ULP Virtual

Tecnicatura Universitaria en Desarrollo de Software

Programación II

Guía 3

Ejercicios Extras

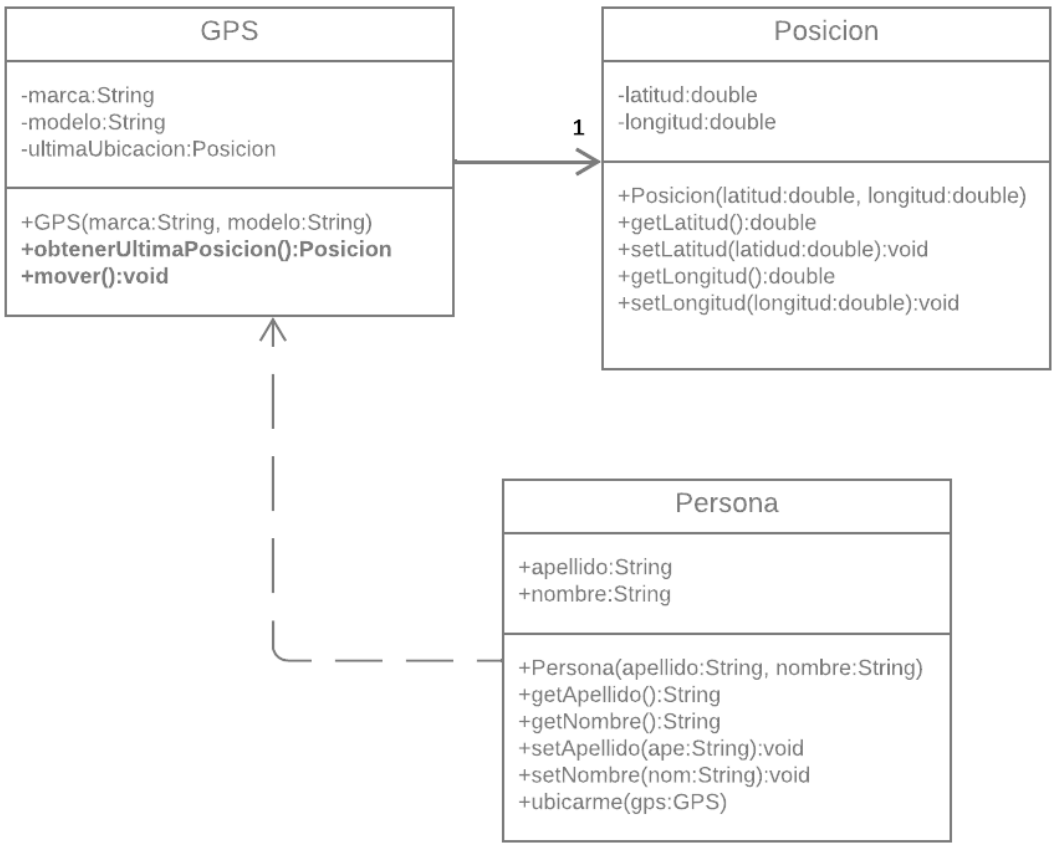
Universidad de LA PUNTA

GOBIERNO DE SAN LUIS

EJERCICIOS DE APRENDIZAJE EXTRA

Estos van a ser ejercicios para reforzar los conocimientos previamente vistos. Estos pueden realizarse cuando hayas terminado la guía y tengas una buena base sobre lo que venimos trabajando. Además, si ya terminaste la guía y te queda tiempo libre en las mesas, podes continuar con estos ejercicios extra, recordando siempre que no es necesario que los termines para continuar con el tema siguiente. Por último, recordá que la prioridad es ayudar a los compañeros de la mesa y que cuando tengas que ayudar, lo más valioso es que puedas explicar el ejercicio con la intención de que tu compañero lo comprenda, y no sólo mostrarlo. ¡Muchas gracias!

Dado el siguiente modelo, impleméntelo en Java.



