The **Multiton design pattern** is an extension of the Singleton pattern that ensures a limited number of instances of a class, each associated with a unique identifier or key. [It allows controlled creation of multiple instances, managed through a map or registry1](https://java-design-patterns.com/patterns/multiton/)[2](https://en.wikipedia.org/wiki/Multiton_pattern).

Here are **five reference links** where you can learn more about the Multiton pattern:

1. [Java Design Patterns: Multiton](https://java-design-patterns.com/patterns/multiton/): Provides an explanation, real-world example, and programmatic implementation in Java.
2. [Demystifying Design Patterns: Multiton](https://medium.com/@codechuckle/demystifying-design-patterns-multiton-pattern-859b50638a9f): A detailed article with code examples.
3. [Design Patterns Tutorial: Multiton](https://riptutorial.com/design-patterns/topic/6857/multiton): Covers the concept and implementation of the Multiton pattern.
4. [BlackWasp: Multiton Design Pattern](https://blackwasp.co.uk/multiton.aspx): Code example and explanation using C#.
5. [Wikipedia: Multiton Pattern](https://en.wikipedia.org/wiki/Multiton_pattern): A concise overview of the pattern.

Feel free to explore these resources to deepen your understanding of the Multiton pattern! 🚀