Records

Learning objectives

- Key concepts about Records
- Comparing JavaBeans and Records
- Creating and using Records

Records

- Records was introduced in Java 14
- Records simplify the creation of classes used for storing data
- Records offer a concise and convenient way to define immutable data classes with a minimum amount of boilerplate code
- Records generate several common methods
- Records can be used to replace JavaBeans for immutable data classes

Records - Key concepts

- Concise syntax - reducing the need for explicit code for variables, constructors, getters and setters, toString, equals and hashCode
- Variables are automatically declared and are automatically private and final
- A constructor with all the variables of the Record is automatically generated
- Accessor methods are automatically generated (instead of getters)
- toString, equals and hashCode are all generated automatically
- Records are immutable - their state cannot be changed once they are created

Inheritance and custom methods

- Records implicitly extend java.lang.Record which is a superclass for all records, and cannot extend any other class
- Records can have custom methods for additional behavior
- The generated methods are always present

Typical JavaBean

```
public class Dog {
    private String name;
    private int age;

    public Dog() {
    }

    public Dog(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Dog{" +
                "name='" + name + '\'' +
                ", age=" + age +
                '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (!(o instanceof Dog dog)) return
false;
        return age == dog.age &&
Objects.equals(name, dog.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, age);
    }
}
```

Typical Record

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

Demo 1 - Records

- Creating a Record
- Using a Record

Exercise 1 - Records

- Create Records that represents objects we have used before as JavaBeans, like Book.
- In the main method, use the Record to create an object.
- Try to use the built-in methods in the Record object.

Learning objectives

- Key concepts about Records
- Comparing JavaBeans and Records
- Creating and using Records