Enums

What is an Enum?

- Enum (enumeration) is a fixed set of constants
- Enum provides type-safe checking
  - it's impossible to create an invalid enum value
- common examples:
  - seasons, compass directions, days of the week, deck of cards, etc.

```
// simple enums
public enum Compass {
   NORTH, SOUTH, EAST, WEST ;
}
```

optional for simple enums

```
Compass N = Compass.NORTH;
System.out.println(N);
   => NORTH
System.out.println(N == Compass.NORTH);
   => true
```

alternative: `Compass.valueOf("NORTH")`

```
NORTH : 0
SOUTH : 1
EAST : 2
WEST : 3
```

returns array of values

```
for (var direction : Compass.values()) {
   System.out.println(direction.ordinal() + " : " + direction.name());
}
```

ordinal number of a value

name of a value

```
// enums are often used in switch

Compass N = Compass.NORTH;

switch(N) {
   case NORTH -> System.out.println("You are headed North.");
   case SOUTH -> System.out.println("You are headed South.");
   default -> System.out.println("Get back!");
}
```

```
// examples of wrong syntax

case Compass.NORTH ->

case 1 ->
```

```
// enums can have constructors and instance methods
enum Compass {
   NORTH("Move Up"), SOUTH("Move Down"), EAST("Move Right"), WEST("Move Left");
   private final String instruction;
   private Compass(String instruction) {
      this.instruction = instruction;
   }
   public void printInstruction () {
      System.out.println(instruction);
   }
}

Compass.NORTH.printInstruction();
```

constructor calls (but without new keyword)

required

enum constructor

(implicitly private)

enum method

1. the constructors are called for each enum (only once)
2. the instruction "Move Up" is printed by `printInstruction()` method

```
Move Up!
```

```
// enums can implement abstract methods
enum Compass {
  NORTH {
    public String getDirection() { return "Up"; }
  },
  SOUTH {
    public String getDirection() { return "Down"; }
  },
  EAST {
    public String getDirection() { return "Right"; }
  },
  WEST {
    public String getDirection() { return "Left"; }
  };
  public abstract String getDirection();
}
System.out.println(Compass.SOUTH.getDirection());
```

NOTE: enum values must be listed at the beginning !!

implementation

abstract method

Down

```
// methods can be overriden my certain enums only
enum Compass {
  NORTH {
    public String getDirection() { return "Up"; }
  },
  SOUTH {
    public String getDirection() { return "Down"; }
  },
  EAST, WEST;
  public String getDirection() { return "Sideways"; }
}

System.out.println(Compass.SOUTH.getDirection());
System.out.println(Compass.EAST.getDirection());
```

overriding `getDirection()` method

use default implementation

```
Down
Sideways
```

```
// enums cannot extend a class, but can implement an interface
interface Planet {
  String getPlanetName();
}
public enum Compass implements Planet {
  NORTH, SOUTH, EAST, WEST;
  public String getPlanetName() {
    return "Earth";
  }
}

System.out.println(Compass.NORTH.getPlanetName());
  => Earth
```