Epsilon Garbage Collector. This collector is designed for use in scenarios where garbage collection is not necessary or can be performed less frequently.

1. Java 11 introduced a new **experimental** garbage collector called the **Epsilon Garbage** Collector (GC), which is designed to provide a no-op garbage collector with very low overhead. Here are some key features of the Epsilon GC:
2. **Minimal Overhead:** The Epsilon GC is designed to provide a no-op garbage collector with virtually no overhead, which means that it does not perform any garbage collection at all. This can be useful in certain situations where garbage collection is not necessary or where performance is critical.
3. **Predictable Behavior**: Since the Epsilon GC does not perform any garbage collection, it has a predictable and consistent behavior, which can help improve the **predictability** and **stability** of Java applications.
4. **Experimental Status**: The Epsilon GC is an experimental feature and is not recommended for production use. It is primarily intended for testing and experimentation purposes, to explore the performance and behavior of the Java Virtual Machine (JVM) with a no-op garbage collector.
5. Configuration Options: The Epsilon GC provides several configuration options, such as heap size, allocation limit, and memory usage thresholds, which can be used to customize its behavior and performance.
6. Overall, the Epsilon GC in Java 11 provides a new experimental option for developers who need a no-op garbage collector with very low overhead. While it is not recommended for production use, it can be a useful tool for testing and experimentation purposes.