Creating an immutable object in Java means creating an object whose state cannot be changed after it is constructed. To achieve immutability, follow these principles:

1. **Declare the class as final** so that it cannot be subclassed.
2. **Make all fields private and final** to ensure they are not modified after the object is constructed.
3. **Provide only getters** for the fields, and no setters.
4. **Ensure that any mutable fields are deeply copied** to prevent modification through references.

*https://raw.githubusercontent.com/vsaravanan/java22/master/src/main/java/com/saravanjs/java22/console/corejava/Immutable.java*

*final class* Person {
 *private final* String name;
 *private final int* age;
 *private final* Date birthDate;

 *public* Person(String name, *int* age, Date birthDate) {
 *this*.name = name;
 *this*.age = age;
 *// Create a defensive copy of the mutable Date object
 this*.birthDate = *new* Date(birthDate.getTime());
 }

 *public* String getName() {
 *return* name;
 }

 *public int* getAge() {
 *return* age;
 }

 *public* Date getBirthDate() {
 *// Return a defensive copy of the mutable Date object
 return new* Date(birthDate.getTime());
 }
}
*public class* Immutable {
 *public static void* main(String[] args) {
 Date birthDate = *new* Date();
 Person person = *new* Person("John Doe", 30, birthDate);

 System.*out*.println("Name: " + person.getName());
 System.*out*.println("Age: " + person.getAge());
 System.*out*.println("Birth Date: " + person.getBirthDate());

 *// Attempt to modify the birthDate object* birthDate.setTime(0);
 System.*out*.println("Variable : " + birthDate);

 *// Verify that the birthDate inside Person is not affected* System.*out*.println("Modified Birth Date: " + person.getBirthDate());
 }
}

Name: John Doe

Age: 30

Birth Date: Mon Jun 17 23:06:06 SGT 2024

Variable : Thu Jan 01 07:30:00 SGT 1970

Modified Birth Date: Mon Jun 17 23:06:06 SGT 2024