Programación Orientada a Objetos
Grado de Ingeniería Informática en Sistemas de Información - Curso 2023/24

ENSEÑANZA PRÁCTICA Y DE DESARROLLO

EPD FINAL

IMPORTANTE: La entrega del trabajo se hará EXCLUSIVAMENTE a través de la actividad correspondiente en el Aula Virtual antes de las 8:00 horas del LUNES 13 de MAYO de 2024. Pasada esta hora la aplicación no permitirá la entrega del trabajo, quedando el ejercicio sin calificar para el alumno. La entrega consistirá en subir un único fichero (.ZIP) con los códigos fuente (ficheros .JAVA) de la solución al problema.

Se pretende desarrollar una aplicación en Java que simule un sistema de gestión para las Olimpiadas de París 2024, permitiendo gestionar atletas, deportes y resultados de competiciones. Este sistema debe proporcionar funcionalidades para registrar nuevos atletas, añadir resultados de las competiciones, y visualizar el medallero olímpico. Para ello, se debe implementar una aplicación siguiendo el diagrama de clases facilitado en este enunciado.

Descripción de Clases e Interfaces

A continuación, se indican los detalles necesarios para implementar todas las clases e interfaces del modelo representado por el diagrama de clases de la última página de este enunciado. Note que esta descripción sólo indica aquellos aspectos que no se pueden derivar directamente del modelo, es decir, que deberá implementar todos los elementos indicados del diagrama de clases para poder modelar la operativa del juego (descrita más adelante) aunque no se indiquen en este apartado.

Interfaz Competidor. Esta interfaz define los métodos comunes que cualquier tipo de competidor debe implementar, proporcionando una base para cualquier extensión futura que pueda involucrar diferentes tipos de competidores. Asimismo, define la constante MAXMEDALLAS=100.

Clase Persona: Esta es la clase base para cualquier persona involucrada en las Olimpiadas, incluyendo atletas, entrenadores, o personal de apoyo, si se deseara extender la funcionalidad del sistema en el futuro.

Atributos:

- nombre: Nombre de la persona.
- país: País de origen de la persona. El país siempre debe ser almacenado en mayúsculas.
- edad: Edad de la persona. La edad siempre tiene que ser positiva.

Métodos:

- Constructor para inicializar los atributos.
- Métodos gets para cada atributo.
- setEdad: La edad siempre tiene que ser positiva.
- toString: El formato del String de salida sería el siguiente:

```
Rafael Nadal (ESPAÑA) - 37 años
```

Clase Atleta: Representa a un atleta participante en las Olimpiadas, implementando la interfaz Competidor y extendiendo la clase Persona y especificando detalles relevantes a su participación en eventos deportivos.

Atributos:

- disciplinas: String con las disciplinas deportivas en la que el atleta compite, separadas por comas. Siempre debe ser almacenada en mayúsculas.
- medallas: Vector de objetos Medalla para almacenar las medallas ganadas.
- contadorMedallas: Contador de medallas añadidas.

Métodos:

- Constructor para inicializar los atributos. Las disciplinas siempre deben ser almacenadas en mayúsculas. El vector de medallas se creará con el tamaño indicado por la constate MAXMEDALLAS de la interfaz Competidor.
- añadirMedalla: Añade una medalla al array de medallas del atleta, siempre que haya espacio en el vector. En caso contrario, mostrará un mensaje por pantalla indicando esta circunstancia.
- toString: El formato del String de salida sería el siguiente:

```
- ATLETA: Rafael Nadal (ESPAÑA) - 37 años
  -- Disciplinas: TENIS INDIVIDUAL, TENIS DOBLES
  -- Medallas:
   ---PLATA en DOBLES MIXTO
   ---ORO en TENIS INDIVIDUAL
   --*Total Medallas: 1 oro, 1 plata y 0 bronce
```

Note que la parte sombreada coincide con lo devuelto por los métodos toString de las clases Persona y Medalla, en cada caso. Asimismo, la última línea muestra el conteo de cada tipo de medalla, por lo que deberá procesar el vector de medallas para obtener esta información.

