Objetivo general del taller

El objetivo general de este taller es dar una introducción a Java y al ambiente de desarrollo que se usará dentro del curso y asegurar que cada estudiante cuenta con un ambiente de trabajo en el que pueda desarrollar todas las actividades que requieran programar.

Objetivos específicos del taller

Durante el desarrollo de este taller se buscará que los estudiantes alcancen los siguientes objetivos:

1. Tener un ambiente de trabajo instalado y configurado en su computador con una distribución para desarrollo de Java instalada (OpenJDK) y un IDE (Eclipse).
2. Entender los elementos que hacen parte de una aplicación Java y un proyecto en Eclipse.
3. Poder compilar y ejecutar una aplicación Java desde Eclipse.
4. Modificar el código fuente de una aplicación Java y ver el resultado de la modificación.

Instrucciones generales

1. Descargue de Bloque Neón el archivo `Taller1-Olimpicos_esqueleto.zip` y descomprímalo en una carpeta dentro de su computador a la que pueda llegar con facilidad.
2. Lea con cuidado este documento y vaya realizando las actividades una por una. Lea el documento al tiempo que vaya resolviendo el taller: en este documento encontrará información importante para completar el programa mientras que va entendiendo lo que está haciendo.
3. No se preocupe si hay términos en este documento que no conozca o si no entiende algunos aspectos particulares del programa sobre el que va a trabajar. Este taller pretende dar sólo una primera mirada a Java e ir introduciendo la terminología relevante. Al final del semestre deberían haber quedado resueltas todas las dudas.

Este taller debe desarrollarse individualmente.

Parte 1: Instalación del ambiente de trabajo

Para realizar las actividades de este taller y del curso, usted necesita tener un ambiente de trabajo completo en su computador. Esto incluye los siguientes dos elementos principales:

- Una máquina virtual de Java (JVM), para poder ejecutar las aplicaciones. Sin embargo, no puede ser una versión SE, debe tener un JDK (Java Development Kit) para poder compilar las aplicaciones y tener disponibles las librerías básicas del lenguaje.
- Un ambiente de desarrollo (IDE) para poder editar las aplicaciones y realizar otras tareas relacionadas.

Para cubrir el primer punto existen varias alternativas, pero hay dos principales: Oracle JDK y Open JDK. La primera alternativa es la "oficial", pero está especialmente dirigida a empresas (de hecho, para poder aprovecharla al máximo se debería pagar una licencia). La segunda es la versión abierta y, aunque no es oficial, no tiene ninguna limitación real que nos interese a nosotros, así que es la que usaremos.

Con respecto a la versión, le recomendamos usar una de las últimas oficialmente disponibles y no una de las que están en versión *Early Access*.

El ambiente de desarrollo que vamos a utilizar es Eclipse, o más específicamente el paquete llamado "Eclipse IDE for Java Developers" (que muchas veces simplemente es llamado "Eclipse"). En realidad, Eclipse es una plataforma sobre la cual se pueden desplegar componentes basados en OSGI, que permiten desarrollar una infinidad de funciones sin tener que cambiar la plataforma. Por ejemplo, con los componentes adecuados es posible usar Eclipse para desarrollar aplicaciones en Java, C, C++, Python, PHP, entre otros. Otros componentes permiten también construir modelos y diagramas, e incluso hay juegos (sencillos) desarrollados sobre esta plataforma.

Existen alternativas para Eclipse, como IntelliJ, pero ninguna tiene el mismo balance tan positivo entre funcionalidades y costo de licencia (en IntelliJ las licencias son gratuitas para cosas académicas, pero no lo son si luego van a usar esa herramienta para algún asunto profesional). Finalmente, todos los esqueletos de proyectos que nosotros les demos durante el semestre van a estar configurados para Eclipse. Se podrían portar a otras herramientas, pero eso sería su responsabilidad y además ustedes tendrían que portarlos de regreso a Eclipse para hacer la entrega.

