**Guía de Ejercicios Nº 11**

*Cada clase debe contener dos constructores, uno por defecto y otro que reciba todos los atributos de la clase. Debe contener además los getter y setter de cada atributo. Las clases deben estar encapsuladas.*

**Excepciones**

1. Crear la excepción **no chequeada** RutNoValidoException. En la clase Rut, en el método validar, se debe lanzar la excepción si el rut es incorrecto.
2. Crear la Clase Usuario que tenga como atributo nombre usuario, password y intentos fallidos, debe tener un método login, para simular un login debe preestablecer los datos validos ej. Usuario: jmaldonado y password 123456. Si los datos son incorrectos debe lanzar la excepción **chequeada** AutenticacionException y aumentar el contador en 1 de intentos fallidos, si se ha intentado ingresar mas de tres veces debe lanzar la excepción **chequeada** CuentaBloqueadaException.
3. Capture la excepción ArrayIndexOutOfBoundsException al intentar acceder a la posición de un arreglo fuera de rango .

String[] str = { "foo" };

String functionName = str[10];

1. Crear la GUI (Interfaz grafica de usuario) para el ejercicio 2 y capturar las excepciones e informarlas en la interfaz.
2. Solicitar un numero por cuadro de dialogo y capturar la excepción NullPointerException, si el usuario presionar cancelar y capturar la excepción si el usuario ingresa una letra en vez de un número. Cada excepción al ser capturada desplegara un nuevo cuadro de dialogo indicando el error y solicitara ingresar nuevamente el valor.

**Threads**

1. Crear la clase cuenta bancaria Cuenta Bancaria que posee los atributos saldo y numero de cuenta. Debe tener el método depositar dinero el cual sumara el dinero a depositar al saldo de la cuenta y un método retirar dinero el cual restara dinero del saldo de la cuenta. Crear el Thread RetirarDineroThread que tiene como atributo una cuenta bancaria y el monto a retirar, sobreescribir el método run el cual retirara el dinero de la cuenta. Crear el Thread DepositarDineroThread que tiene como atributo una cuenta bancaria el el monto a depositar en la cuenta, sobreescribir el método run el cual deberá depositar el dinero solicitado en la cuenta.

Crear una instancia de Cuenta Bancaria, 3 instancias de RetirarDineroThread y 3 instancias de DepositarDineroThread, a los 6 threads deberá pasar la instancia de Cuenta Bancaria creada anteriormente.

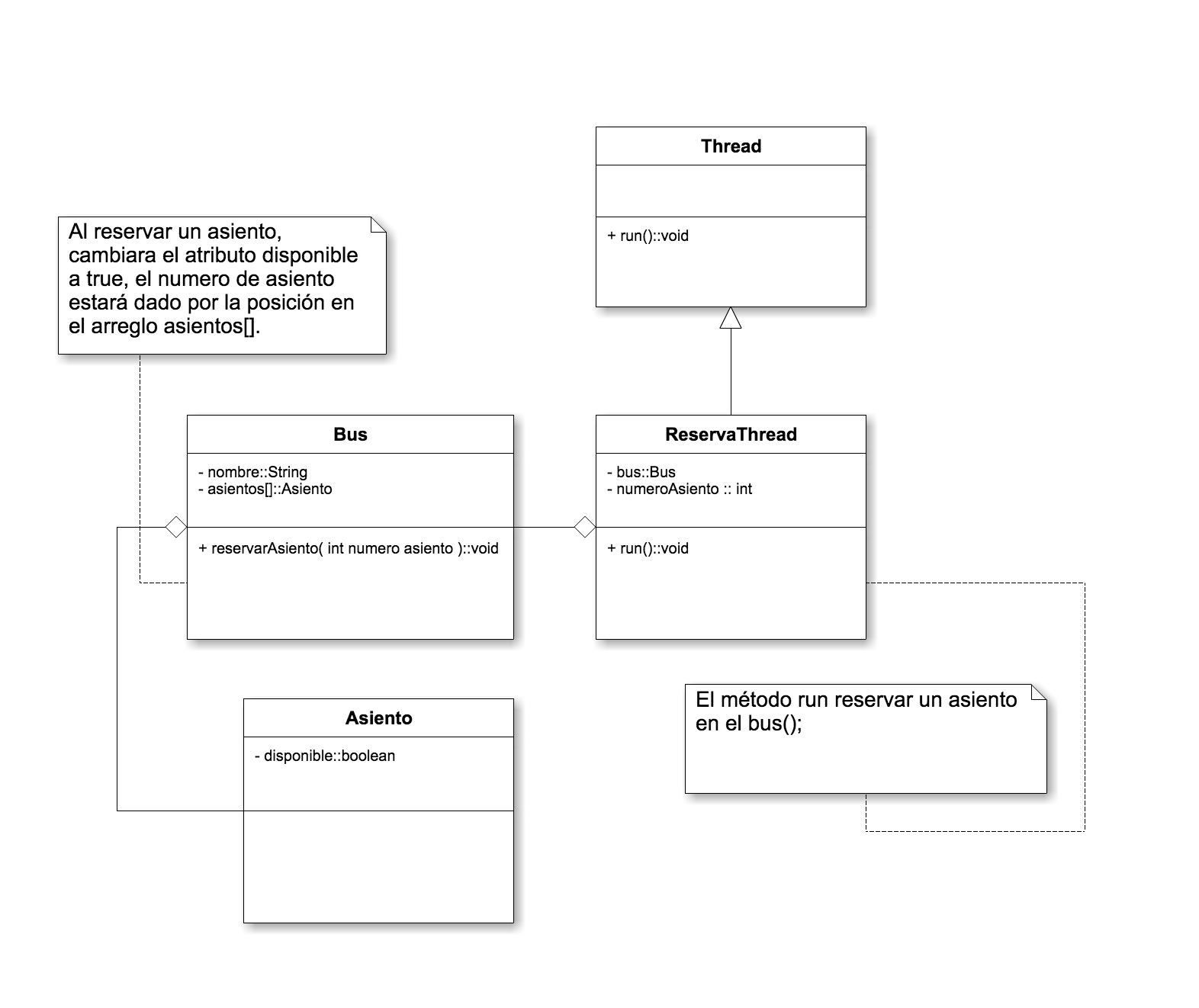
1. Crear la clase animal que tiene como atributos de instancia: nombre(String), velocidad(int) descanso (long) y posición (int). Además deberá tener un atributo de clase llamado carrera Finalizada de tipo boolean. La clase animal implementa la Interfaz Runnable. El animal se desplaza según su velocidad, para representar el desplazamiento se cambiara la posición, (posición+=velocidad) y cada vez que avance el animal deberá tomarse un descanso.

Thread.sleep((long) (0 + (Math.random() \* this.descanso)));

El método run representara una carrera de animales, la carrera termina cuando un animal llega a la posición 100.

Crear Tres instancias de Animal y ejecutar cada una en un thread diferente e inicializar los thread. Se debe imprimir por consola el nombre del animal que gano la carrera.

1. Desarrollar las siguientes clases



Crear una instancia de Bus con 10 asientos y 2 instancias de ReservaThread solicitado un mismo asiento, debe sincronizar los hilos para evitar el problema de concurrencia data-race.