Clase Medalla. Representa una medalla otorgada a un/una atleta en las Olimpiadas.

Atributos:

- tipo: Tipo de medalla. Sólo puede tomar valores entre las tres constantes declaradas en la propia clase.
- disciplina: Disciplina deportiva en el que se otorgó la medalla. El atributo disciplinas siembre debe ser almacenado en mayúsculas.
- Las constantes ORO=1, PLATA=2 y BRONCE=3 representan los tres tipos de medalla.

Métodos:

- Constructor para inicializar los atributos. El atributo tipo sólo puede tomar valores entre las tres constantes declaradas en la propia clase. En caso de no ser un valor válido, se inicializará por defecto a BRONCE.
- getTipoString: devuelve un String con la cadena “ORO”, “PLATA” o “BRONCE”, según sea el valor del atributo tipo.
- toString(): El formato del String de salida sería el siguiente (siendo “ORO” el tipo y “TENIS INDIVIDUAL” la disciplina):

```
PLATA en DOBLES MIXTO
ORO en TENIS INDIVIDUAL
```

Clase Deporte. Representa un deporte practicado en las Olimpiadas y contiene los eventos asociados a ese deporte.

Atributos:

- nombre: Nombre del deporte, que siempre deberá estar en mayúsculas.
- eventos: Vector de objetos Evento.
- contadorEventos: Contador de eventos añadidos.

Métodos:

- Constructor para inicializar los atributos. El vector de eventos se instancia con un tamaño máximo indicado por el parámetro maxEventos, e inicialmente no contendrá ningún evento.
- añadirEvento: Añade un evento al deporte, siempre que haya espacio en el vector. En caso contrario, mostrará un mensaje por pantalla indicando esta circunstancia.
- Método get para el nombre.
- toString: El formato del String de salida sería el siguiente:

```
- DEPORTE: Tenis - Lista de Eventos:
   -> EVENTO: 1ª Semifinal Individual <25/07/2024>
   ---> Rafael Nadal (ESPAÑA)
   ---> Novak Djokivik (SERBIA)
   -> EVENTO: 2ª Semifinal Individual <25/07/2024>
   ---> Carlos Alcaraz (ESPAÑA)
   ---> Jannik Sinner (ITALIA)
   -> EVENTO: Final Individual <28/07/2024>
       <Aún no hay atletas registrados en este evento>
```

Note que la parte sombreada coincide con lo devuelto por el método toString de la clase Evento.

Clase Evento. Representa un evento específico en un deporte durante las Olimpiadas.

Atributos:

- nombre: Nombre del evento.
- fecha: Fecha en que se realiza el evento en formato String, que siempre debe cumplir el patrón "dd/mm/aaaa".
- atletas: Vector de atletas participantes en el evento.
- contadorAtletas: Contador de atletas añadidos.

Métodos:

- Constructor para inicializar los atributos. El atributo fecha debe cumplir el patrón "dd/mm/aaaa". Si no lo cumple, se inicializa a "01/01/1900". El vector de atletas participantes se instancia con un tamaño máximo indicado por el parámetro maxAtletas, e inicialmente no contendrá ningún atleta.
- añadirAtleta: Añade un atleta al vector de atletas, siempre que haya espacio para ello. En caso contrario, mostrará un mensaje por pantalla indicando esta circunstancia.
- Método get para el nombre.
- setFecha: establece el atributo fecha comprobando que se cumple el patrón "dd/mm/aaaa". Si no se cumple, no cambia la fecha.
- toString: El formato del String de salida sería el siguiente:

```
EVENTO: 1ª Semifinal Individual <25/07/2024>
   ---> Rafael Nadal (ESPAÑA)
   ---> Novak Djokivik (SERBIA)
```

Si el evento no cuenta con atletas participantes, mostrará el mensaje "<Aún no hay atletas registrados en este evento>". Por ejemplo:

```
EVENTO: Final Individual <28/07/2024>
       <Aún no hay atletas registrados en este evento>
```

Clase SistemaOlimiadas. Clase central del sistema que integra todas las clases.

