Flyweight Design pattern :

Flyweight pattern is one of the [structural design patterns](https://www.geeksforgeeks.org/design-patterns-set-1-introduction/) as this pattern provides ways to decrease object count thus improving application required objects structure. Flyweight pattern is used when we need to create a large number of similar objects (say 105). One important feature of flyweight objects is that they are **immutable**. This means that they cannot be modified once they have been constructed.

**Why do we care for number of objects in our program?**

* Less number of objects reduces the memory usage, and it manages to keep us away from errors related to memory like [java.lang.OutOfMemoryError.](https://docs.oracle.com/javase/7/docs/api/java/lang/OutOfMemoryError.html)
* Although creating an object in Java is really fast, we can still reduce the execution time of our program by sharing objects.

In Flyweight pattern we use a [HashMap](https://www.geeksforgeeks.org/hashmap-treemap-java/) that stores reference to the object which have already been created, every object is associated with a key. Now when a client wants to create an object, he simply has to pass a key associated with it and if the object has already been created we simply get the reference to that object else it creates a new object and then returns it reference to the client.

**Flyweight in the Real World**

The classic example of the Flyweight pattern is the representation of a character in a word processor. Rather than each character having separate glyph objects that represent the font and formatting data, each character could have a reference to a flyweight glyph object shared by every instance of the same character in the document. In this case, the character need only store it's position in the document, rather than it's entire formatting information.

**The Flyweight Pattern**

Flyweight is known as a **structural**pattern,as it's used to form large object structures across many disparate objects. The definition of Flyweight provided in the original Gang of Four book on DesignPatterns states:

***Facilitates the reuse of many fine grained objects, making the utilization of large numbers of objects more efficient.***

When considering this pattern, you will need to think about intrinsic and extrinsic data. **Intrinsic**data is the data that makes this object instance unique. Meanwhile, **extrinsic**data is information that can be passed in through arguments. So, if you can make some data extrinsic for cases that you have a large number of objects, the Flyweight pattern may be exactly what you are looking for.

References :

<https://www.geeksforgeeks.org/flyweight-design-pattern/>

<https://refactoring.guru/design-patterns/flyweight>

<https://dzone.com/articles/design-patterns-flyweight>