Como vemos en el modelo, una Persona usa un GPS y el GPS tiene una Posición. Con respecto a la clase GPS la lógica de sus métodos especiales es la siguiente:

- **mover()** Este método generará una nueva Posición con valores aleatorios para la latitud y longitud de la misma entre 0 y 1. Esta nueva Posicion quedará registrada en la variable atributo "*ultimaUbicacion*"
- **obtenerUltimaPosicion()** Este método retornará el estado actual de la variable atributo "*ultimaUbicacion*"

En la clase Persona, el método **ubicarme()** es el que usa el GPS para solicitar que se mueva y luego le pide la última posición que luego mostrará por pantalla.

Ejercicio 2:

En el siguiente modelo, tenemos una clase Jugador con los atributos: nombre y clasificación; un constructor que inicializa su atributo nombre y los métodos:

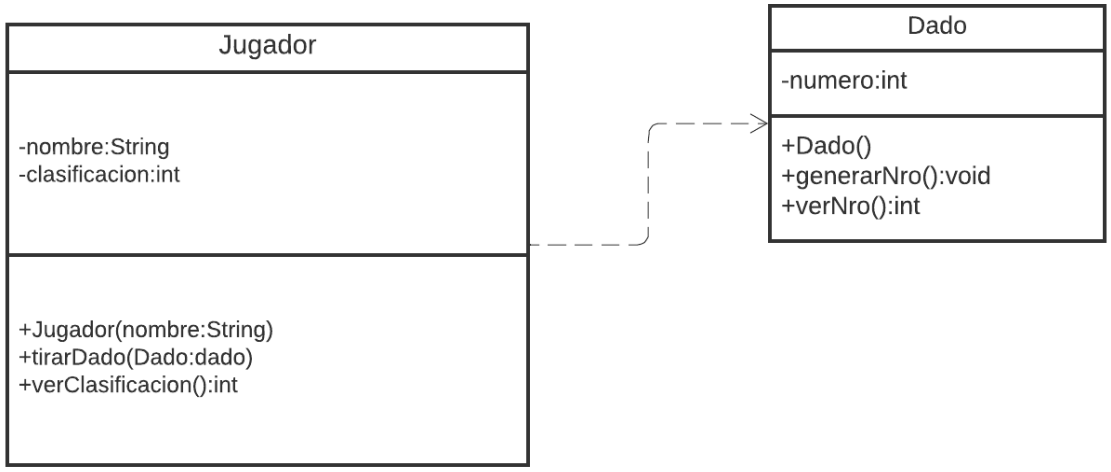
tirarDado(): Que recibe por parámetro un Dado, lo que hará este método es invocar el método generarNro del dado y acumulará el valor del Dado en su atributo clasificación.

verClasificación(): Retornará la clasificación del jugador.

Por otro lado, tenemos la clase Dado que posee como atributo: número que es un valor entero y los métodos:

generarNro(): Que genera un número al azar entre 1 y 6 que se lo asigna a su variable atributo numero, si el atributo tiene un valor mayor a 0, significa que ese dado ya generó un valor y lo mantendrá.

verNro(): Retornará el valor guardado en la variable numero.



Desde una clase TestJuego se pide:

a) Crear 3 Dados.
b) Crear un Jugador de nombre “Pocho” y pedirle que tire los 3 Dados.
c) Mostrar al final la clasificación del Jugador.
d) Ahora Crear otros 3 Dados más.
e) Crear un nuevo Jugador de nombre “Pepe” y pedirle que tire los 3 últimos dados creados.
f) Al finalizar mostrar de los 2 jugadores el nombre que obtuvo la mayor clasificación.

Ejercicio 3:

En un nuevo proyecto, nos piden modelar e implementar lo siguiente:

Definir un paquete de nombre “entidades” con las clases: Cliente, Producto y Vendedor. En otro paquete de nombre “Negocio” tendremos la clase Venta que posee como atributos: un Cliente, el Vendedor y 3 Productos, además de un método calcularTotal que sumará los precios públicos de los productos y hará un descuento de acuerdo a la ciudad en donde vive el Cliente: si es de San Luis, posee un descuento del 10% y si es de otra provincia un 15% de descuento. Los Productos tienen como atributos: descripción, precio de lista, stock y tipo de producto; además de un método especial calcularPrecioPublico que retornará el precio de lista más un 25% si el producto es de tipo “Lácteo”, un 35% si el producto es de tipo “Limpieza” y un 45% para cualquier otro tipo de producto.

