Records in Java 14+

Are special type of class that help **avoid boilerplate code**. They are considered “data carriers.”

- The data in the record is held in private final fields and there is only a getter method. Therefore, data in the record is immutable.
- A record cannot inherit another class, because implicitly inherit from java.lang.Record class.
  As such, overrides equals(), hashCode(), toString() methods of the Object class.
- A record can implement one or more interfaces.
- All the instance fields are listed as “components” in the record declaration. Any other fields, except the list of components, must be declared static.
- Records are immutable and final by default.
- Records can have both static fields and static methods.
- Records can have instance methods.

Syntax:

```
record recordName(list-of-components) {
        //optional statements
}
```

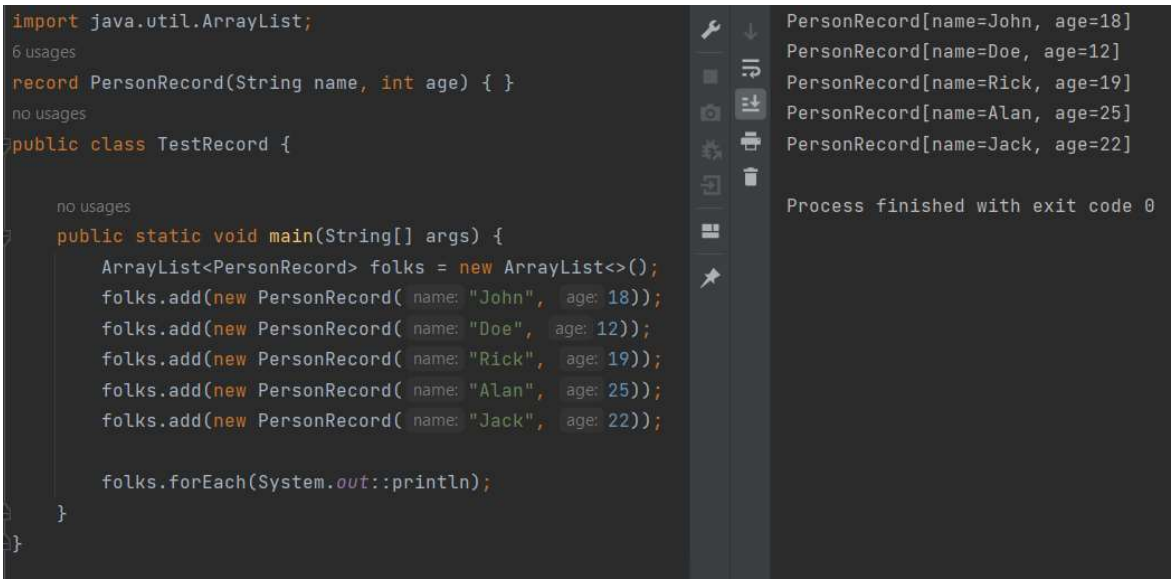
Example:

```
public record PersonRecord(String name, int age) { }
```

Here is the implicitly generated code from the statement above:

- Canonical (all-arguments) constructor
- public accessor methods with the same name as the components.
- toString () the string representation of all the record class's components , with their equals() and hashCode() which specify that two record classes are equal if they are of the same type and contain equal component values.
- You can override all the default implementations including the canonical constructor (i.e. for data validation).

Java record Code Example



```
import java.util.ArrayList;
6 usages
record PersonRecord(String name, int age) { }
no usages
public class TestRecord {

    no usages
    public static void main(String[] args) {
        ArrayList<PersonRecord> folks = new ArrayList<>();
        folks.add(new PersonRecord( name: "John",  age: 18));
        folks.add(new PersonRecord( name: "Doe",  age: 12));
        folks.add(new PersonRecord( name: "Rick",  age: 19));
        folks.add(new PersonRecord( name: "Alan",  age: 25));
        folks.add(new PersonRecord( name: "Jack",  age: 22));

        folks.forEach(System.out::println);
    }
}
```

```
PersonRecord[name=John, age=18]
PersonRecord[name=Doe, age=12]
PersonRecord[name=Rick, age=19]
PersonRecord[name=Alan, age=25]
PersonRecord[name=Jack, age=22]

Process finished with exit code 0
```