**Key things you should remember about the record class in Java:**

**1. Final Class**

* A record class is implicitly final, which means it cannot be subclassed. You cannot extend a record class, as it’s designed to be a data carrier, not intended for inheritance.

**2. Immutable Fields**

* All the fields in a record are automatically public and final.
* You cannot modify the fields after the record is created (i.e., they are immutable).
* This ensures the data contained in the record is not changed once it is initialized.

**3. Automatic Method Generation**

* toString(): By default, a record class has a toString() method that prints all the fields in the format RecordName[field1=value1, field2=value2, ...].
* equals(): The equals() method is automatically generated to compare records based on the values of their fields. Two records are considered equal if their fields have the same values.
* hashCode(): The hashCode() method is automatically generated based on the fields of the record. This ensures consistency with the equals() method.

**4. Compact Constructor**

* The constructor is implicitly generated for the record, based on the fields declared in the record. You can add additional validation or logic in the constructor if necessary.
* You can’t omit the parameters from the constructor; the constructor automatically accepts parameters that match the record's fields.
* You can also customize the constructor to include validation or transformation logic.

**5. Custom Methods**

* You can define instance methods inside a record to add custom behavior (e.g., utility functions).
* Static methods can also be defined, just like regular classes. However, they don’t have access to the fields of the record unless they are passed in as parameters.

**6. Cannot Extend Other Classes**

* Records cannot extend other classes. However, they can implement interfaces. If needed, a record can implement one or more interfaces, but it cannot extend another class.

**7. No Setter Methods**

* Since records are immutable, you cannot add setter methods to modify the fields of the record after it’s created.

**8. Descriptive toString() Output**

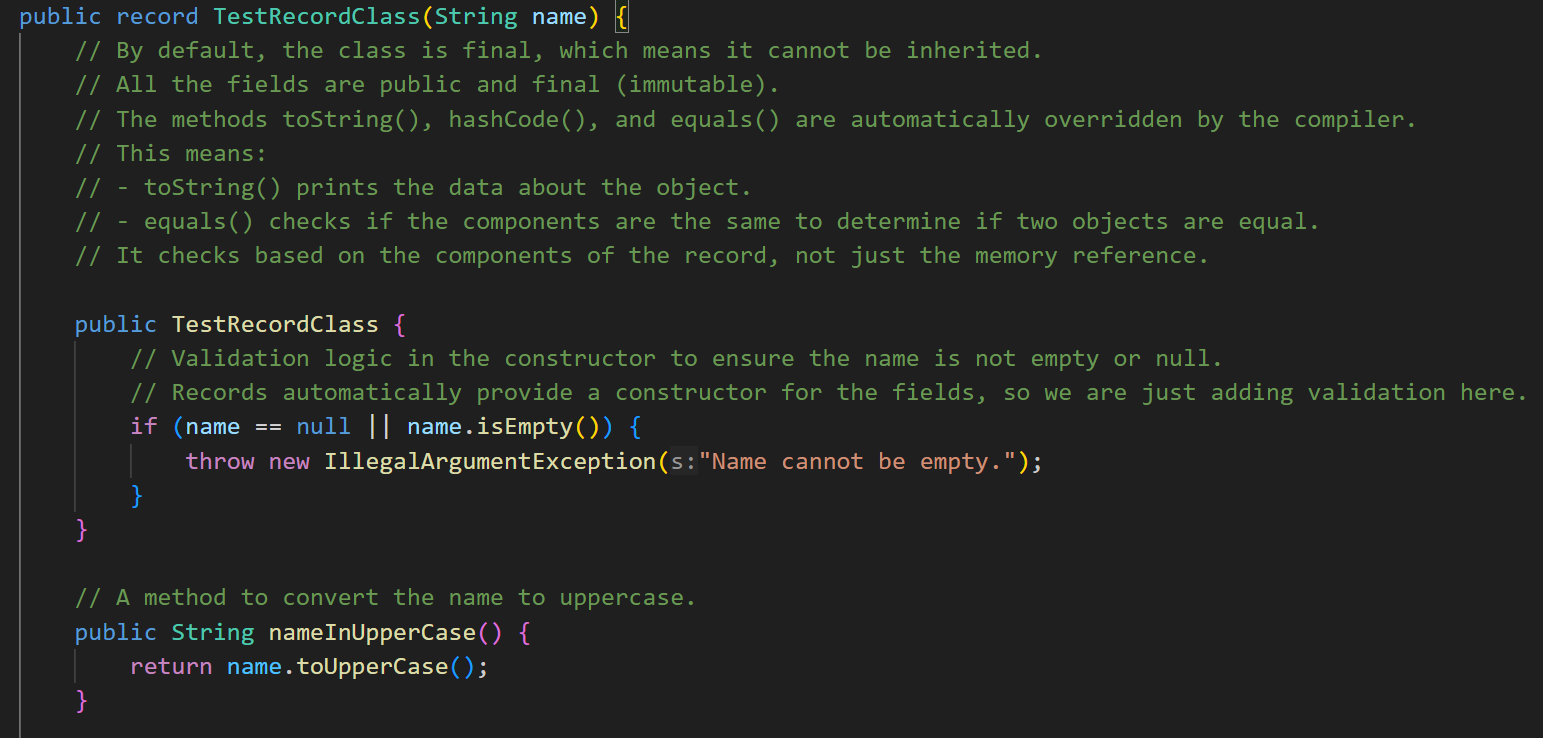
* The automatically generated toString() method gives you a useful representation of the record instance with its field names and values, which is great for debugging**.**

**9. Canonical Constructor**

* The canonical constructor is automatically provided, which takes in arguments for all fields in the record. It’s a concise way to create a record.
* You can define an additional constructor with validation or custom logic, if necessary, but the canonical constructor always exists.

A black screen with white text

AI-generated content may be incorrect.



A computer screen shot of a program

AI-generated content may be incorrect.

A screenshot of a computer program

AI-generated content may be incorrect.

Points to remember:

* 1. Final Class.
  2. Final and private fields.
  3. Can add Instance methods and static methods.
  4. Can not extend other classes but can implement multiple interfaces.
  5. No setter methods only getter methods. To use any getter methods just do className.variableName()
  6. Canonical constructors.
  7. Descriptive toString method is present
  8. Equals method is also present which compares records by the value of components.