Desde el método main de una clase TestComercio se pide:

a) Crear 3 Productos.
b) Crear un Cliente y un Vendedor.
c) Crear una Venta del Vendedor creado en el punto b para el Cliente creado en el punto b y con los 3 productos del punto a.
d) Solicitar a la Venta que informe el total.

Ejercicio 4:

En un nuevo proyecto, nos piden modelar e implementar lo siguiente:

Realizar el juego de la ruleta rusa de agua en Java. Como muchos saben, el juego se trata de un número de jugadores, que, con un revolver de agua, el cual posee una sola carga de agua, se dispara y se moja. Las clases a hacer del juego son las siguientes:

Clase Revolver de agua: esta clase posee los siguientes atributos: posición actual (posición del tambor que se dispara, puede que esté el agua o no) y posición agua (la posición del tambor donde se encuentra el agua). Estas dos posiciones, se generarán aleatoriamente.

Métodos:

• llenarRevolver(): le pone los valores de posición actual y de posición del agua. Los valores deben ser aleatorios.

• mojar(): devuelve true si la posición del agua coincide con la posición actual

• siguienteChorro(): cambia a la siguiente posición del tambor.

• toString(): muestra información del revolver (posición actual y donde está el agua)

Clase Jugador: esta clase posee los siguientes atributos: id (representa el número del jugador), nombre (Empezara con Jugador más su ID, “Jugador 1” por ejemplo) y mojado (indica si está mojado o no el jugador).

Métodos:

disparo(Revolver r): el método, recibe el revolver de agua y llama a los métodos de mojar() y siguienteChorro() de Revolver. El jugador se apunta, aprieta el gatillo y si el revolver tira el agua, el jugador se moja; por lo tanto el atributo mojado pasa a true, sino false.

Clase Juego: esta clase posee los siguientes atributos: 2 Jugadores y Revolver

Métodos:

• llenarJuego(Jugador jugador1,Jugador jugador2, Revolver r): este método recibe los 2 jugadores y el revolver para guardarlos en los atributos del juego.

• ronda(): cada ronda consiste en un jugador que se apunta con el revolver de agua y aprieta el gatillo. Sí el revolver tira el agua el jugador se moja y se termina el juego, sino se moja, se pasa al siguiente jugador hasta que uno se moje. Si o si alguien se tiene que mojar. Al final del juego, se debe mostrar que jugador se mojó. Pensar la lógica necesaria para realizar esto, usando los atributos de la clase Juego.

Ejercicio 5:

En un nuevo proyecto, nos piden modelar e implementar lo siguiente:

Para regar un parque se dispone de una Bomba, la cual posee un Motor y este un Tanque de Agua. El Tanque de Agua tiene una capacidad máxima es de 10.000 litros y los comportamientos para saber si su capacidad está al límite o vacía y otro cargar que cada vez que se lo invoca carga 500 litros. El Motor puede trabajar con 3 velocidades distintas (ALTA, MEDIA y BAJA) y un comportamiento consumirAgua que cada vez que se lo invoca consume 10 litros si trabaja a velocidad ALTA, 5 litros si trabaja a velocidad MEDIA y 1 litro si trabaja a velocidad BAJA; otro comportamiento cambiarVelocidad que cada vez que se lo usa, cambia la velocidad de Alta a Media y de Media a Baja y vuelve a empezar. La Bomba tiene un comportamiento regar que hace que el Motor consuma agua hasta que su tanque quede completamente vacío y otro comportamiento mostrarInfor que muestra por pantalla la carga inicial del tanque, la velocidad del motor.

Se pide desde el método main de una clase TestRegar:

a) Crear un Tanque y llenarlo con 5000 litros de agua.
b) Crear un Motor con este Tanque y que trabaje a velocidad Media.
c) Crear una Bomba con el motor instanciado en el punto b.
d) Hacer que la Bomba riegue y al finalizar invocar al método mostrarInfo.