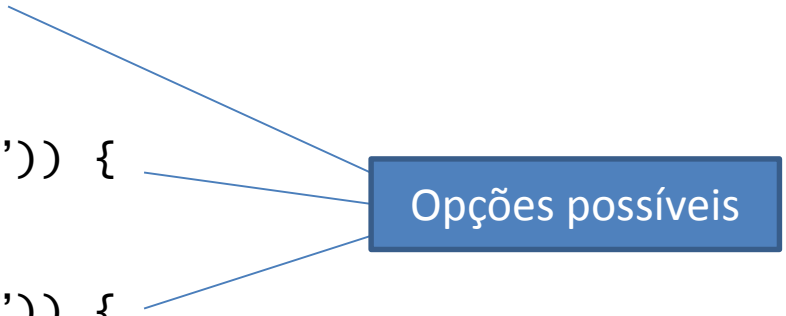
Enumerados

Enumerado

- Tipo com conjunto fixo e finito de valores
  - Exemplos: dias da semana, direcções, estado civil
- Podem ter atributos e construtores
- Têm operações associadas
- Melhores que inteiros ou cadeias de caracteres para representar pequenos conjuntos

Exemplo (opções de menu)

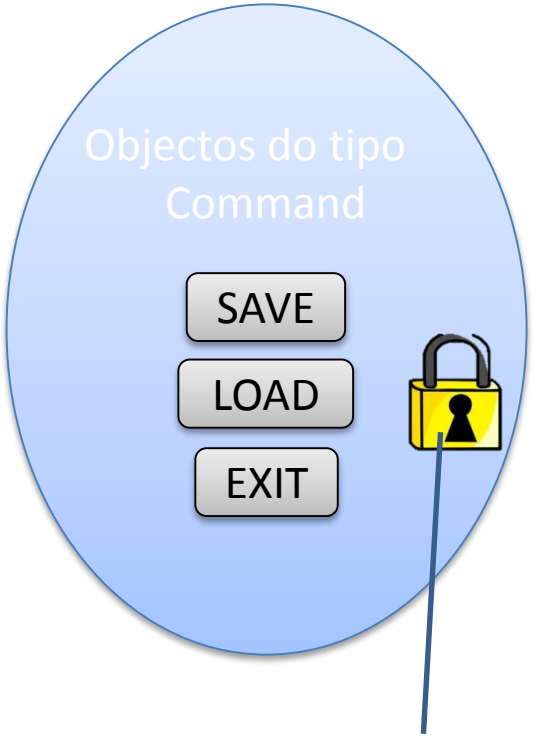
```
...
Scanner scanner = new Scanner(System.in);
System.out.println("Introduza um comando:");
String command = scanner.nextLine();
if(command.equals("SAVE")) {
  // gravar...
}
else if(command.equals("LOAD")) {
  // carregar...
}
else if(command.equals("EXIT")) {
  // sair...
}
...
```



Exemplo (opções de menu)

```
public enum Command {
    SAVE, LOAD, EXIT;
}

...
Scanner scanner = new Scanner(System.in);
System.out.println("Introduza uma comando:");
String line = scanner.nextLine();
Command command = Command.valueOf(line);
if(command == Command.SAVE) {
  // gravar...
}
else if(command == Command.LOAD) {
  // carregar...
}
else if(command == Command.EXIT) {
  // sair...
}
...
```



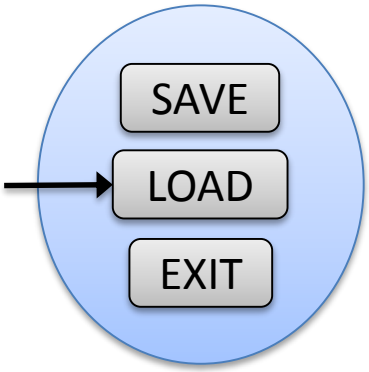
"Conjunto fixo"... não é possível remover ou adicionar objectos em tempo de execução

