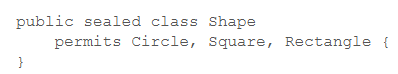
New features in JDK 17

# 13.1 Sealed class

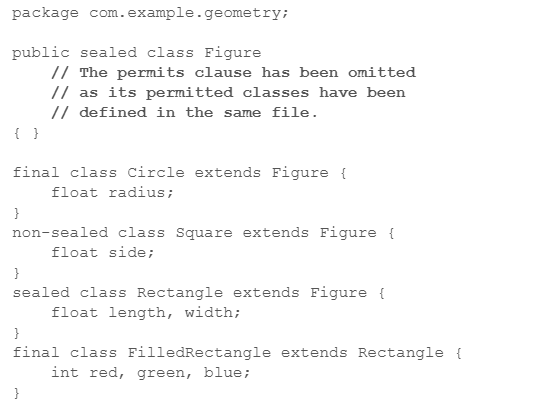
-Sealed classes and interfaces: restrict which other classes or interfaces may extend or implement them.

-Declare:



+The permitted subclasses are in the **same module** or in the **same package.**

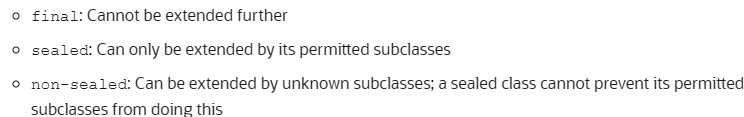
+You can define permitted subclasses in the **same file**, then you **can omit permits**:



-**Permitted subclasses**:

+Must **extend** the **sealed class**.

+Must be one of:



+Must be in the **same module** or **same package**.

-**Sealed interface**: specifies classes and interfaces that can extend and implement

-**Record** classes as **permitted subclasses** of a **sealed interface**: record is **final**

-**java.lang.Class methods**:

permittedSubClasses()

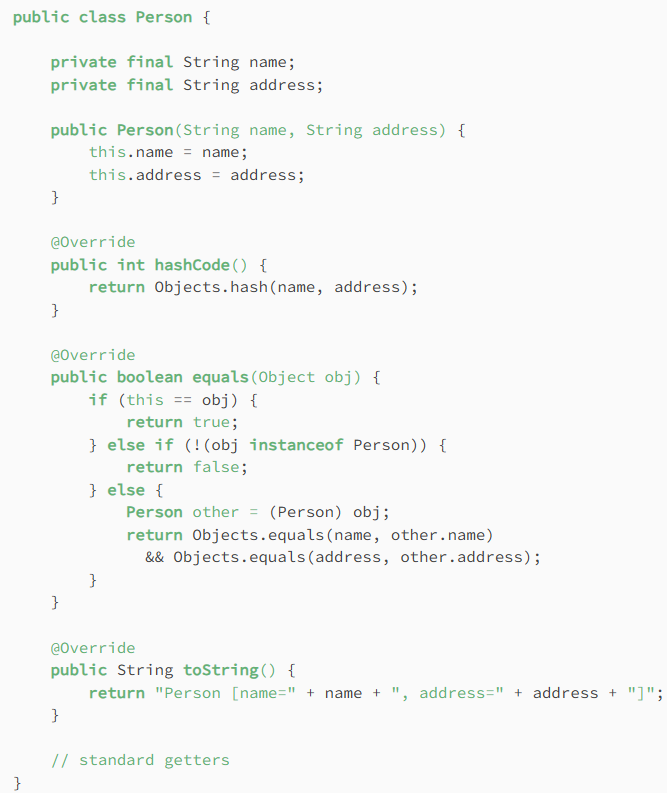
isSealed()

# 13.2 Record

<https://www.baeldung.com/java-record-keyword>

13.2.1 Purpose

-We write classes to hold data (database result, query result, information from a service), this data is **immutable**.



+There’s lots of boilerplate code.

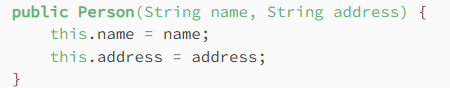
+Obscure the purpose of the class

13.2.2 The basics

-JDK 14: Records are immutable data classes that requires only the type and name of fields. final equals(), final hashCode(), final toString(), private, final fields and constructor are generated by Java compiler.



-The equivalent constructor:



+Instantiate objects from record:



-The equivalent **getters** (Note: name of getter = name of instance fields, example: name()), equals (true if objects of same type and values match), hashCode (return same value for 2 objects if all field values match), toString(name of record+field names+field values)

-Record **can’t be extended** and **can’t have extend clause**.

-Record can implement Serializable interface

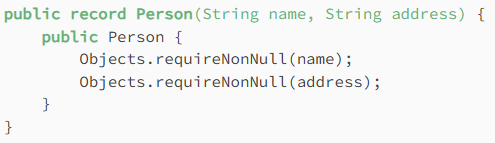
-Record may have at **most 1 varagrs** component.

* + 1. Constructors

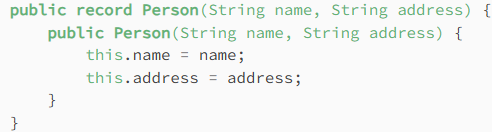
-**Canonical constructor**:

+We can **customize** constructor for **validation**: (**compact form**)

+Example: fields aren’t null

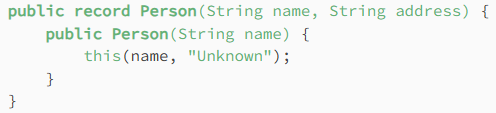


+Creating a constructor with **same arguments** as the generated public constructor is valid, but each field is **manually initialized**: **regular form canonical constructor**



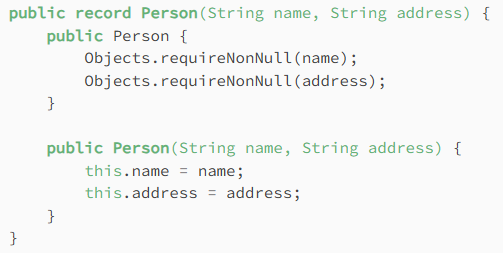
Or public Journal {id=id+1}

-**Non-canonical constructor**: Create **new constructors** with different arguments by supplying either a **canonical constructor** or **another constructor**:



-**Error** when declare **2 canonical constructors**:

+Example: compact form + regular form



13.2.4 Static variables and methods

-We can use **static variables** and **static (and instance, not setter)** **methods** in record

