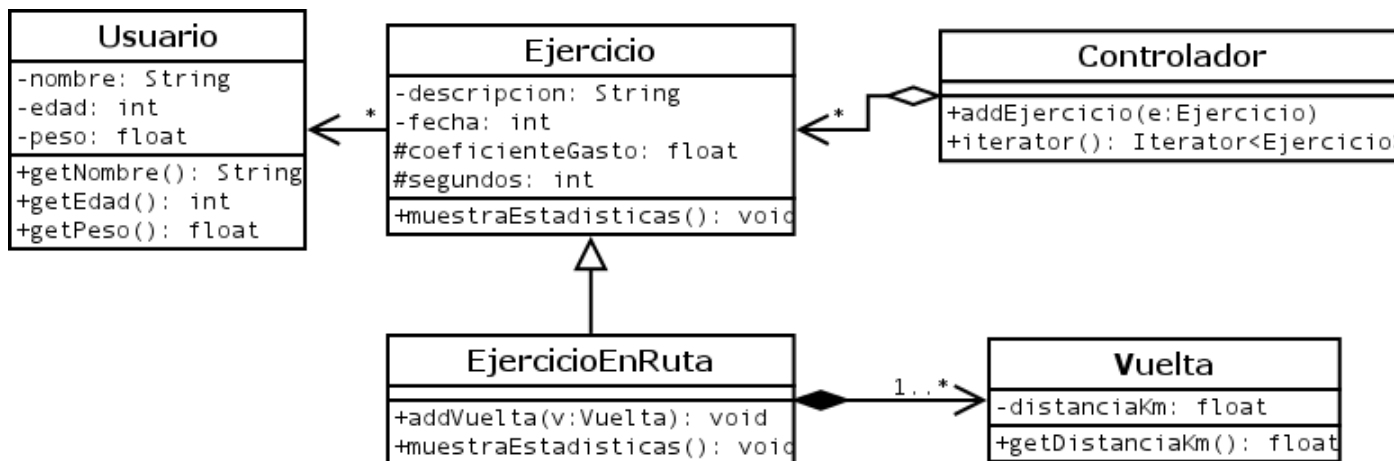


Metodología i Programació Orientada a Objectes

Examen parcial G40. Viernes 24/4/2015.

Estamos diseñando una app para guardar un registro de nuestra actividad física, según el siguiente diagrama UML:



El **Controlador** de la aplicación guarda un registro de **Ejercicios** que pertenecen en **Usuario**, descrito por su nombre, edad y peso en Kg.

De cada ejercicio guardamos una descripción en texto (por ejemplo, "saltar a la comba"), la fecha de realización como un entero en formato AAAAMMDD (por ejemplo, el 25/5/2014 se guardaría como 20140525), el tiempo en segundos que ha durado el ejercicio, así como un coeficiente de gasto calórico, que nos da las calorías gastadas por un ejercicio según la fórmula:

$$\text{KCalorías gastadas} = \text{peso del usuario} * \text{segundos} * \text{coeficiente gasto}$$

Hay un tipo especial de ejercicio, el **Ejercicio en Ruta**, que guarda información ejercicios en los que se recorren distancias (ciclismo, atletismo...). Los ejercicios en ruta guardan un registro de **Vueltas**, que guardan información sobre las distancias parciales del ejercicio. Cada vuelta guarda la distancia de ésta, en Kilómetros.

Ejercicios

- (2.5 puntos) Define las clases y sus atributos en Java, de tal manera que se refleje las información de las clases, así como las relaciones entre éstas. No hay que definir cabeceras de métodos ni de constructores, solo la de las clases y sus atributos.
- (0.25 puntos) Pregunta: para que el controlador guarde Ejercicios en Ruta, ¿debemos trazar alguna relación directa entre la clase **Controlador** y la clase **EjercicioEnRuta**? Justifica tu respuesta.
- (0.25 puntos) Pregunta: el rombo de la flecha entre la clase **Controlador** y **Ejercicio** es blanco mientras el rombo de la flecha entre la clase **EjercicioEnRuta** y **Vuelta** es negro. ¿Qué diferencia hay?
- (1.5 puntos) Programar clase **Usuario** entera, de tal manera que se puedan inicializar y leer sus atributos desde fuera de la clase.

