**Design Pattern🡪**

**Creational Design Pattern🡪**

**Builder Design pattern🡪**

The Builder Pattern is a creational design pattern used to construct complex objects step by step. It allows you to create an object by specifying only the required parameters and then optionally setting additional parameters through a series of method calls. This pattern is particularly useful when you have a class with many optional parameters or configurations.

Steps to create 🡪

1. Create class eg- User.
2. Add Field like name, id, email etc.
3. Then private Constructor.
4. Write getter method for all field.
5. Create another inner class with static keyword.
6. Add all field which added in outer class and make one public constructor.
7. Add setter with inner class return type.
8. And make one build method and return User (outer class);

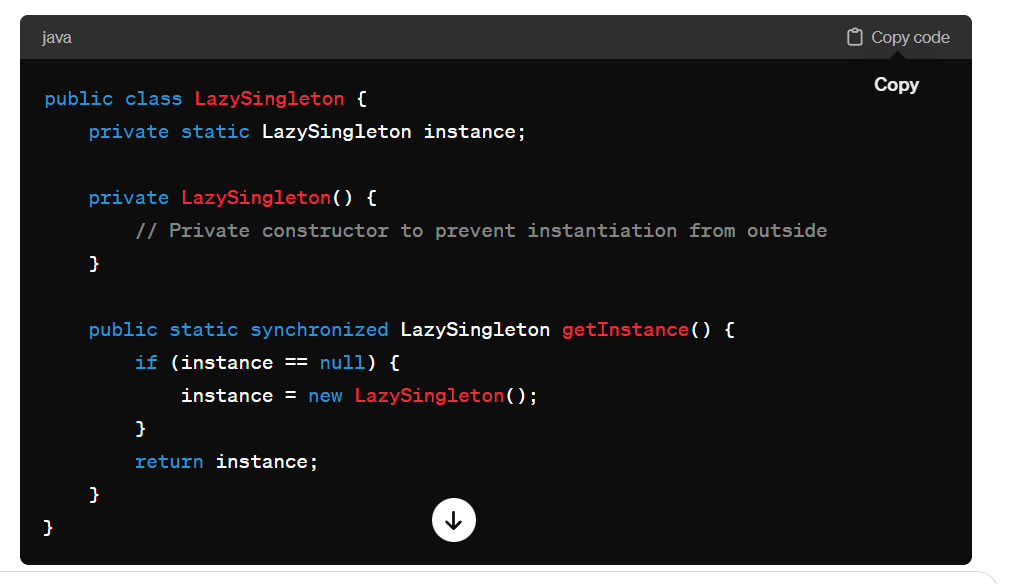
Example🡪



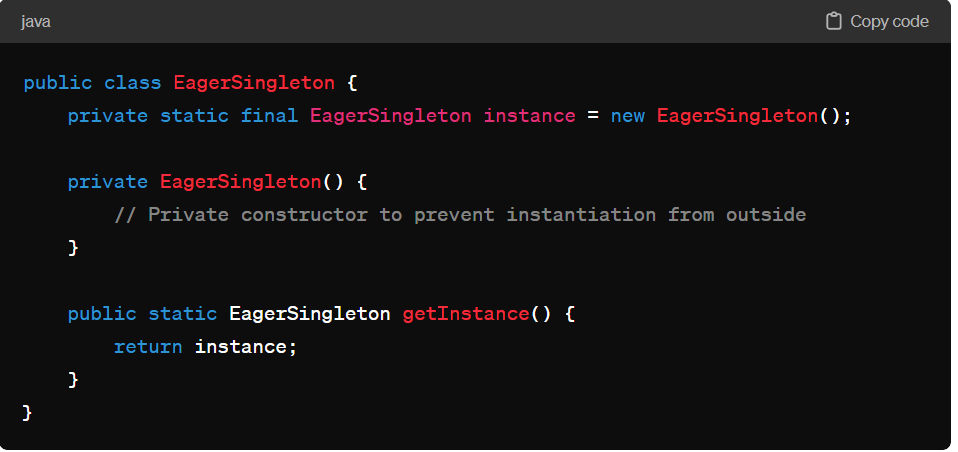
**Singleton pattern🡪**

The Singleton Pattern is a creational design pattern that ensures a class has only one instance and provides a global point of access to that instance. It's commonly used when exactly one object is needed to coordinate actions across the system.

Lazy Initialization🡪



Eager Initialization🡪



Note🡪 Use lazy initialization when you want to optimize memory usage, defer object creation, or reduce startup time, especially for non-essential or infrequently used objects. Use eager initialization when you prioritize thread safety, guaranteed availability, simplicity, or predictable performance. Ultimately, the choice depends on your application's specific requirements and performance considerations.