



Aula 12

Enumerados e Exceções

Iniciativa Conjunta:



Com o apoio de:





Enumerados

- Tipo com conjunto fixo e finito de valores
 - Exemplos: dias da semana, direções, 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...
}
```

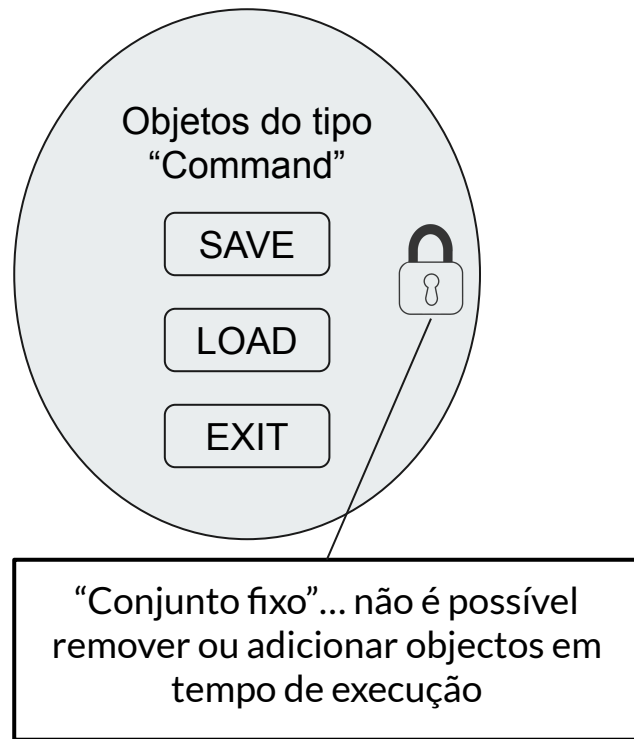
..



Opções
possíveis

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...  
}  
...
```





Exercício A

- Como ficaria o exemplo anterior (opções do menu) se fosse utilizada a estrutura de seleção 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);
```

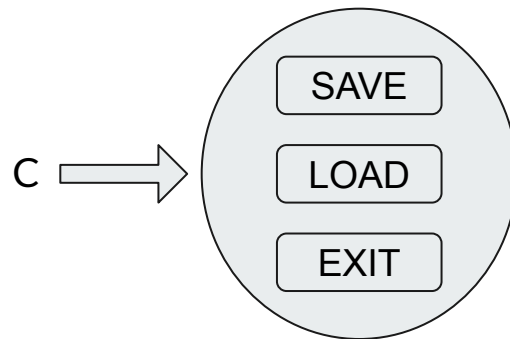
```
switch(command) {  
    case SAVE:  
        // gravar  
        ...  
        break;  
    case LOAD:  
        // carregar  
        ...  
        break;  
    case EXIT:  
        // sair  
        ...  
        break;  
}
```

Operação valueOf(String)

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

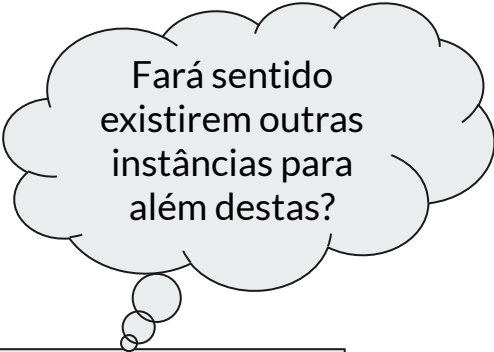
```
public enum Command {  
    SAVE, LOAD, EXIT;  
}
```

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



Exemplo - Direções

```
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?

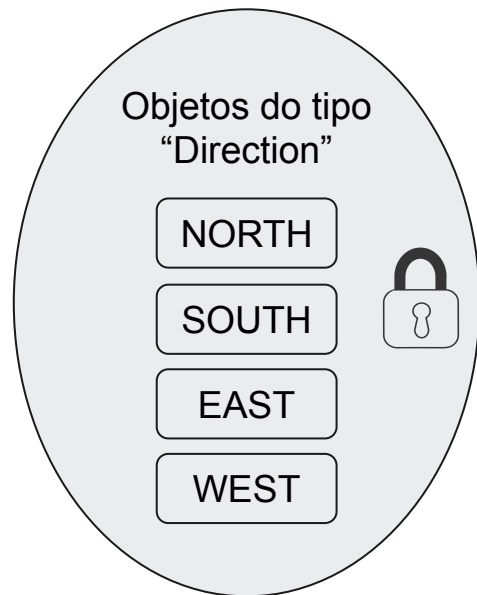
```
Direction north = new Direction("North");  
Direction south = new Direction("South");  
Direction east = new Direction("East");  
Direction west = new Direction("West");
```