Operação `valueOf(String)`

- Disponível em todos os tipos enumerados
- Obtem o elemento de um enumerado dado o seu nome (objecto String)

```
public enum Command {
    SAVE, LOAD, EXIT;
}
```

```
Command c = Command.valueOf("LOAD");
```



Instrução de selecção `switch`

- Alternativa ao if-else
  - Adequada quando as diferentes alternativas de execução são determinadas pelo valor de determinada variável
    - A variável pode ter um dos tipos primitivos numéricos para representar inteiros (byte, short, int) , char, ou ser de um tipo enumerado

Exemplo (opções de menu / switch)

```
public enum Command {
   SAVE, LOAD, EXIT;
}

...
Scanner scanner = new Scanner(System.in);
System.out.println("Introduza uma comando:");
String line = scanner.nextLine();
Command command = Command.valueOf(line);
switch(command) {
  case SAVE:
     // gravar...
     break;
  case LOAD:
     // carregar...
     break;
  case EXIT:
     // sair...
     break;
}
```

Exemplo (direcção)

```
public class Direction {
   private String name;

   public Direction(String name) {
      this.name = name;
   }

   public String getName() {
      return name;
   }
}
```

Fará sentido existirem outras instâncias para além destas?

```
Direction north = new Direction("North");
Direction south = new Direction("South");
Direction east = new Direction("East");
Direction west = new Direction("West");
```

Exemplo (direcção)

```
public enum Direction {
    NORTH, SOUTH, EAST, WEST;

    public String prettyName() {
        return name().charAt(0) +
               name().substring(1).toLowerCase();
    }
}
```

```
String s1 = Direction.NORTH.name();
System.out.println(s1);
String s2 = Direction.SOUTH.prettyName();
System.out.println(s2);
```

```
> NORTH
> South
```

Operação `name()`

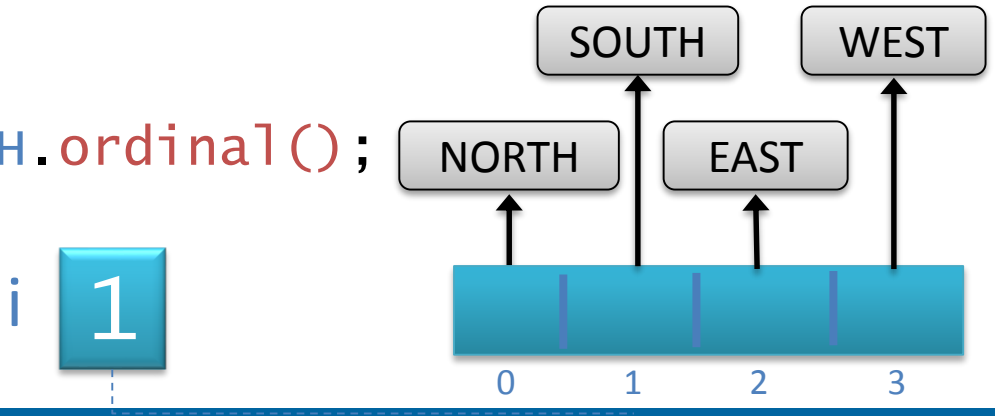
- Disponível em todos os tipos enumerados
- Devolve um objecto String com o identificador do elemento do enumerado

```
String s = Direction.WEST.name();
```

Operação `ordinal()`

- Disponível em todos os tipos enumerados
- Devolve o índice do elemento do enumerado de acordo com a ordem de declaração

```
public enum Direction {
    NORTH, SOUTH, EAST, WEST;
    ...
}
int i = Direction.SOUTH.ordinal();
```



Exemplo (dias da semana)

```
public class WeekDay {
    private String name;
    private int number;

    public WeekDay(String name, int number) {
        this.name = name;
        this.number = number;
    }

    public String getName() {
        return name;
    }

    public int getNumber() {
        return number;
    }
}
```