Actividades

1. Descargue e instale una versión de desarrollo de Java en su computador. Puede visitar este sitio web para encontrar los archivos de instalación: https://jdk.java.net/ Asegúrese de descargar un JDK.
2. Abra la línea de comandos de su equipo (cmd en Windows o la terminal de Linux o Mac OS) y ejecute el siguiente comando para asegurar que todo esté correctamente configurado:

   ```
   javac –version
   ```

   Si todo está bien, usted debería ver un mensaje con el número de la versión que tiene instalada. Si no funciona (le dice que el comando `javac` no existe), revise por qué no quedaron bien configuradas las rutas de ejecución (el *PATH* del sistema).
3. Descargue e instale Eclipse en su computador. Puede encontrar los últimos instaladores en https://www.eclipse.org/downloads/ .
4. Ejecute Eclipse desde donde haya quedado instalado.
5. Eclipse va a preguntarle en qué carpeta quiere tener el *workspace* o espacio de trabajo. Seleccione una nueva carpeta que pueda ubicar con facilidad, para dejar ahí todos los proyectos en los que trabaje con Eclipse. Le recomendamos utilizar siempre el mismo *workspace* porque ahí también se almacenarán la configuración personalizada que usted haga del IDE (por ejemplo, los tipos de letra, los estándares para organizar el código, etc.).

