Contents

Este código é unha clase utilitaria en Java que axuda a xestionar a entrada de datos do usuario cunha validación robusta. É ideal para comezar a entender conceptos importantes como clases, métodos, validación de datos e manexo de excepcións. Vou explicarche as partes máis importantes:

0.1 Introdución xeral

Esta clase `Input` pertence ao paquete `com.iesrodeira.utils` e funciona coma unha ferramenta que facilita a interacción co usuario. Contén métodos para ler datos (textos, números, emails, etc.) e validar que estean correctos antes de usalos.

0.2 Constantes e Expresións Regulares

```
private static final String REGEX_PHONE="\\d{9}";
private static final String REGEX_EMAIL_SIMPLE=...;
private static final String REGEX_EMAIL=...;
```

- **`REGEX_PHONE`**: Define un patrón para números de teléfono españoles (9 díxitos).
- **`REGEX_EMAIL`**: Unha expresión regular complexa que valida emails seguindo normas estritas (permite letras, números, puntos e guións en posicións correctas).
- **`static final`**: Estas variables son constantes (non cambian) e pertencen á clase, non a obxectos específicos.

0.3 Métodos de Validación

0.3.1 Validar NIF (DNI)

```
public static void testNif(String nif) throws IllegalArgumentException
```

- Verifica que un NIF teña 8 números + 1 letra de control correcta.
- Usa o algoritmo oficial: o resto da división dos 8 números entre 23 determina a letra correcta.
- Se algo falla, lanza unha `IllegalArgumentException` (excepción que indica que o argumento non é válido).

0.3.2 Validar Email e Teléfono

```
public static void testEmail(String email) throws IllegalArgumentException
public static void testPhone(String phone) throws IllegalArgumentException
```

- Usan as expresións regulares definidas arriba para verificar o formato.
- `matches()` é un método que comproba se unha cadea segue un patrón específico.

0.4 Métodos de Lectura de Datos

0.4.1 Lectura básica con cancelación

```
public static String readText(String title) throws CancelException
```

- Mostra unha mensaxe ao usuario e agarda a súa resposta.
- Se o usuario escribe `*`, cancela a operación lanzando unha `CancelException`.
- Isto ensíñache unha forma elegante de permitir saídas de urxencia.

0.4.2 Lectura con validación automática

```
public static String readPhone(String title) throws CancelException
public static String readNif(String title) throws CancelException
public static String readEmail(String title) throws CancelException
```

- Estes métodos combinan `readText()` coas funcións de validación (`testPhone()`, `testNif()`, `testEmail()`).
- Se o usuario introduce datos incorrectos, mostra un erro e volve pedir os datos **ata que sexan válidos**. Isto úsase un bucle `while (true)` que só remata cando os datos son correctos.

0.4.3 Lectura con valor por defecto

```
public static String readText(String title, String defaultValue) throws CancelException
```

- Permite ao usuario premer `ENTER` para aceptar un valor por defecto (mostrado entre corchetes `[ ]`).
- Útil para agilizar a entrada de datos frecuentes.

0.5 Interface de Usuario (Menús e Formatos)

0.5.1 Menús interactivos

```
public static int menu(String title, Object[] opciones) throws CancelException
```

- Crea un menú numerado automaticamente.
- O usuario escolle un número e o método devolve a opción seleccionada.
- Ten opción de saír (última opción).

0.5.2 Formato visual

```
public static void printTitle(String title)
public static String subline(String text)
```

- `printTitle()` mostra un título con liña de guións por debaixo para mellorar a presentación.
- `subline()` xera a liña de guións co mesmo tamaño que o título.

0.6 Xestión de Excepcións

```
throws CancelException
throws IllegalArgumentException
```

- **`CancelException`**: Creouse para xestionar cancelacións do usuario (cando escribe `*`).
- **`IllegalArgumentException`**: Úsase cando os datos non cumpren os requisitos (NIF incorrecto, teléfono mal formato, etc.).
- Isto ensíñache a **separar** o código normal do código de xestión de erros, o que fai o programa máis robusto.

0.7 Consellos para un principiante

1. **Comeza por métodos simples**: Entende primeiro `readText()` e `printTitle()` antes de abordar as expresións regulares complexas.
2. **Proba os métodos**: Crea un programa simple que use `readPhone()` ou `menu()` para ver cómo funcionan.
3. **Non temas as excepcións**: Son ferramentas para manexar erros de forma ordenada. Observa como se usan os bloques `try-catch`.
4. **As expresións regulares** (`REGEX_EMAIL`) son complexas ao principio. Podes comezar coa versión simple (`REGEX_EMAIL_SIMPLE`) para entender a lóxica básica.

5. **O bucle `while (true)`** con `try-catch` é un patrón moi útil para validacións repetidas.

0.8 Para que serves esta clase?

Imaxina que estás a facer un programa para xestionar contactos: - Cando queres engadir un novo contacto, usas `readNif()` para asegurar que o NIF é válido. - Con `readEmail()` verificas que o email teña formato correcto. - Con `menu()` amosas opcións como "Engadir contacto", "Buscar", "Sair". - Se o usuario se arrepente, pode cancelar en calquera momento premendo `*`.

Esta clase **encapsula** (oculta) toda a complexidade da validación e interacción, de xeito que noutras partes do teu programa só tes que chamar a métodos sinxelos como `Input.readPhone("Teléfono: ")`.

É un excelente exemplo de boas prácticas en programación: reutilización de código, validación de datos, manexo de erros e interface de usuario clara.