OPTIONAL

`null`

"I call it my billion-dollar mistake. It was the invention of the null reference in 1965."

Sir Tony Hoare

"My goal was to ensure that all use of references should be absolutely safe, with checking performed automatically by the compiler. But I couldn't resist the temptation to put in a null reference, simply because it was so easy to implement. "

```
String saluda (Persona persona) {
    return "Hola " + persona.getNombre();
}
```

```
saluda(null);
```

```
Exception in thread "main" java.lang.NullPointerException
  at Main.saluda(Main.java:28)
  at Main.main(Main.java:23)
```

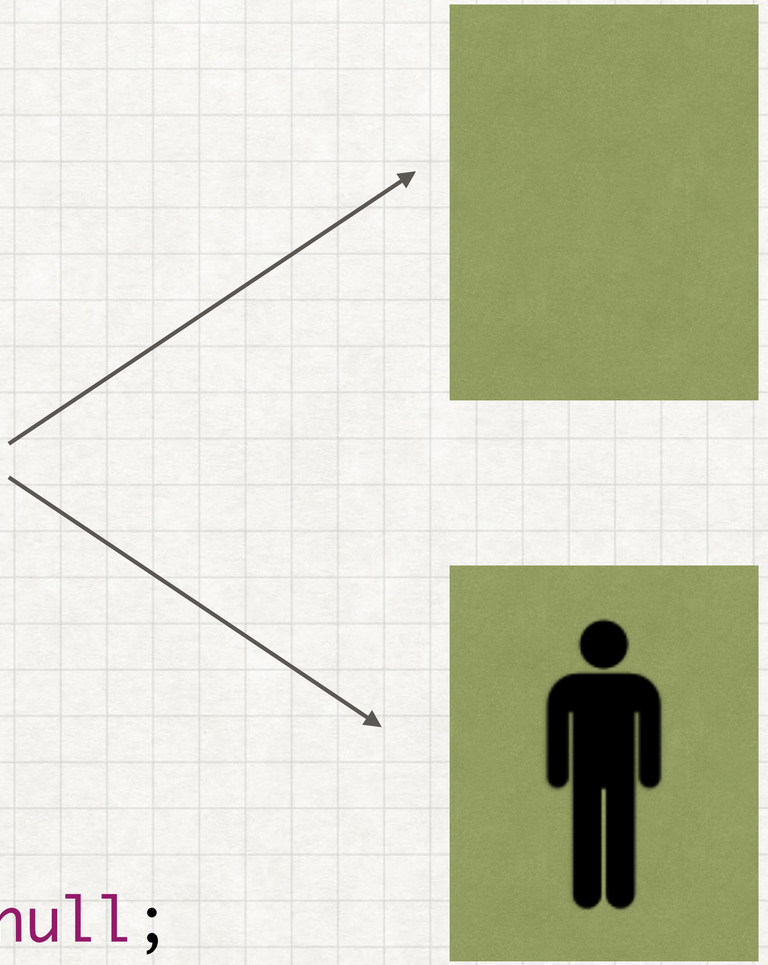
```
String saluda (Persona persona) {
    return "Hola " + persona.getNombre();
}
```

¿Es obligatorio pasar un objeto persona al método?

```
String saluda (Persona persona) {
    if (persona == null) {
        return "Estoy solo";
    }
    return "Hola " + persona.getNombre();
}
```

```
String saluda ( Optional<Persona> persona ) {
    ...
}
```

Optional<Persona> persona



persona = null;

Creación

```
Optional<Persona> persona = Optional.of(juan);

Optional<Persona> persona = Optional.empty();

Optional<Persona> persona = Optional.ofNullable(juan);
```

```
String saluda (Optional<Persona> persona) {

    if (persona.isPresent()) {
      return "Hola " + persona.get().getNombre();
    } else {
      return "Estoy solo";
    }
}
```

poco "idiomático"

map

```
Optional<String> saluda (Optional<Persona> persona) {
    return persona.map(it -> "Hola " + it.getNombre());
}
```

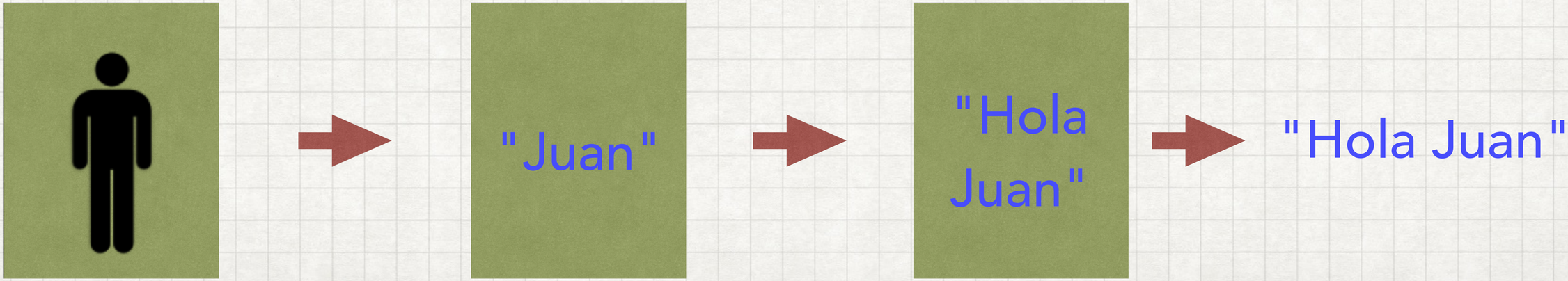
orElse

```
String saluda (Optional<Persona> persona) {
    return persona.map(it -> "Hola " + it.getNombre())
                  .orElse("Estoy solo");
}
```

