JAVA PROGRAMMING BASICS

Module 2: Java Object-oriented Programming

Training program

1. Classes and Instances
2. The Methods
3. The Constructors
4. Static Elements
5. Initialization sections
6. Package
7. Inheritance and Polymorphism
8. Abstract classes and Interfaces
9. String processing
10. Wrapper classes for primitive types
11. Exceptions and Assertions
12. Nested classes
13. **Enums**
14. Generics
15. Collections
16. Method overload resolution
17. Multithreads
18. Core Java classes
19. Object Oriented Design
20. Functional Programming

Module contents

- The enums
  - The Enums and operations with it
  - Enum as type. Enums methods overriding
  - Enums restrictions

Module contents

- The enums
  - The Enums and operations with it
  - Enum as type. Enums methods overriding
  - Enums restrictions

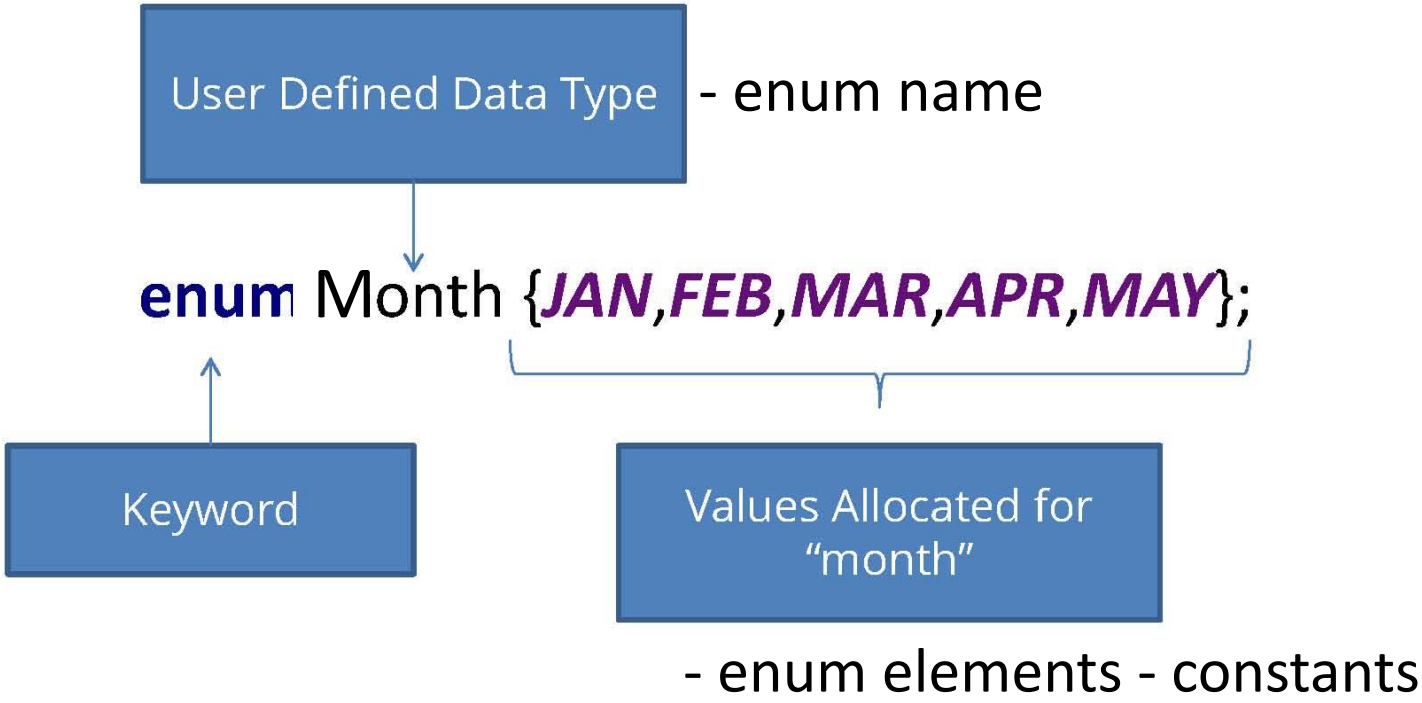
The Enums and operations with it 1/7

- In computer programming, an enumerated type (also called enumeration or enum) is a data type consisting of a set of named values called elements, members or enumerators of the type.
- An ***enum*** *type* is a special data type that enables for a variable to be a set of predefined constants.
- It is allowed to compare the elements of enumerations (the order of the elements is taken into account), as well as their use in the case clauses of the switch statement
- Enumeration makes programs more readable and type-safe.

since JDK 1.5

The Enums and operations with it 2/7

- Enum example



The Enums and operations with it 3/7

- Define an enum type by using the **enum** keyword

```
1. public enum MyDirection {
2.         NORTH,
3.         EAST,
4.         SOUTH,
5.         WEST
6. }
```

