

# Programación Java: supuesto práctico 1Evaluación

## Ruta hacia Isla Tortuga

### V1.0 (4 puntos)

Se pide un programa que simule una carrera de abordajes pirata hasta Isla Tortuga, un puerto amigo. Para esto tenemos:

#### AventuraPirata.java

Una clase ejecutable, llamada AventuraPirata, que tendrá el main. Esta clase tendrá:

- Una constante int NUMERO\_DE\_RONDAS, donde pondremos el número de rondas que se juegan en la partida. En principio, esta constante vale 10.
- Un String **pirataMasRico** con el nombre del pirata que haya alcanzado el botín más alto
- Un entero con el valor de ese botín (**botinMasAlto**).
- Un String **pirataMasTemido** con el nombre del pirata que haya conseguido más abordajes
- Un entero con el número de abordajes más alto (**maxAbordajesExitosos**)

Los métodos obligatorios en esta clase son:

- **public void puntuacionesFinales(Pirata pirata1, Pirata pirata2):** este método imprime las puntuaciones finales de botín y abordajes exitosos, y notifica quién ha ganado (puede ganar una persona distinta cada cosa, aunque lo normal será que las dos cosas las gane el mismo pirata).
- **public void comprobarRecord(Pirata pirata):** este método comprueba si se ha superado el botín más alto, y el número de abordajes máximo, y si es así los actualiza.
  - Este método se puede separar en comprobarRecordAbordajes y comprobarRecordBotin si te parece más cómodo

Puedes añadir más métodos si consideras que los necesitas.

El método main ejecuta lo siguiente:

1. Pide por teclado los nombres de los dos piratas, y los nombres de sus barcos.
2. Crea los dos piratas (objetos de la clase Pirata, explicada más adelante).
3. Ejecuta la carrera (tantos abordajes como indique NUMERO\_DE\_RONDAS).
4. Muestra los resultados de la carrera
5. Comprueba quién ha ganado, y anuncia el ganador con la puntuación correspondiente.
6. Comprueba si el ganador ha superado el botín de récord, y lo actualiza
7. Pregunta si se quiere jugar otra vez. Si es así, el juego se reinicia.

## Pirata.java

Una clase Pirata. Esta clase describe los piratas de la carrera. Tendrá:

- El **nombre del pirata** (String).
- El **nombre de su barco** (String).
- Los **barcos que el pirata haya abordado**. En cada ronda, el pirata intentará el abordaje de un barco (objeto de tipo Barco, explicado más adelante), y puede tener éxito o no. (Barco[], con una longitud igual al número de rondas que haya).
- Los **abordajes exitosos** (boolean[]), es decir, si el abordaje[3] ha sido exitoso, guardaremos true, y si no guardaremos false

Los métodos obligatorios para esta clase son:

- **public Pirata (String nombre, String barco)**: el constructor necesitará solamente el nombre del pirata y el de su barco, para imprimir mensajes relacionados. Los abordajes y los abordajes exitosos se crearán aquí como arrays del tamaño que indique NUMERO\_DE\_RONDAS.
- **public void intentoDeAbordaje (int ronda)**: este método crea un objeto de tipo Barco, lo intenta abordar, y si tiene éxito se queda con su cargamento.
  - Primero, guardamos el barco atacado en abordajes[ronda], e imprimimos su nombre ("pirata" ataca a "barco atacado").
  - Luego sacamos un nº aleatorio entre 1 y 20 para el pirata, y otro para el barco.
    - Si el pirata tiene el número mayor, el abordaje tiene éxito, y se marca como true en abordajesExitosos.
    - Si no, el abordaje fracasa, y se marca como false en abordajesExitosos.
- **public void status()**: este método imprime el estado del pirata. De momento, imprimirá cuánto lleva en su botín, y cuántos abordajes exitosos ha realizado. Se llama después de un intento de abordaje, sea exitoso o no.
- **public int getAbordajesExitosos()**: este método calcula cuántos abordajes exitosos ha hecho el pirata hasta ahora y devuelve el número.
- **public void resumenViaje()**: este método se usa al final de la partida, y sirve para imprimir el resultado del viaje. Será una lista de nombres de los barcos abordados, y si el abordaje ha sido exitoso o no.

## Barco.java

Esta clase Barco define los barcos que se van a abordar. Tendrá:

- El nombre del barco (String)
- El valor del cargamento (int)
- **public Barco()**: el constructor vacío inicializará el nombre a un nombre aleatorio (con el método **generarNombreBarco()**, que ahora veremos), y también inicializará el valor del cargamento a un valor aleatorio entre 100 y 100.000.
- **public String generarNombreBarco()**: este método tendrá un array de String con nombres de barcos disponibles (mínimo 10, máximo los que queráis), y cuando lo llamemos elegirá uno aleatoriamente y lo devolverá. En un mismo viaje se pueden repetir.



## V2.0 (3 puntos)

Vamos a incluir dos factores a tener en cuenta: la tripulación y la resistencia del barco.

- La tripulación nos dará ventaja a la hora de atacar: Cuanta más tripulación tengamos, más fácil será hacer un abordaje.
- La resistencia es lo que aguanta el barco. Cada abordaje fallido va a dañar nuestro barco.

Esto tiene los siguientes cambios en el programa:

### Barco.java

Se añade `tripulacion` como un atributo (int).

- Constructor: se inicializa con un número aleatorio entre 3 y 10.
- Se incluyen los getters y los setters, como con el resto de atributos.

