Sealed Classes and Interfaces

Jesper de Jong

Software Architect

@jesperdj | www.jesperdj.com

Sealed Classes and Interfaces

A way to get detailed control over class hierarchies, and specify what classes or interfaces can extend or implement a class or interface.

Extensibility of Classes and Interfaces

Make a class final

No subclasses anywhere

Make the constructor package-private

Subclasses in the same package only

Interfaces

No way to limit implementing or extending

Why?

The rules - How

The Rules of Sealed Classes and Interfaces

Creating a Sealed Class

```
class Animal {}

class Pet extends Animal {}
```

Creating a Sealed Class

```
class Animal {}

sealed class Pet extends Animal {}
```

Creating a Sealed Class

```
class Animal {}

sealed class Pet extends Animal permits Dog, Cat, Fish {}

class Dog extends Pet {}

class Cat extends Pet {}

class Fish extends Pet {}
```

Creating a Sealed Class

```
class Animal {}

sealed class Pet extends Animal permits Dog, Cat, Fish {}

class Dog extends Pet {}

class Cat extends Pet {}

class Fish extends Pet {}
```

Creating a Sealed Class

```
class Animal {}

sealed class Pet extends Animal permits Dog, Cat, Fish {}

final class Dog extends Pet {}

sealed class Cat extends Pet {}

non-sealed class Fish extends Pet {}
```

Creating a Sealed Class

```
class Animal {}

sealed class Pet extends Animal permits Dog, Cat, Fish {}

final class Dog extends Pet {}

sealed class Cat extends Pet {}

non-sealed class Fish extends Pet {}
```

Creating a Sealed Class

```
class Animal {}

sealed class Pet extends Animal permits Dog, Cat, Fish {}

final class Dog extends Pet {}

sealed class Cat extends Pet permits MaineCoon, Siamese {}
final class MaineCoon extends Cat {}
final class Siamese extends Cat {}

non-sealed class Fish extends Pet {}
```

Creating a Sealed Class

```
class Animal {}

sealed class Pet extends Animal permits Dog, Cat, Fish {}

final class Dog extends Pet {}

sealed class Cat extends Pet permits MaineCoon, Siamese {}
final class MaineCoon extends Cat {}
final class Siamese extends Cat {}

non-sealed class Fish extends Pet {}
```

Creating a Sealed Class

```
class Animal {}

sealed class Pet extends Animal permits Dog, Cat, Fish {}

final class Dog extends Pet {}

sealed class Cat extends Pet permits MaineCoon, Siamese {}
final class MaineCoon extends Cat {}
final class Siamese extends Cat {}

non-sealed class Fish extends Pet {}
class Goldfish extends Fish {}
```

Creating a Sealed Class

```
class Animal {}

sealed class Pet extends Animal permits Dog, Cat, Fish {}

final class Dog extends Pet {}

sealed class Cat extends Pet permits MaineCoon, Siamese {}
final class MaineCoon extends Cat {}
final class Siamese extends Cat {}

non-sealed class Fish extends Pet {}
class Goldfish extends Fish {}
```

Creating a Sealed Class

Same source file

```
class Animal {}

sealed class Pet extends Animal permits Dog, Cat, Fish {}

final class Dog extends Pet {}

sealed class Cat extends Pet permits MaineCoon, Siamese {}
final class MaineCoon extends Cat {}
final class Siamese extends Cat {}

non-sealed class Fish extends Pet {}
class Goldfish extends Fish {}
```

Creating a Sealed Class

```
class Animal {}

sealed class Pet extends Animal permits Dog, Cat, Fish {}

final class Dog extends Pet {}

sealed class Cat extends Pet permits MaineCoon, Siamese {}
final class MaineCoon extends Cat {}
final class Siamese extends Cat {}

non-sealed class Fish extends Pet {}
class Goldfish extends Fish {}
```

- **In the same named module**
- **In the same package (unnamed module)**

Creating a Sealed Interface

```
interface Animal {}

sealed interface Pet extends Animal permits Dog, Cat, Fish {}

final class Dog implements Pet {}

sealed interface Cat extends Pet permits MaineCoon, Siamese {}
final class MaineCoon implements Cat {}
final class Siamese implements Cat {}

non-sealed interface Fish extends Pet {}
class Goldfish implements Fish {}
```

Creating a Sealed Interface

```
interface Animal {}

sealed interface Pet extends Animal permits Dog, Cat, Fish {}

final class Dog implements Pet {}

sealed interface Cat extends Pet permits MaineCoon, Siamese {}
final class MaineCoon implements Cat {}
final class Siamese implements Cat {}

non-sealed interface Fish extends Pet {}
class Goldfish implements Fish {}
```

Creating a Sealed Interface

```
interface Animal {}

sealed interface Pet extends Animal permits Dog, Cat, Fish {}

final class Dog implements Pet {}

sealed interface Cat extends Pet permits MaineCoon, Siamese {}
final class MaineCoon implements Cat {}
final class Siamese implements Cat {}

non-sealed interface Fish extends Pet {}
class Goldfish implements Fish {}
```

Creating a Sealed Interface

```
interface Animal {}

sealed interface Pet extends Animal permits Dog, Cat, Fish {}

final class Dog implements Pet {}

sealed interface Cat extends Pet permits MaineCoon, Siamese {}
final class MaineCoon implements Cat {}
final class Siamese implements Cat {}

non-sealed interface Fish extends Pet {}
class Goldfish implements Fish {}
```

- **In the same named module**
- **In the same package (unnamed module)**

Sealed Classes, Interfaces and Records

Records cannot extend sealed classes

Records cannot be sealed

Records can implement sealed interfaces

Algebraic data types

Sealed Classes and Interfaces in Practice

Sealed Classes and Interfaces in Practice

Keep classes and interfaces open for extension

Allow mocks for unit testing

Special use cases

Model the grammar of
a language or protocol

Interface for immutable collections

Algebraic data types

Algebraic Data Types with Sealed Interfaces and Records

Modeling Fixed Sets of Values or Types

An enum defines a type with a fixed set of alternative values

With a sealed interface and records you can define a fixed set of alternative types

Summary

Sealed Classes and Interfaces

- Control which classes or interfaces extend a class or interface

The rules

- permits
- Direct subclasses or subinterfaces
- final, sealed, non-sealed
- Same named module or same package

Practical use cases

Algebraic data types

Up Next:

Pattern Matching