Contents

0.1 Clase Forms: Documentación e Explicación

0.1.1 Que é esta clase?

Esta clase `Forms` é como un **asistente** que axuda aos usuarios a introducir datos correctamente. En lugar de ter que escribir código repetido cada vez que queremos pedir información ao usuario, creamos métodos que fan este traballo de xeito automático e seguro.

0.1.2 Partes Importantes que Debes Entender:

0.1.2.1 1. Métodos Estáticos (`static`) Todos os métodos son `static`, o que significa que podemos chamalos directamente sen necesitar crear un obxecto da clase `Forms`. Por exemplo:

```
Persoa novaPersoa = Forms.persoaForm();
```

Esto é útil para clases utilitarias que só conteñen funcionalidades.

0.1.2.2 2. Validación de Datos Observa como se validan os datos automaticamente: - `Input.readNif()` asegura que o NIF teña o formato correcto - `Input.readPhone()` comproba que o teléfono teña 9 díxitos - `readStringNotEmpty()` non permite campos baleiros para nome e apelidos

Por que é importante? Porque evita que o programa falle máis adiante por datos incorrectos.

0.1.2.3 3. Reutilización de Código O método `contactoForm()` chama a `persoaForm()` para obter os datos da persoa. O método `showContacto()` chama a `showPersona()` para amosar os datos. Isto evita repetir código e fai o programa: - **Máis curto** - **Máis fácil de manter** - **Menos propenso a erros**

0.1.2.4 4. Xestión de Cancelacións Todos os métodos lanzan `CancelException`. Isto significa que se o usuario escribe `*` en calquera momento, o proceso detense e o programa pode reaccionar adecuadamente (por exemplo, voltar a un menú principal).

0.1.2.5 5. Diferentes Tipos de Formularios

- `persoaForm()`: Crea unha nova persoa dende cero

- `contactoForm()`: Crea un novo contacto (incluíndo unha nova persoa)
- `contactoForm(Contacto c)`: Edita un contacto existente, mantendo os datos actuais como opcións por defecto

0.1.3 Conselllos para Principiantes:

1. **Comeza por métodos simples**: Entende primeiro `showPersona()` e `showContacto()` antes de abordar os formularios complexos.
2. **Observa o fluxo de execución**: Cando chamas a `contactoForm()`, o programa:
   - Amosa "Alta de Contacto"
   - Chama a `persoaForm()`
   - Espera que introduzas NIF, nome e apelidos
   - Despois pide teléfono, email e descrición
   - Finalmente crea o obxecto
3. **Practica a reutilización**: Cada vez que teñas código que se repite, pensa se podes crear un método para el. Neste caso, `showPersona()` úsase en varios sitios.
4. **Entende as excepcións**: O `throws CancelException` indica que este método pode cancelarse. Máis adiante aprenderás a manexar estas excepcións con `try-catch`.
5. **Métodos privados**: `readStringNotEmpty()` é `private` porque só se usa dentro desta clase. Isto mantén o código limpo e oculta detalles de implementación.

0.1.4 Por que é un Bo Exemplo?

Esta clase mostra **boas prácticas** de programación: - **Separa responsabilidades**: Cada método fai unha única cousa ben - **Valida datos**: Non acepta información incorrecta - **É reutilizable**: Podes usar estes formularios en diferentes partes do programa - **Ten boa presentación**: Usa títulos e formatos para que o usuario entenda o que se lle pide - **Permite cancelar**: O usuario pode saír en calquera momento