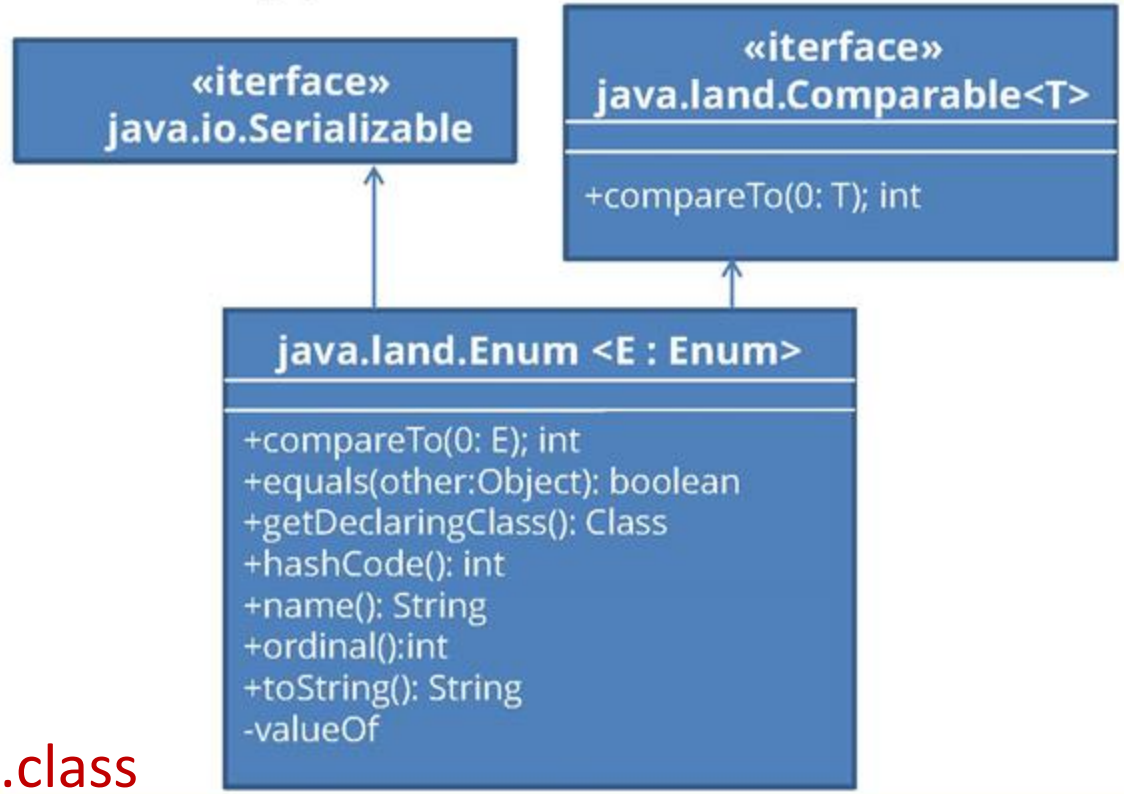
The enumeration is a custom class inherited the java.lang.Enum class

Module contents

- The enums
  - The Enums and operations with it
  - Enum as type. Enums methods overriding
  - Enums restrictions

Enum as type. Enums methods overriding

- Java enums extend the java.lang.Enum class implicitly, so your enum types cannot extend another class.



See javap -p MyDirection.class

Enum as type. Enums methods overriding

Every enum has the **added by compiler methods**:

- **public static** EnumName[] **values()** - returns an array EnumName[] $VALUES of all enum constants
- **public static** EnumName **valueOf(String name)** - parse enum constant

Every enum has the **methods inherited from**
public abstract class Enum<E **extends** Enum<E>>
implements Constable, Comparable<E>, Serializable

public final String **name()** - returns the name of this enum constant

- **public final int ordinal()** - returns an ordinal value of this enum constant
- **public static** <T **extends** Enum<T>> T valueOf(Class<T> enumClass, String name) - Returns the enum constant of the specified enum class with the specified enum constant name

See MyDirectionTest

Enum as type. Enums methods overriding

Because the enum is a class You can:

- add custom fields to it;
- add overloaded constructors to it;
- add custom methods to it;
- override standard methods.

See Shape

See Fruit

See DocumentStatus

Module contents

- The enums
  - The Enums and operations with it
  - Enum as type. Enums methods overriding
  - Enums restrictions

Enums restrictions 1/2

- You can NEVER invoke an enum constructor directly

```
public static void main(String[] arg) {
    MyDirection myDir1 = new MyDirection();
}
```

Enums restrictions 2/2

- The next Enums methods declared as **final** so you cannot override it:
- **name()**
- **ordinal()**
- **equals(...)**
- **hashCode()**
- **compareTo(...)**
- **clone()**

toString() - is not final