**1. Default Methods (Introduced in Java 8)**

Default methods allow interfaces to have concrete methods with a default implementation. This was introduced to enable adding new functionality to interfaces without breaking existing implementations.

**Key Points:**

* Declared using the default keyword.
* Can be overridden by implementing classes.
* Used to provide backward compatibility.

**2. Static Methods (Introduced in Java 8)**

Static methods allow interfaces to have utility or helper methods that are associated with the interface itself rather than instances of the implementing classes.

**Key Points:**

* Declared using the static keyword.
* Can only be called using the interface name, not through an object.
* Cannot be overridden.

**3. Private Methods (Introduced in Java 9)**

Private methods in interfaces help to encapsulate code and avoid duplication in default and static methods within the same interface. They cannot be accessed by implementing classes.

**Key Points:**

* Declared using the private keyword.
* Used only within the interface.
* Cannot be overridden or accessed directly.

### Summary of Introduction Versions:

| **Feature** | **Introduced In** | **Purpose** |
| --- | --- | --- |
| **Default Methods** | Java 8 | Provide a default implementation for backward compatibility. |
| **Static Methods** | Java 8 | Allow utility methods within interfaces. |
| **Private Methods** | Java 9 | Encapsulate logic to avoid duplication in default/static methods. |

**Abstraction** in programming is a concept of hiding the complex implementation details and showing only the essential features of an object. It focuses on **what an object does** rather than **how it does it**. Abstraction helps in reducing programming complexity and increases efficiency by simplifying the interface.

**Key Points of Abstraction:**

1. It is implemented in **object-oriented programming** (OOP).
2. Achieved using **abstract classes** and **interfaces** in Java.
3. It hides unnecessary details and provides only relevant information to the user.

In Java, **100% abstraction** can be achieved using **interfaces**. An interface is a reference type in Java that contains only abstract methods (prior to Java 8) and constants. Since all methods in an interface are implicitly abstract and public, it ensures complete abstraction.

**Key Points of Interfaces:**

1. **Methods**: Only abstract methods (prior to Java 8). Default and static methods are allowed from Java 8 but do not affect abstraction.
2. **Fields**: All fields are public, static, and final.
3. A class implements an interface to define its behavior.

**Encapsulation** in Java is a mechanism of wrapping the data (fields) and code (methods) together as a single unit. It helps to restrict direct access to some of the object's components and only expose essential parts of the object to the outside world.

Encapsulation ensures **data hiding** by declaring the fields as private and providing public getter and setter methods to access and modify the data.

**Key Benefits of Encapsulation:**

1. Protects the integrity of an object's data by restricting direct access.
2. Increases maintainability and flexibility of the code.
3. Enables controlled access to data through getter and setter methods.