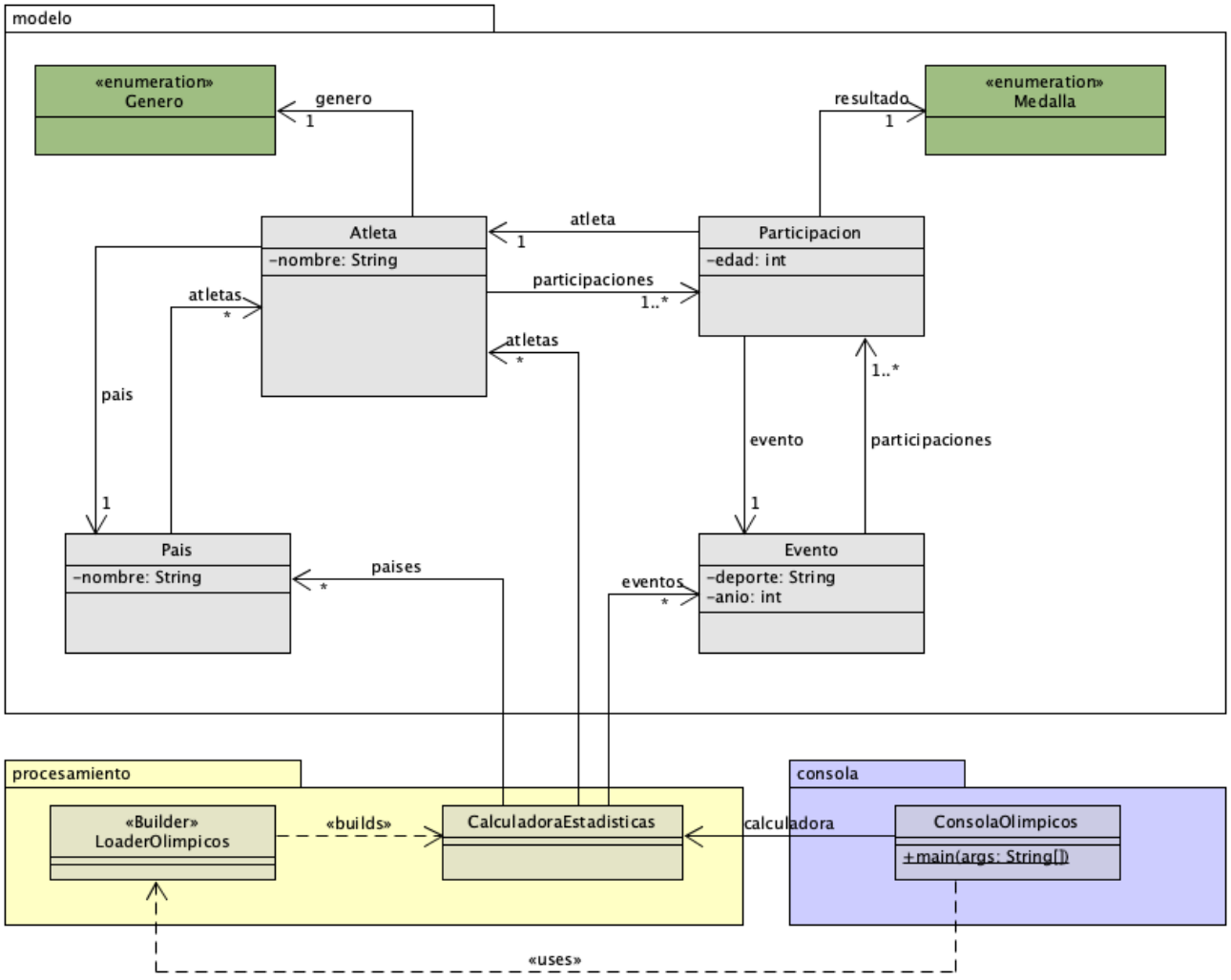
Parte 2: Caso de estudio

En esta parte del taller vamos a cargar en Eclipse un proyecto existente construido con Java y estudiar algunos de sus componentes principales.

Descripción del caso de estudio: Juegos Olímpicos

El caso de estudio que vamos a utilizar para esta parte del taller está basado en el proyecto de nivel 3 del curso de Introducción a la Programación durante el semetres 2020-10. Si usted hizo parte de ese curso, es posible que reconozca que la aplicación que vamos a estudiar es equivalente funcionalmente - hace lo mismo – pero tiene una estructura muy diferente para poder aprovechar las ventajas de la programación orientada a objetos.

El siguiente diagrama muestra todas las clases, enumeraciones y paquetes que hacen parte del proyecto. En este diagrama puede verse que hay tres paquetes en la aplicación (consola, procesamiento y modelo) y que cada uno de estos paquetes contiene una o varias clases. En el diagrama también pueden verse las relaciones entre las clases: una flecha indica que una clase (de donde sale la flecha) *conoce* a otra clase (a donde llega la flecha). Estas relaciones las estudiaremos con mucho detalle durante el curso. Cada uno de las clases y enumeraciones que aparecen en este diagrama corresponde a un archivo java dentro del proyecto. Cada uno de los paquetes corresponde a una carpeta.



En el paquete `modelo` se encuentran las clases que representan el problema que maneja la aplicación: tenemos la información de los atletas que participaron en diferentes eventos de uno o varios juegos olímpicos, representaron a un país, y podrían haber ganado o no medallas. Cada atleta tiene un nombre, un género y además sabemos a qué país representó. Un evento corresponde a una competencia de un deporte y se hizo en un año específico (ej.

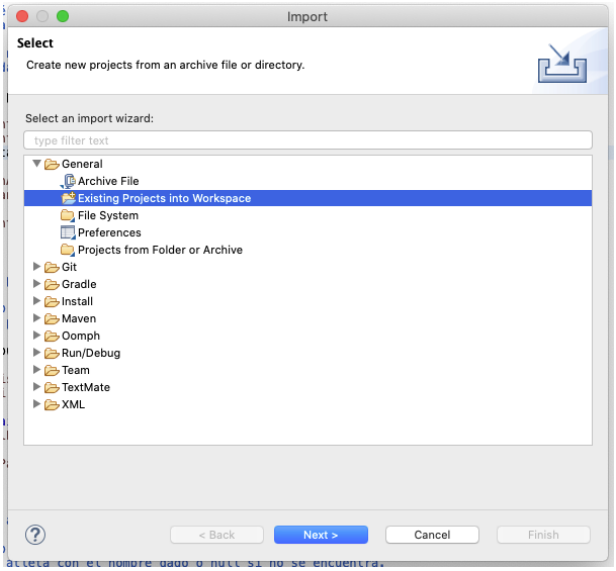
100 metros planos en 2012). Una participación indica que un atleta compitió en un determinado evento, cuando tenía una cierta edad y obtuvo o no una medalla como resultado.

En el paquete `procesamiento` hay dos clases: la clase `LoaderOlimpicos` que se encarga de cargar la información de los atletas a partir de un archivo CSV, y la clase `CalculadoraEstadisticas` que se encarga de cálcular estadísticas y hacer búsquedas sobre la información de los atletas.

