Examen - Unidad 2

Evaluación

Esta prueba permite evaluar, del **RA 2**, los criterios de evaluación **e), f) g)** y **h)**

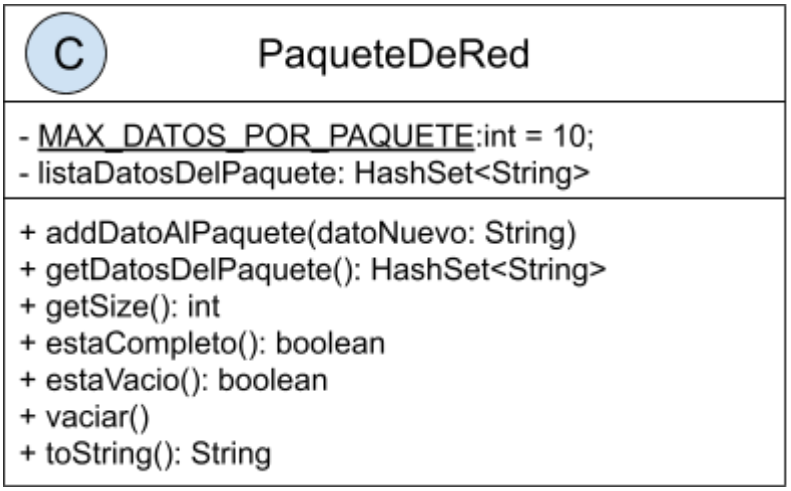
Índice:

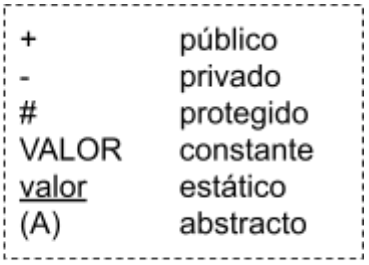
Vamos a implementar la simulación del envío de un paquete de red.

Los paquetes de red de este ejemplo recibirán datos (cadenas de texto) hasta alcanzar su capacidad máxima. Entonces se envían a la red.

PaqueteDeRed - (2 puntos)

Esta clase representa los paquetes de red de nuestro ejercicio:





El estado de los objetos de este tipo se representa por los siguientes atributos:

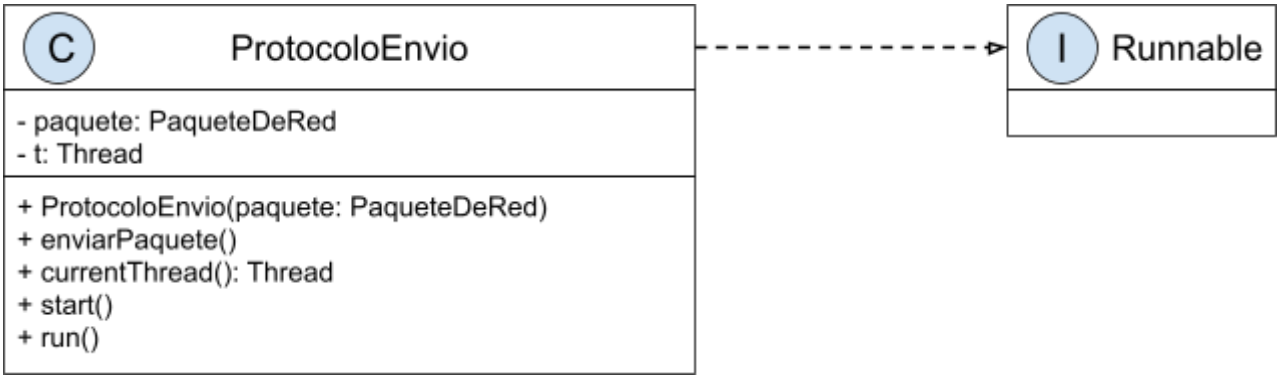
- **`MAX_DATOS_POR_PAQUETE`**: representa la capacidad máxima (10) de datos que puede transportar el paquete.
- **`listaDatosDelPaquete: HashSet<String>`**: conjunto de datos (textos) del paquete.

Los paquetes ofrecen estas funcionalidades:

- **`addDatoAlPaquete(datoNuevo: String)`**: añade un nuevo dato al paquete siempre que no esté completo. Mostrará por pantalla el dato añadido.
- **`getDatosDelPaquete(): HashSet<String>`**: devuelve el conjunto de datos del paquete.
- **`getSize(): int`**: devuelve la cantidad de datos que contiene el paquete.
- **`estaCompleto(): boolean`**: comprueba si el paquete está lleno.
- **`estaVacio(): boolean`**: comprueba si el paquete está vacío.
- **`vaciar()`**: elimina todos los datos del paquete.
- **`toString()`**: devuelve una representación textual del paquete, incluyendo la lista de datos que contiene.

ProtocoloEnvio - (4 puntos)

Esta clase representa el protocolo que se encarga de enviar paquetes de red:



El estado para el protocolo se define mediante los atributos:

- **`paquete: PaqueteDeRed`**: representa el paquete de red que esta clase se encarga de enviar.
- **`t: Thread`**: hilo en el que se ejecutará el protocolo de envío.

El protocolo puede realizar las siguientes acciones:

- **`ProtocoloEnvio(paquete: PaqueteDeRed)`**: recibe como parámetro el paquete de red que se enviará e inicializa el hilo.
- **`enviarPaquete()`**: simula el envío del paquete de red mostrando un mensaje por pantalla indicando el tamaño del paquete y luego vaciando el paquete.
- **`currentThread(): Thread`**: devuelve una referencia al hilo actual asociado a este protocolo.
- **`start()`**: inicia la ejecución del hilo t.
- **`run()`**: contiene el trabajo que realizará el hilo, que es el siguiente:
  - Mientras el hilo no sea interrumpido, ejecutará un bloque de código sincronizado (indica el objeto monitor para dicho bloque) que verifica si el paquete está completo.
    - Si el paquete no está completo, el hilo esperará a que se le notifique que puede enviarlo utilizando paquete.wait().
    - Si el hilo es interrumpido mientras espera, mostrará un mensaje y, si el paquete no está vacío, lo enviará con los datos que tenga (aunque no esté completo).
  - En cualquier otro caso, cuando se le notifique, el hilo enviará el paquete.

Main - (4 puntos)

Para probar el funcionamiento, vamos a instanciar un objeto de cada una de las clase anteriores y lanzar el hilo del proceso.

Haz un bucle que vaya llenando el paquete. El bucle debe añadir más de 10 elementos.

Dentro del bucle, crea un bloque de código sincronizado (indica el objeto monitor para dicho bloque) que vaya añadiendo datos al paquete (del tipo “Dato i”, por ejemplo) hasta que el paquete esté lleno, en cuyo caso notificará al protocolo que lo envíe y mostrará un mensaje indicándolo.

Cuando el bucle haya finalizado, lanza una interrupción al hilo para que sepa que debe finalizar su ejecución.

Nota: es recomendable hacer distintas pruebas, cambiando el nº de iteraciones el bucle: menos de 10, 10, más de 10 siendo múltiplo de 10, más de 10 no siendo múltiplo de 10 (para ver que se envían los paquetes incompletos), etc.