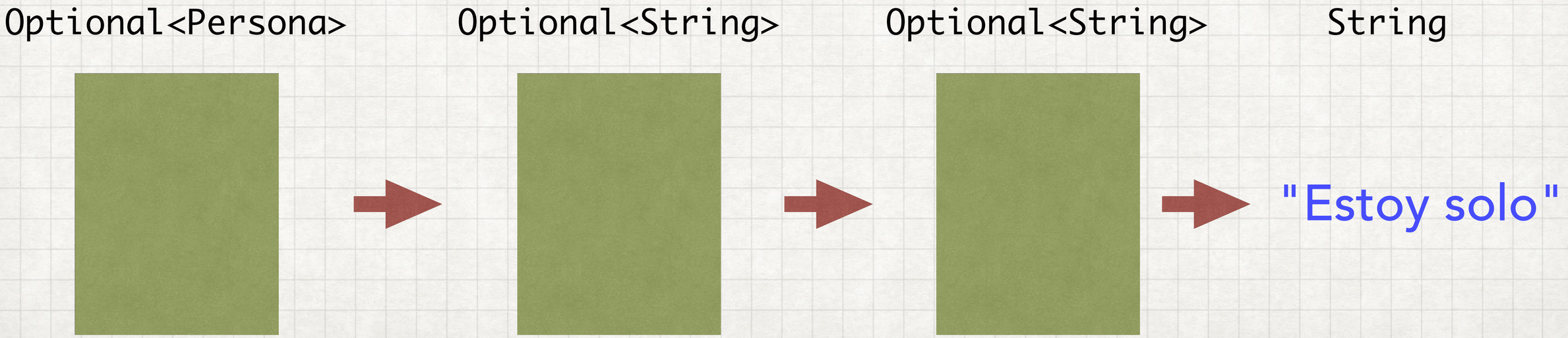
```
String saluda (Optional<Persona> persona) {

  return persona.map(Persona::getNombre)
                .map("Hola "::concat)
                .orElse("Estoy solo");
}
```

```
String saluda (Optional<Persona> persona) {

  return persona.map(Persona::getNombre)
                .map("Hola "::concat)
                .orElse("Estoy solo");
}
```

Optional<Persona> Optional<String> Optional<String> String



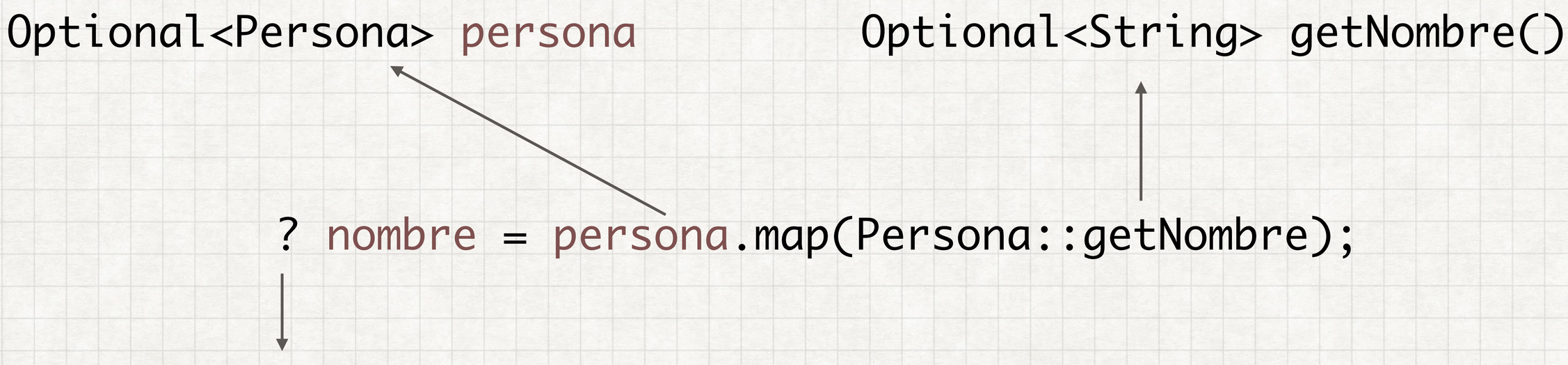
```
String saluda (Optional<Persona> persona) {

  return persona.map(Persona::getNombre)
                .map("Hola "::concat)
                .orElse("Estoy solo");
}
```



```
        static class Persona {
            private Optional<String> nombre;

            Optional<String> getNombre() {
              return nombre;
            }
        }

String saluda(Optional<Persona> persona) {

    ? nombre = persona.map(Persona::getNombre);
    ...
}
```

¿Cual es el tipo de nombre?

¿Cual es el tipo de nombre?



¿Cual es el tipo de nombre?

```
Optional<Persona> persona                    Optional<String> getNombre()
```

```
Optional<String> nombre = persona.???(Persona::getNombre);
```

```
public<U> Optional<U> flatMap(Function<? super T, Optional<U>> mapper)
```

```
Optional<String> nombre = persona.flatMap(Persona::getNombre);
```

```
static class Persona {
  private Optional<String> nombre;

  Optional<String> getNombre() {
    return nombre;
  }
}
```

```
String saluda (Optional<Persona> persona) {
  return persona.flatMap(Persona::getNombre)
                .map("Hola "::concat)
                .orElse("Estoy solo");
}
```

OPTIONAL