Atributos:

- atletas: Vector de atletas registrados en las olimpiadas.
- deportes: Vector de deportes de las olimpiadas.
- contadorAtletas: Contador de atletas registrados (añadidos al vector)
- contadorDeportes: Contador de deportes añadidos al vector.

Métodos:

- Constructor para inicializar los atributos. El vector de atletas se instancia con un tamaño máximo indicado por el parámetro maxAtletas, e inicialmente no contendrá ningún atleta. Igualmente, el vector de deportes se instancia con un tamaño máximo indicado por el parámetro maxDeportes, e inicialmente no contendrá ningún deporte.
- registrarAtleta: Añade un atleta al vector de atletas, siempre que haya espacio para ello. En caso contrario, mostrará un mensaje por pantalla indicando esta circunstancia.
- añadirDeporte: Añade un deporte al vector de deportes, siempre que haya espacio para ello. En caso contrario, mostrará un mensaje por pantalla indicando esta circunstancia.
- mostrarSistema: Muestra por pantalla toda la información almacenada en el sistema de Olimpiadas (Atletas regisyrados y Eventos añadidos), basándose en los formatos de los métodos toString expuestos anteriormente, con el siguiente formato:

```
## SISTEMA OLIMPIADAS ##

** ATLETAS REGISTRADOS **
 - ATLETA: Jenny Thompson (USA) - 31 años
  -- Disciplinas: NATACIÓN INDIVIDUAL
  -- Medallas:
   ---ORO en 100M MARIPOSA
   --*Total Medallas: 1 oro, 0 plata y 0 bronce
```

```
 - ATLETA: Laure Manaudou (FRANCIA) - 25 años
  -- Disciplinas: NATACIÓN INDIVIDUAL
  -- Medallas:
   --*Total Medallas: 0 oro, 0 plata y 0 bronce
 - ATLETA: Mireia Belmonte (USA) - 34 años
  -- Disciplinas: NATACIÓN INDIVIDUAL
  -- Medallas:
   --*Total Medallas: 0 oro, 0 plata y 0 bronce
 - ATLETA: Rafael Nadal (ESPAÑA) - 37 años
  -- Disciplinas: TENIS INDIVIDUAL, TENIS DOBLES
  -- Medallas:
   ---PLATA en DOBLES MIXTO
   ---ORO en TENIS INDIVIDUAL
   --*Total Medallas: 1 oro, 1 plata y 0 bronce
 - ATLETA: Novak Djokivik (SERBIA) - 36 años
  -- Disciplinas: TENIS INDIVIDUAL
  -- Medallas:
   --*Total Medallas: 0 oro, 0 plata y 0 bronce
 - ATLETA: Carlos Alcaraz (ESPAÑA) - 20 años
  -- Disciplinas: TENIS INDIVIDUAL, TENIS DOBLES
  -- Medallas:
   --*Total Medallas: 0 oro, 0 plata y 0 bronce
 - ATLETA: Jannik Sinner (ITALIA) - 22 años
  -- Disciplinas: TENIS INDIVIDUAL, TENIS DOBLES MIXTO
  -- Medallas:
   --*Total Medallas: 0 oro, 0 plata y 0 bronce

** DEPORTES Y EVENTOS **
 - DEPORTE: Natación - Lista de Eventos:
   -> EVENTO: 100m Libres Natación <24/07/2024>
   ---> Jenny Thompson (USA)
   ---> Laure Manaudou (FRANCIA)
   ---> Mireia Belmonte (USA)
 - DEPORTE: Tenis - Lista de Eventos:
   -> EVENTO: 1ª Semifinal Individual <25/07/2024>
   ---> Rafael Nadal (ESPAÑA)
   ---> Novak Djokivik (SERVIA)
   -> EVENTO: 2ª Semifinal Individual <25/07/2024>
   ---> Carlos Alcaraz (ESPAÑA)
   ---> Jannik Sinner (ITALIA)
   -> EVENTO: Final Individual <28/07/2024>
       <Aún no hay atletas registrados en este evento>
```