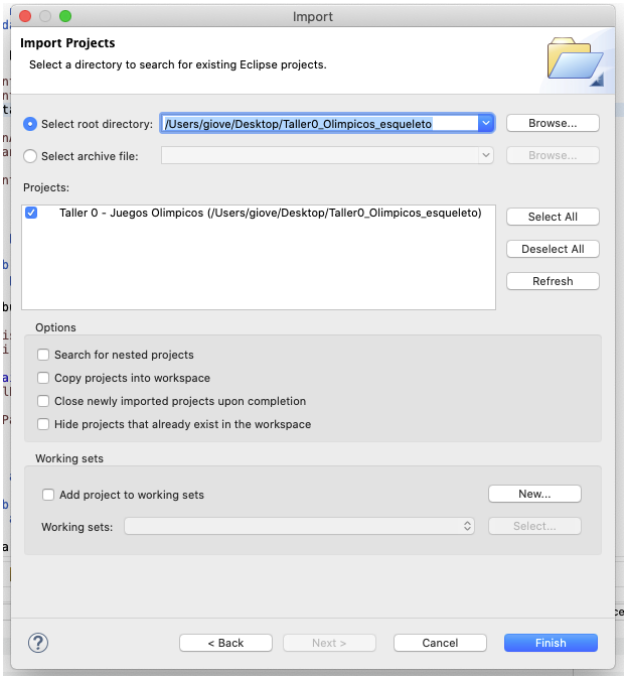
En el paquete `consola` se encuentra una única clase que tiene toda la lógica relacionada con la consola del programa.

Cargar el proyecto en Eclipse

En Eclipse, escoja la opción para importar algo a su espacio de trabajo (File – Import) e indique que quiere importar un proyecto existente.



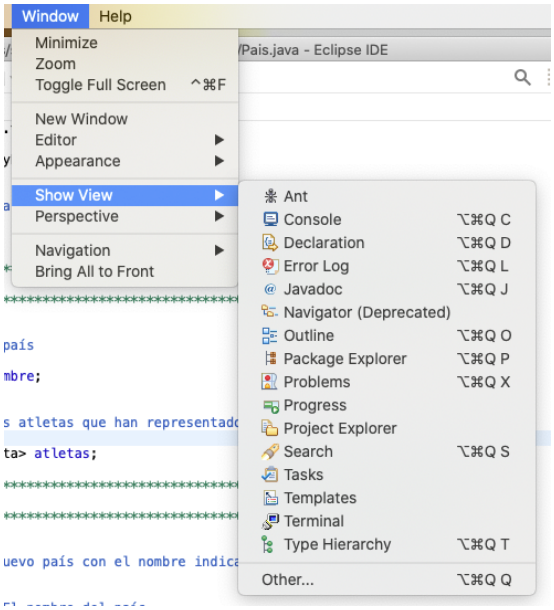
Seleccione la carpeta donde descomprimió el archivo `Taller1-Olimpicos_esqueleto.zip` y seleccione el proyecto para importar.



Si selecciona la opción “Copy projects into workspace”, los archivos se copiarán al espacio de trabajo de Eclipse para que s utilicen desde allí. En general es mejor tener todos los proyectos de Eclipse dentro del workspace.

Observe ahora la ventana de Eclipse: debería ser similar a la siguiente captura de pantalla e incluir las siguientes vistas predeterminadas.

En Eclipse, cada uno de los espacios del editor se denomina una vista (view). A través del menú Window – Show View usted puede seleccionar qué vistas quiere tener disponibles en cualquier momento.