### Pirata.java

Incluimos estos conceptos de la siguiente manera:

- `TRIPULACION_INICIAL`: int constante que almacena el número inicial de tripulantes que tiene nuestro barco pirata. Valor= 5;
- `RESISTENCIA_INICIAL`: int constante que almacena el valor inicial de la resistencia de nuestro barco. Valor = 6;
- Se incluyen también estas dos cosas como atributos:
  - o Un int `tripulacion`.
  - o Un int `resistencia`.

Cambios en métodos existentes:

- Constructor: se da valor a los atributos `tripulacion` y `resistencia` (el valor inicial que hemos declarado en las constantes).
- `status()`: se incluye el valor de la tripulación y la resistencia del barco en el reporte post-abordaje.
- `intentamosAbordaje()`:
  - o Números aleatorios: ahora, en lugar de tener un número entre 1 y 20, el número será entre 1 y 20 (+tripulación).
    - Si nuestra tripulación es 3, el número será entre 4 y 23
    - Pasará lo mismo con el barco atacado
  - o Si el abordaje tiene éxito, captamos un tripulante para nuestra causa (sumamos 1 a la tripulación).
    - Si el abordaje tiene MUCHO éxito (sacamos el número máximo, 20+tripulación), se suman 2 a nuestra tripulación.
  - o Si el abordaje fracasa, restamos 1 a la resistencia del barco.
    - Si el abordaje fracasa rotundamente (sacamos el número mínimo, es decir, 1+tripulacion), perdemos un tripulante (restamos 1 a la tripulación).

Métodos nuevos:

- **public boolean necesitaRecuperarse()**: devuelve true si la tripulación es 1 (sólo queda el capitán) o si la resistencia del barco es 0 (estamos ya en las últimas).
- **public void reposo()**: el pirata se toma un turno de reposo (atraca en un puerto seguro, para arreglar su barco y conseguir tripulación nueva).
  - Los atributos tripulacion y resistencia vuelven a los valores iniciales.
  - El pirata pierde el 20% de su botín (se lo ha gastado en arreglos del barco y en alojarse en tierra firme).

Por último, el método resumenViaje() puede que necesite un cambio para representar estos turnos de reposo.

### AventuraPirata.java

El único cambio necesario se hará durante la ejecución principal del viaje:

- Antes de intentar el abordaje, se consultará si el pirata necesita descansar.
  - Si lo necesita, ese turno ese pirata hará reposo.
  - Si no, se intentará el abordaje de forma normal.

Opcionalmente, se puede incluir también el valor final de la tripulación y de la resistencia en el conteo final, en el método puntuacionesFinales().

El resto de la clase sigue sin cambios.

### V3.0 (2 puntos)

En esta versión se incluye el tipo de cargamento. Los barcos que los piratas abordan no sólo llevan dinero, sino que a veces llevan joyas, especias, telas o porcelanas. Vamos a indicar todo esto en nuestro programa.

Los cambios a realizar son los siguientes:

#### Barco.java

- Se añade el atributo `tipoCargamento` (`String`).
- Se crea el método `public void generarTipoCargamento()`: este método funcionará de forma muy similar a `generarNombreBarco()`, teniendo un array de al menos 10 tipos de cargamentos, y devolviendo uno aleatorio.
- Se inicializará el atributo `tipoCargamento` dentro del constructor, llamando al método `generarTipoCargamento()`.

#### Pirata.java

Se añade el atributo `tipoBotin` (`String`). En esta cadena se van a concatenar todos los cargamentos que vayamos adquiriendo.

En `intentoAbordaje()`:

- Al tener éxito en un abordaje, añadimos el tipo de cargamento que tuviera el barco.
  - Si ya lo tenemos, no lo volvemos a añadir.
- Al fracasar en un abordaje, perdemos un 10% del botín y un tipo de cargamento. Si no tenemos ningún tipo de cargamento, no pasa nada (se entiende que sólo tenemos monedas).

En `status()`: se incluye el cargamento que llevamos hasta ahora.

En `resumenViaje()`: se incluye el cargamento de cada barco, se haya abordado o no.

Se crea un método nuevo:

- `public void botinMasGrande()`: este método contará la historia del botín más grande con el que el pirata se ha cruzado, haya sido un éxito su abordaje o no. Para esto, recorrerá el array de abordajes, y buscará el Barco con el botín más alto. Cuando lo tenga, si es un éxito dirá que fue su mayor éxito, y si fue un fracaso dirá que fue el fracaso más rotundo, y contará la información del barco en cuestión.

#### AventuraPirata.java

Después de resumir el viaje de cada pirata, se contará su botín más grande.

## Documentación (1 punto)

Se pide de forma obligatoria en todas las clases de la práctica:

- Nombres de variables explicativos
- Código documentado con comentarios y con JavaDoc

La no existencia de esto implicará **1 punto menos por fichero** que no lo tenga.

Como tarea opcional (+1 punto), se puede complementar la práctica con:

- Manual de uso: documento explicativo en el que se detalla cómo funciona el programa y cómo tiene que interactuar con él el usuario.
  - o Es suficiente con tener el manual para la última versión hecha.

## Bola Extra (+0.5 en la nota final)

En lugar de tener solo el nombre del pirata más sanguinario, tenemos una lista de los piratas más buscados de 3 posiciones, ordenada de más a menos abordajes.

- listaMasBuscados (String[3])
- listaAbordajes (int[3])
- comprobarListaMasBuscados() funciona de la siguiente manera:
  - o Comprueba si el pirata entra en la lista de más buscados de acuerdo con el número de abordajes exitosos que tiene.
  - o Si debería estar en la lista y no está, lo incluye. Para esto tiene que buscar la posición en la que tiene que estar, e incluirlo, moviendo los resultados para que siga estando en orden. Haremos lo mismo con la lista de abordajes.