Fará sentido existirem outras instâncias para além destas?

```
WeekDay mon = new WeekDay("Monday", 1);
WeekDay tue = new WeekDay("Tuesday", 2);
WeekDay wed = new WeekDay("Wednesday", 3);
WeekDay thu = new WeekDay("Thursday", 4);
WeekDay fri = new WeekDay("Friday", 5);
WeekDay sat = new WeekDay("Saturday", 6);
WeekDay sun = new WeekDay("Sunday", 7);
```

Exemplo (dias da semana)

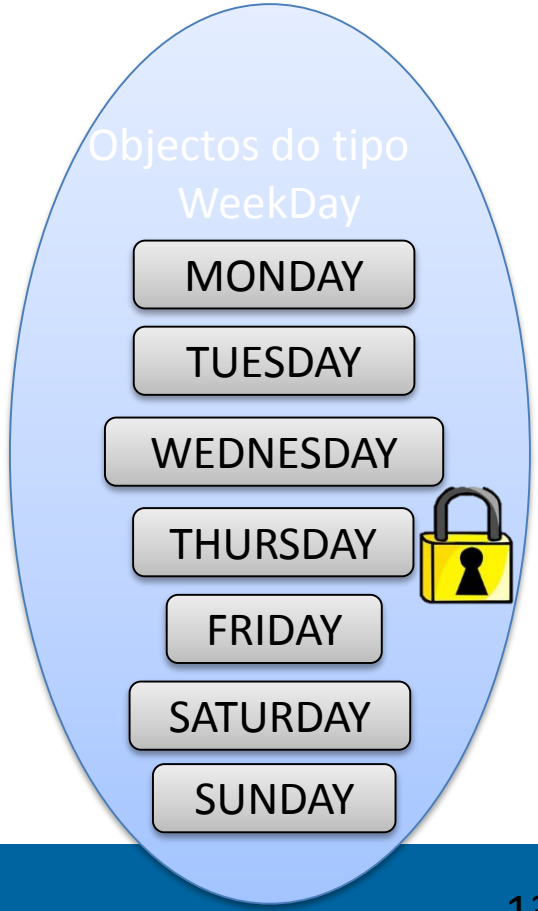
```
public enum WeekDay{
    MONDAY(1), TUESDAY(2), WEDNESDAY(3), THURSDAY(4), FRIDAY(5),
    SATURDAY(6), SUNDAY(7);

    private int number;

    private WeekDay(int number) {
        this.number = number;
    }

    public int getNumber() {
        return number;
    }
}
```

Obrigatoriamente privado… (pois as instâncias estão fixas à partida, não podem ser criadas mais)



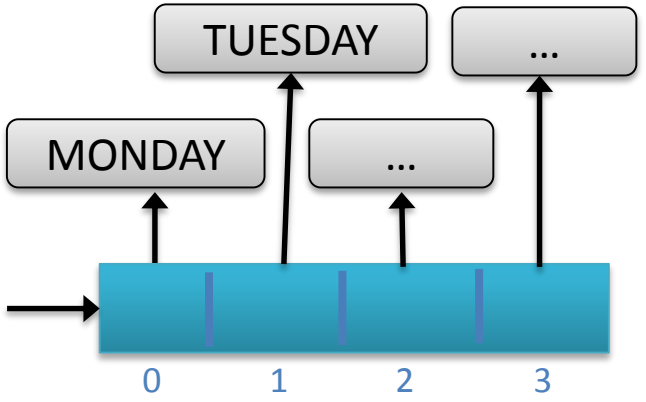
Operação `values()`

- Disponível em todos os tipos enumerados
- Devolve um vector com todos os elementos do enumerado (pela ordem que são declarados)

```
WeekDay[] days = WeekDay.values();

for(int i = 0; i < days.length; i++) {
   WeekDay day = days[i];
   String name = day.name();
   System.out.println(name);
}
```

```
> MONDAY
> TUESDAY
> WEDNESDAY
> THURSDAY
> FRIDAY
> SATURDAY
> SUNDAY
```



Referências

- Y. Daniel Liang, "Introduction to Java Programming" 7th Ed. Prentice-Hall, 2010.
- http://download.oracle.com/javase/tutorial/java/javaOO/enum.html

Sumário

- Enumerados