Proyecto Titanic — Manual de Usuario y Plan de Pruebas

Integrantes del proyecto

- **Ángel Del Río García y Alejandro Rodríguez Rodríguez**
- Módulo: PSP (Programación de Servicios y Procesos)
- Curso: 2º DAM

Índice

1. Manual de Usuario
    - Cómo ejecutar el programa
        - Requisitos previos
        - Ejecución desde un IDE
        - Ejecución desde línea de comandos
2. Plan de pruebas
    1. Tests de formatos de informe
        - `formatoHtml_debe_lanzar_excepcion_no_implementada`
        - `formatoMarkdown_encabezado_fecha_y_titulo_presentes`
        - `formatoMarkdown_bote_y_totales_formato_correcto`
        - `generadorInforme_crea_fichero_markdown_con_contenido`
    2. Tests para CreadorBote
        - `creadorBote_genera_valores_en_rango_valido`
        - `creadorBote_genera_datos_aleatorios_diferentes`
    3. Test para ServicioEmergencia
        - `servicioEmergencia_no_lanza_excepciones_en_ejecucion_basica`
3. Cómo ejecutar los tests
    - Con Maven (desde terminal)
    - Con VSCode (interfaz gráfica)
4. Comprobante de éxito de los tests realizados

Manual de Usuario

Cómo ejecutar el programa

Para que el programa funcione correctamente, es suficiente con ejecutar la clase principal `AppTitanic`. A continuación, se detallan los pasos:

Requisitos previos

- Tener instalado **Java JDK 8 o superior**.
- Contar con un entorno de desarrollo como **VScode** o ejecutar desde la línea de comandos.

Ejecución desde un IDE

1. Abrir el proyecto `PROYECTOTITANIC` en tu IDE favorito.
2. Navegar a la clase `AppTitanic.java`, ubicada en:

```
src/main/java/es/etg/dam/psp/titanic/AppTitanic.java
```

3. Ejecutar la clase como **Java Application**.
4. El programa iniciará y realizará las operaciones definidas automáticamente.

Ejecución desde línea de comandos

1. Abrir una terminal y navegar a la carpeta raíz del proyecto (`PROYECTOTITANIC`).
2. Compilar el proyecto:

```
mvn clean compile
```

3. Ejecutar la clase principal:

```
java -cp target/classes es.etg.dam.psp.titanic.AppTitanic
```

NOTA IMPORTANTE: Para este proyecto se ha usado la versión 17 de java, por tanto para ejecutar el programa por la terminal se ha usado una terminal JavaSE-17. Se comenta esto ya que al ejecutarlo en versiones como, por ejemplo JDK 25, el programa no compila ni ejecuta correctamente al ser una versión tan reciente. Se recomienda el uso de la versión 17 o 21 y revisar que la terminal sea de una de estás dos versiones.

Plan de pruebas

A continuación se explican los tests implementados en la clase `JUnitTestsTitanic.java` del proyecto, desarrollados con **JUnit 5**.

---

1. Tests de formatos de informe

`formatoHtml_debe_lanzar_excepcion_no_implementada`

- **Objetivo:** Verificar que las funciones de `FormatoHTML` aún no implementadas lancen la excepción `UnsupportedOperationException`.
- **Qué hace:**
  - Comprueba que `escribirEncabezado`, `escribirBote` y `escribirTotales` lancen la excepción esperada.

- **Motivo:** El formato HTML aún no está implementado, y se espera que las funciones den error para indicar que no hay implementación.

`formatoMarkdown_encabezado_fecha_y_titulo_presentes`

- **Objetivo:** Comprobar que el encabezado del informe en formato Markdown contenga la información básica.
- **Qué hace:**
  - Llama a `escribirEncabezado` de `FormatoMarkdown`.
  - Verifica que el encabezado contenga:
    - `# SERVICIO DE EMERGENCIAS` (título principal)
    - `Ejecucion realizada el dia` (fecha de ejecución)
- **Motivo:** Asegurar que el encabezado se genera correctamente antes de listar los botes.

`formatoMarkdown_bote_y_totales_formato_correcto`

- **Objetivo:** Verificar que los datos de cada bote y los totales se escriban correctamente en Markdown.
- **Qué hace:**
  - Crea un `BoteData` con datos de ejemplo.
  - Llama a `escribirBote` y `escribirTotales`.
  - Comprueba que el texto generado contiene:
    - ID del bote (`B-123`)
    - Campos: Total, Hombres, Mujeres, Niños
    - Valores numéricos correctos (por ejemplo `12` total)

`generadorInforme_crea_fichero_markdown_con_contenido`

- **Objetivo:** Verificar que el generador de informes crea un fichero Markdown válido con contenido esperado.
- **Qué hace:**
  - Crea un informe temporal con dos botes de ejemplo.
  - Llama a `GeneradorInforme.generarInforme`.
  - Comprueba que el archivo generado contenga:
    - Encabezado con "SERVICIO DE EMERGENCIAS"
    - Información de ambos botes (`B1`, `B2`)
    - Total acumulado o campo `Total`
  - El archivo temporal se elimina al final del test.

---

2. Tests para CreadorBote

`creadorBote_genera_valores_en_rango_valido`

- **Objetivo:** Comprobar que los valores generados para cada bote sean correctos y coherentes.
- **Qué hace:**
  - Crea un `CreadorBote` con ID `"B1"`.

- Ejecuta `contarPasajeros()` varias veces (5 iteraciones para tests rápidos).
- Comprueba que:
  - Total ≥ 0
  - Hombres, Mujeres y Niños ≥ 0
  - Total = Hombres + Mujeres + Niños

`creadorBote_genera_datos_aleatorios_diferentes`

- **Objetivo:** Comprobar que los datos generados sean aleatorios y no siempre iguales.
- **Qué hace:**
  - Crea un `CreadorBote`.
  - Ejecuta `contarPasajeros()` varias veces (5 iteraciones).
  - Comprueba que haya **más de un total distinto** entre los botes generados.

---

3. Test para ServicioEmergencia

`servicioEmergencia_no_lanza_excepciones_en_ejecucion_basica`

- **Objetivo:** Verificar que la simulación completa de rescate se ejecute sin lanzar excepciones.
- **Qué hace:**
  - Crea un `ServicioEmergencia`.
  - Llama a `lanzarSimulacion()` dentro de `assertDoesNotThrow`.
  - Garantiza que la simulación de los 20 botes se ejecute correctamente, incluyendo la generación de informe Markdown.

Cómo ejecutar los tests

Con Maven (desde terminal)

```
mvn clean test
```

- `clean` elimina compilaciones previas.
- `test` compila y ejecuta los tests en `src/test/java`.
- Los resultados aparecen en la terminal y en `target/surefire-reports` (esta carpeta no aparece en la entrega debido al `.gitignore`).

con VScode (interfaz gráfica)

1. Abrir el proyecto en VScode.
2. Abrir el **panel de Testing**(icono de probeta en la barra lateral izquierda).
3. Ver todos los tests listados bajo `JUnitTestsTitanic`.
4. Ejecutar. Hay distintas formas de ejecutar los tests según preferencia.
   - Todos los tests.

- Una clase completa.
- Un test individual (clic en el icono "play" al lado del test).

5. Los resultados se muestran en el panel de Testing, indicando si pasaron o fallaron.

Comprobante de éxito de los tests realizados

A continuación se muestra una captura de la terminal del proyecto indicando el éxito de la ejecución de los tests realizados para este proyecto.