5. (1.5 puntos) Programa el método `muestraEstadisticas()` de la clase `Ejercicio`. Por ejemplo, un ejercicio cuya descripción = "salto a la comba", fecha = 20150420, segundos = 914, `coeficienteGasto`=0.0075 y su usuario pesa 78 Kg, mostraría por pantalla su información de la siguiente manera:

```
== Estadísticas de 'Salto a la comba' ==  
Fecha: 20/4/2015  
Duracion: 15:14  
Calorías consumidas: 534.69 KCal
```

6. (2 puntos) Programa el método `muestraEstadisticas()` de la clase `EjercicioEnRuta`. Ésta debe mostrar la misma información que la clase `Ejercicio`, además de las estadísticas de ruta: la distancia total recorrida (calculada como la suma de todas las vueltas) y la velocidad media en Km/h (calculada como la distancia total dividida por el número/fracción de horas durante las cuales se ha practicado el ejercicio). **IMPORTANTE: la implementación método no debe repetir código que ya se haya escrito en la función "muestraEstadisticas()" de la superclase.** Un ejemplo de salida por pantalla sería:

```
== Estadísticas de 'Carrerilla por el parque' ==  
Fecha: 22/4/2015  
Duracion: 25:25  
Calorías consumidas: 505.53754 KCal  
Estadísticas de ruta:  
  Distancia total: 5.33 Km  
  Velocidad media: 12.582295 Km/h
```

7. (2 puntos) Consideremos la siguiente clase incompleta:

```
public class UnaClaseCualquiera {  
    public static void muestraTodosLosEjercicios(Controlador c) {  
        /* Tu código aquí */  
    }  
}
```

Rellena el cuerpo del método `muestraTodosLosEjercicios`, que mostraría por pantalla todos los ejercicios almacenados en el objeto `Controlador` que se le pasa por parámetro.

A considerar: el contenedor de Ejercicios es privado en la clase `Controlador`, y no hay *getter*. Puedes usar el método `Iterator<Ejercicio> iterator()` que está definido en la clase `Controlador`, que te devuelve un nuevo iterador a los Ejercicios registrados.

Solución

Ejercicio 1

```
public class Controlador {
    private ArrayList<Ejercicio> ejercicios = new ArrayList<Ejercicio>();
}

public class Ejercicio {
    private String descripcion;
    private int fecha;
    protected float coeficienteGasto;
    protected int segundos;
    private Usuario usuario;
}

public class EjercicioEnRuta extends Ejercicio {
    private List<Vuelta> vueltas = new ArrayList<Vuelta>();
}

public class Usuario {
    private String nombre;
    private int edad;
    private float peso;
}

public class Vuelta {
    private float distanciaKm;
}
```

Ejercicio 2

No se debe trazar ninguna relación directa entre Controlador y EjercicioEnRuta, ya que esta última hereda las relaciones de su superclase Ejercicio con Controlador.

Ejercicio 3

Entre Controlador y Ejercicio existe una relación de agregación. Si se elimina la instancia del controlador, los ejercicios que ésta contuviera no deben ser eliminados (podrían formar parte de otro controlador). Entre EjercicioEnRuta y Vuelta hay una relación de composición: si se elimina el Ejercicio en Ruta, las vueltas deberían eliminarse, ya que éstas pertenecen exclusivamente a dicho ejercicio y no tendría sentido que existieran a parte.

Ejercicio 4

```
public class Usuario {
    private String nombre;
    private int edad;
    private float peso;
    public Usuario(String nombre, int edad, float peso) {
        this.nombre = nombre;
        this.edad = edad;
        this.peso = peso;
    }
    public String getNombre() {
        return nombre;
    }
    public int getEdad() {
        return edad;
    }
    public float getPeso() {
        return peso;
    }
}
```

Ejercicio 5

```
public void muestraEstadisticas() {
    System.out.println("== Estadísticas de '"
        + descripcion + "' ==");
    System.out.println("Fecha: " + (fecha%100) + "/" +
        (fecha/100)%100 + "/" + (fecha/10000));
    System.out.print("Duracion: " + (segundos/60) + ":");
    if(segundos%60<10) {
        System.out.print("0");
    }
    System.out.println(segundos%60);
    System.out.println("Calorías consumidas: "
        + usuario.getPeso() * segundos * coeficienteGasto + " KCal");
}
```

Ejercicio 6

```
public void muestraEstadisticas() {
    super.muestraEstadisticas();
    System.out.println("Estadísticas de ruta:");
    float distanciaTotal = 0;
    for(Vuelta v : vueltas) {
        distanciaTotal += v.getDistanciaKm();
    }
    System.out.println("\tDistancia total: "
        + distanciaTotal + " Km");
    System.out.println("\tVelocidad media: " +
        distanciaTotal/(segundos/3600f) + " Km/h");
}
```

Ejercicio 7

```
public void muestraTodosLosEjercicios(Controlador c) {
    Iterator<Ejercicio> itej = c.iterator();
    while(itej.hasNext()) {
        itej.next().muestraEstadisticas();
    }
}
```