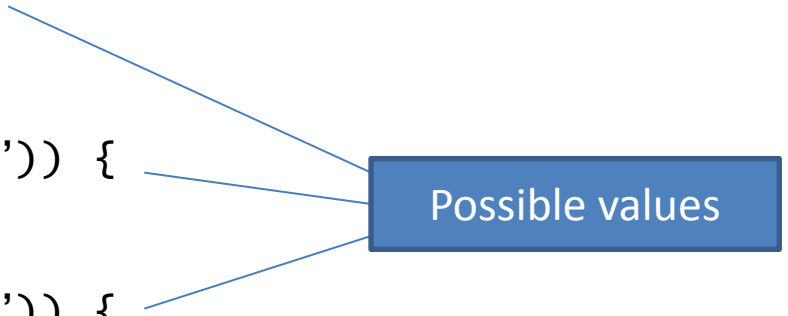
Enumerated

Enumerated

- Type with fixed set of values
  - Example: days of week, directions, marital status
- Can have attributes and constructors (in Java)
- Has operations
- Better than int, char or String to represent small sets

Example (menu options)

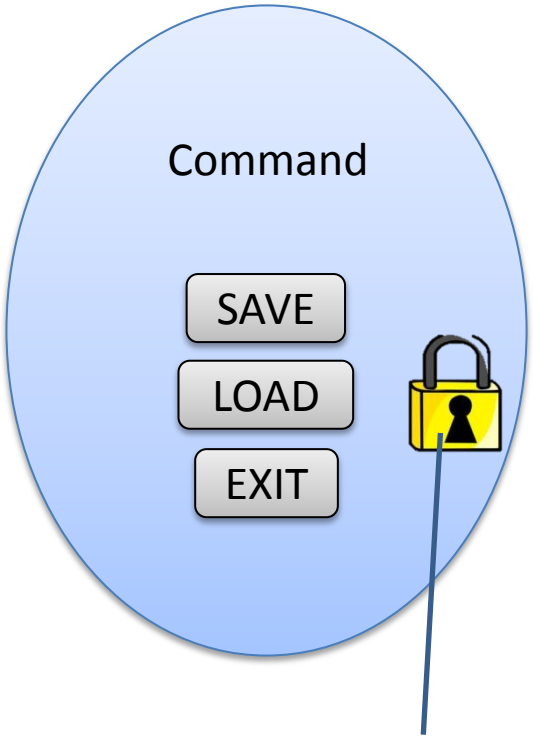
```
...
Scanner scanner = new Scanner(System.in);
System.out.println("Command:");
String command = scanner.nextLine();
if(command.equals("SAVE")) {
  // save ...
}
else if(command.equals("LOAD")) {
  // load ...
}
else if(command.equals("EXIT")) {
  // exit ...
}
...
```



Example (menu options)

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

...
Scanner scanner = new Scanner(System.in);
System.out.println("Command:");
String line = scanner.nextLine();
Command command = Command.valueOf(line);
if(command == Command.SAVE) {
  // save ...
}
else if(command == Command.LOAD) {
  // load ...
}
else if(command == Command.EXIT) {
  // exit ...
}
...
```



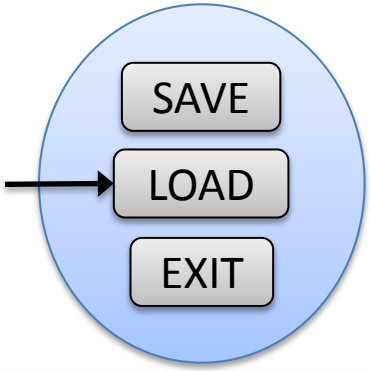
Not possible to remove or add objects when executing

Operation `valueOf(String)`

- Available to all enumerated
- Returns the enum value given its name (a String object)

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

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



The `switch` instruction

- Alternative to if-else: Adequate when alternatives are related to different values of an enumerated, (also possible for byte, short, int or char)

Example (menu / switch)

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

...
Scanner scanner = new Scanner(System.in);
System.out.println("Command:");
String line = scanner.nextLine();
Command command = Command.valueOf(line);
switch(command) {
  case SAVE:
     // save ...
     break;
  case LOAD:
     // load ...
     break;
  case EXIT:
     // exit ...
     break;
}
```

Example (direction)

```
public class Direction {
   private String name;

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

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

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

Example (direction)

```
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
```

Operation name()

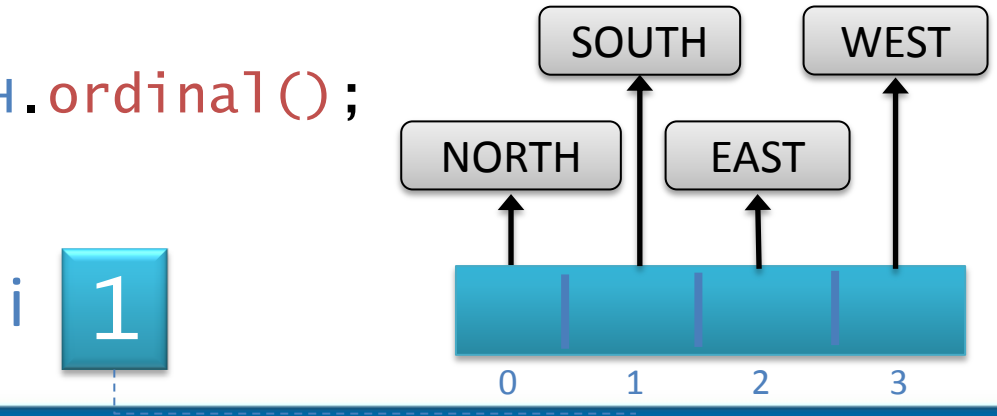
- Available to all enumerates
- Returns a String

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

Operation `ordinal()`

- Available to all enumerates
- Return the value's index in the declaration

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



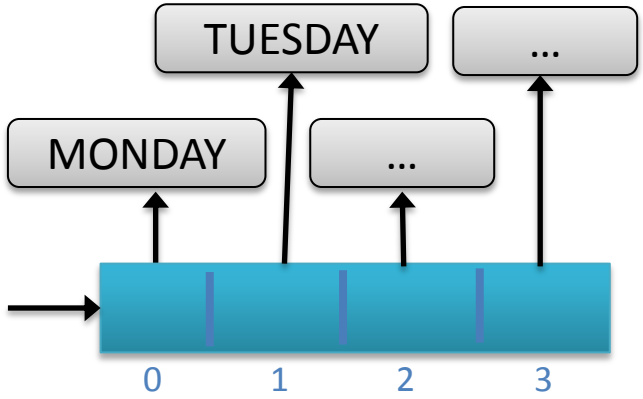
Operation values()

- Available to all enumerates
- Returns vector with all values (in declaration order)

```
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
```



More information/ References

- http://download.oracle.com/javase/tutorial/java/javaOO/enum.html

Summary

- Enumerated