In Java, a nested class is a class defined within another class. There are four types of nested classes in Java:

1. **Static Nested Class:** A static nested class is a nested class that is declared as a static member of the outer class. It can be instantiated without an instance of the outer class. Static nested classes are often used for grouping related functionality within the outer class.

  ```java
  class Outer {
      static class StaticNested {
          // Code for static nested class
      }
  }
  ```

2. **Inner Class (Non-static Nested Class):** An inner class is a non-static nested class that is declared within another class. It has access to the members of the outer class, including private members. An instance of an inner class can only be created within an instance of the outer class.

  ```java
  class Outer {
      class Inner {
          // Code for inner class
      }
  }
  ```

3. **Local Inner Class:** A local inner class is a class defined within a method or a code block. It can access the local variables of the enclosing block if they are marked as `final` or effectively `final`.

  ```java
  class Outer {
      void someMethod() {
          class LocalInner {
              // Code for local inner class
          }
      }
  }
  ```

4. **Anonymous Inner Class:** An anonymous inner class is a type of local inner class that is defined without a class name, typically while creating an instance of an interface or an abstract class. It is used for one-time implementation.

  ```java
  Interface myInterface = new Interface() {
      // Anonymous inner class implementation
  };
  ```

Nested classes are often used for encapsulating related functionality and improving code organization. They also help in achieving better encapsulation by restricting access to certain class members within a specific scope. The choice of which type of nested class to use depends on the specific design and functional requirements of your program.