Exemplo - Direções

```
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**

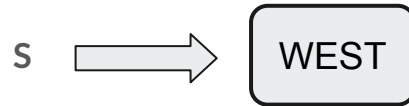




Operação name()

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

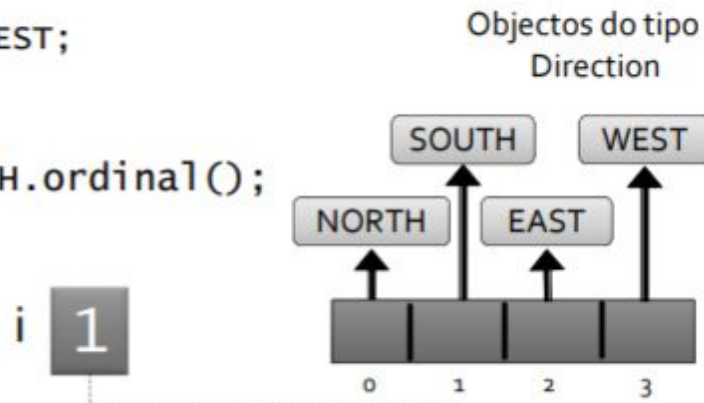
```
String s = Direction.WEST.name();
```



Operação ordinal()

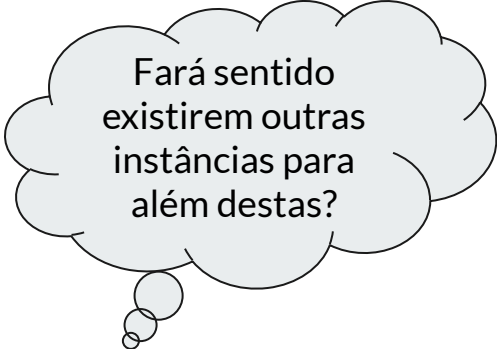
- Disponível em todos os tipos enumerados
- Devolve o índice do elemento do enumerado de acordo com a ordem de declaração

```
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)

Objetos do tipo
"Direction"

MONDAY

TUESDAY

WEDNESDAY

THURSDAY

FRIDAY

SATURDAY

SUNDAY



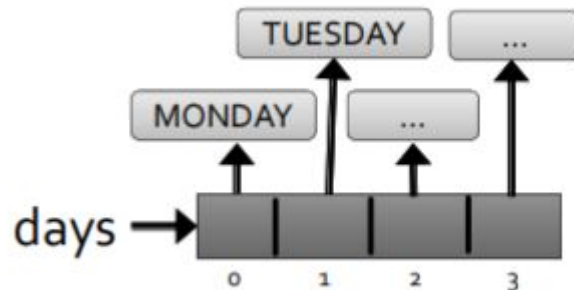
Operação values()

- 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





Exceções

O lançamento de exceções pode ser utilizado como um mecanismo para interromper a execução normal de um método, caso o objeto tenha sido utilizado de forma incorreta

- Invocação de uma operação com argumentos inválidos.
- Sequência de invocações inválida.

As exceções elas próprias **são objetos** (com atributos, operações, e construtores).



Tipos de Exceções

- Existem imensas exceções em Java
- Tipos relacionados com a utilização incorreta de objetos:
 - **IllegalArgumentException**: adequada quando um argumento inválido é utilizado na invocação de uma operação;
 - **NullPointerException**: adequada quando é passada uma referência null não permitida como argumento;
 - **IllegalStateException**: adequado quando é invocada uma operação não permitida dado o estado atual do objeto.



Lançamento de Exceções

IllegalArgumentException

```
class Point {  
    final int x;  
    final int y;  
    Point(int x, int y) {  
        if(x < 0 || y < 0){  
            throw new IllegalArgumentException("Valores não negativos!");  
        }  
        this.x = x;  
        this.y = y;  
    }  
    ...  
}
```




Lançamento de Exceções

NullPointerException

```
class ImageUtils {  
    static void invert(ColorImage img) {  
        if(img == null){  
            throw new NullPointerException("O argumento não pode ser null!");  
        }  
        ...  
    }  
    ...  
}
```



Lançamento de Exceções

IllegalStateException

```
class IntSet {  
    ...  
    boolean isFull() {  
        ...  
    }  
    void add(int element) {  
        if(isFull())  
            throw new IllegalStateException("O conjunto está cheio!");  
        ...  
    }  
}
```



Exercício B

- Escreva um enumerado que represente as quatro operações matemáticas: somar, subtrair, multiplicar e dividir.
- Desenvolva um método que realize todas as operações do enumerado.
 - Realize todas as exceções que considerar necessárias.