Clase SimulacionOlimpiadas. Clase principal que se proporciona implementada. Esta clase puede ser modificada para probar los diferentes aspectos desarrollados en el proyecto, pero es muy importante que se tenga en cuenta que la clase SimulacionOlimpiadas original debe funcionar con el desarrollo sin hacer ningún cambio en ella.

```
public class SimulacionOlimpiadas {
    public static void main(String[] args) {

        SistemaOlimpiadas sistema = new SistemaOlimpiadas(20, 5);

        // Crear algunos deportes
        Deporte natacion = new Deporte("Natación", 3);
        Deporte tenis = new Deporte("Tenis", 3);

        // Añadir deportes al sistema
        sistema.añadirDeporte(natacion);
        sistema.añadirDeporte(tenis);

        // Crear eventos
        Evento eventoTenis1 = new Evento("1ª Semifinal Individual", "25/07/2024", 10);
        Evento eventoTenis2 = new Evento("2ª Semifinal Individual", "25/07/2024", 10);
        Evento eventoTenis3 = new Evento("Final Individual", "28/07/2024", 10);
        Evento eventoNatacion = new Evento("100m Libres Natación", "24/07/2024", 10);

        // Añadir eventos a los deportes
        natacion.añadirEvento(eventoNatacion);
        tenis.añadirEvento(eventoTenis1);
```

```
        tenis.añadirEvento(eventoTenis2);
        tenis.añadirEvento(eventoTenis3);

        // Crear atletas
        Atleta atletaNat1 = new Atleta("Jenny Thompson", "USA", 31, "Natación individual");
        Atleta atletaNat2 = new Atleta("Laure Manaudou", "Francia", 25, "Natación individual");
        Atleta atletaNat3 = new Atleta("Mireia Belmonte", "USA", 34, "Natación individual");
        Atleta atletaTen1 = new Atleta("Rafael Nadal", "España", 37,
                                       "Tenis individual, Tenis dobles");
        Atleta atletaTen2 = new Atleta("Novak Djokivik", "Servia", 36, "Tenis individual");
        Atleta atletaTen3 = new Atleta("Carlos Alcaraz", "España", 20,
                                        "Tenis individual, Tenis dobles");
        Atleta atletaTen4 = new Atleta("Jannik Sinner", "Italia", 22,
                                        "Tenis individual, Tenis dobles mixto");

        // Registrar atletas en el sistema
        sistema.registrarAtleta(atletaNat1);
        sistema.registrarAtleta(atletaNat2);
        sistema.registrarAtleta(atletaNat3);
        sistema.registrarAtleta(atletaTen1);
        sistema.registrarAtleta(atletaTen2);
        sistema.registrarAtleta(atletaTen3);
        sistema.registrarAtleta(atletaTen4);

        // Añadir atletas a eventos
        eventoNatacion.añadirAtleta(atletaNat1);
        eventoNatacion.añadirAtleta(atletaNat2);
        eventoNatacion.añadirAtleta(atletaNat3);
        eventoTenis1.añadirAtleta(atletaTen1);
        eventoTenis1.añadirAtleta(atletaTen2);
        eventoTenis2.añadirAtleta(atletaTen3);
        eventoTenis2.añadirAtleta(atletaTen4);

        // Simular resultados y añadir medallas
        atletaNat1.añadirMedalla(new Medalla(Medalla.ORO, "100m Mariposa"));
        atletaTen1.añadirMedalla(new Medalla(Medalla.PLATA, "Dobles Mixto"));
        atletaTen1.añadirMedalla(new Medalla(Medalla.ORO, "Tenis Individual"));

        // Mostrar el sistema

        sistema.mostrarSistema();


    }
}
```

NOTAS IMPORTANTES

- No está permitido realizar el programa de forma conjunta. Se trata de un trabajo individual.
- El plagio de cualquier parte del código supondrá la anulación del trabajo y, por tanto, no podrá ser evaluado.
- Para que el trabajo pueda ser evaluado, es obligatorio que el proyecto compile (que no tenga errores de compilación) y ejecute (aunque tenga errores de ejecución). Si el programa no compila, no podrá ser evaluado.
- Se tendrá muy en cuenta el uso de la herencia y la reutilización de código.
- No se permite incluir comentarios en el código.

Diagrama de Clases