Records in Java 14

```
record Point(int x, int y) { }
```

Records in Java 16

https://openjdk.java.net/jeps/395

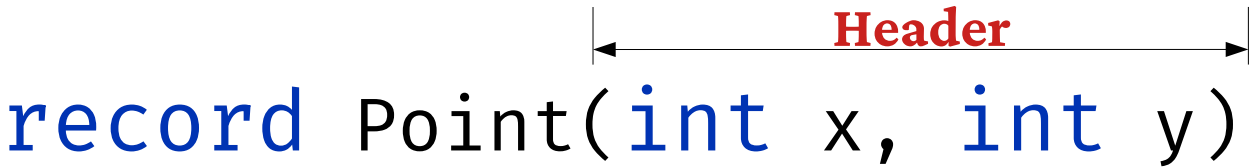
common complaint

"Java is too verbose" or has "too much ceremony"

a lot of low-value, repetitive, error-prone code

```
class Point {
    private final int x;
    private final int y;

    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }


    int x() { return x; }
    int y() { return y; }


    public boolean equals(Object o) {
        if (!(o instanceof Point)) return false;
        Point other = (Point) o;
        return other.x == x && other.y == y;
    }

    public int hashCode() {
        return Objects.hash(x, y);
    }
    public String toString() {
        return String.

                format("Point[x=%d, y=%d]"

                , x, y);
    }
}
```

Records : immutable data



The header of a record class describes its state

Records : immutable data

1) Super class is **`java.lang.Record`**

2) Two **members** (private final) and two public **accessors**

3) A canonical **constructor**

4) `equals` and **`hashCode`** methods

5) A **`toString`** method

Records : Behavior

1) Creation using a new expression
2) Can be declared top level or nested, and can be generic
3) Can declare static methods, fields, and initializers
4) Can declare instance methods
5) Can implement interfaces

Records : Behavior

6) Can declare nested types
7) A record class, and the components in its header, may be decorated with annotations
8) Instances of record classes can be serialized and deserialized

Records : Rules

1) Does not support `extends`
2) A record class is implicitly final
3) The fields are final
4) You cannot explicitly declare instance fields
5) Any implementation of methods should preserve the definition of the record class
6) A record class cannot declare native methods