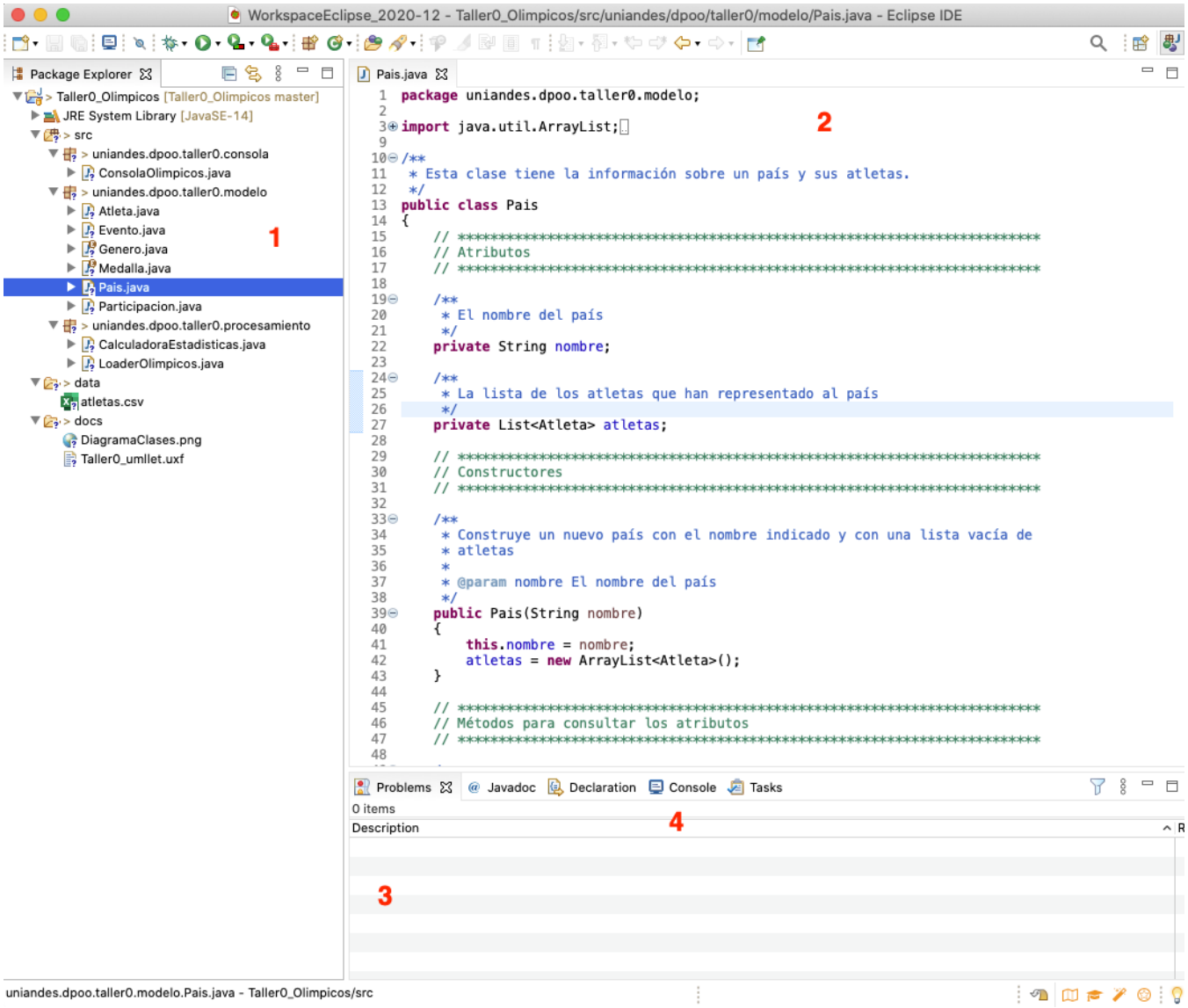
Si no ha hecho ningún cambio, usted debería estar viendo en este momento una ventana similar a la siguiente en la cual aparecen las vistas predeterminadas:

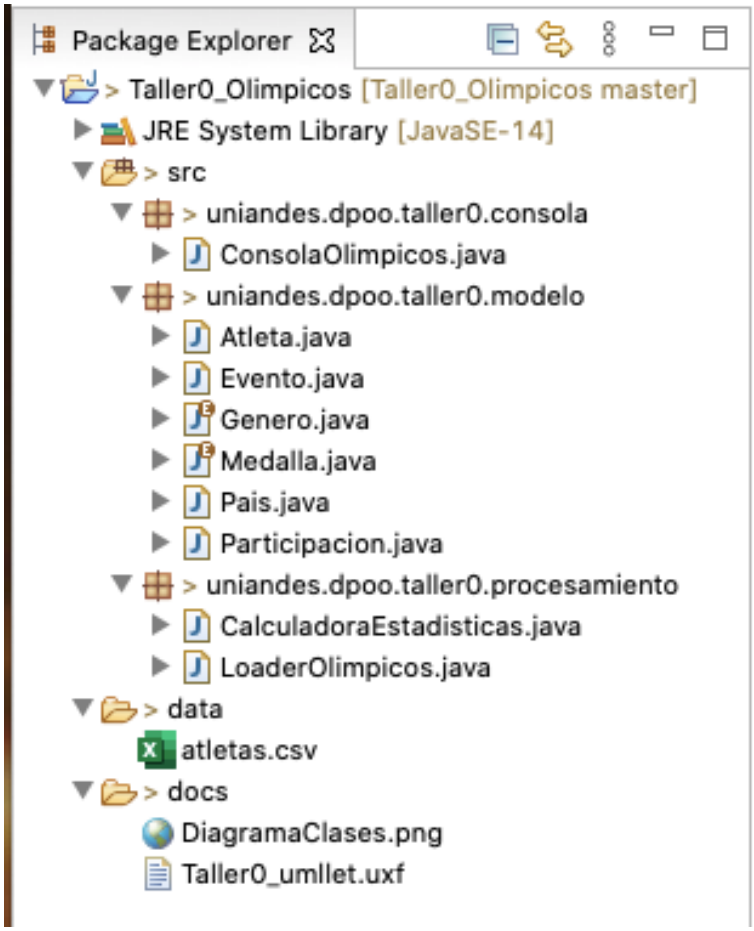
1. Esta vista se denomina el Package Explorer y es el espacio en el cual se puede ver el contenido de los proyectos que actualmente están cargados en el workspace. Cada proyecto tiene su propia estructura interna que estudiaremos a continuación.

2. Esta vista es un editor de Java. En este caso, estamos viendo el contenido del archivo Pais.java.

3. En la vista de problemas aparece la información de todos los **Errores** y **Warnings** de compilación que haya detectado Eclipse sobre todos los proyectos que estén abiertos en un momento dado. Los errores son problemas que impiden que un proyecto se compile y por tanto tienen que ser solucionados antes de continuar. Los warnings o alertas son problemas que no son serios y que no impiden la compilación. Sin embargo, Eclipse nos avisa de estos problemas potenciales para que los solucionemos lo más pronto posible.

4. En la captura de pantalla no se puede ver porque la vista de problemas está por delante, pero también existe una vista de Consola, en la cual se puede ver el resultado de la ejecución de un programa y se puede ingresar información si es necesario.





Estudiar la estructura del proyecto

La estructura del proyecto que se ve en el explorador de paquetes permite identificar los principales elementos que hacen parte del proyecto. A continuación, explicamos qué significa cada uno de ellos.

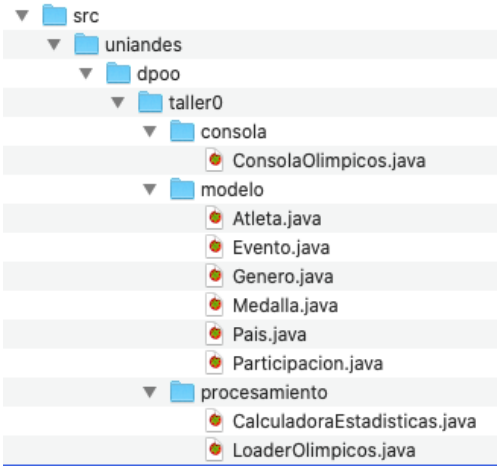
1. Taller0_Olimpicos: este es el nivel superior y corresponde al proyecto entero. La parte que aparece entre [y] en la captura de pantalla probablemente no le aparezca a usted porque su proyecto no debe estar siendo versionado con GIT.
2. JRE System Library: esto muestra cuál es la versión del JDK que se está usando para construir y correr la aplicación.
3. src: el ícono de esta carpeta nos muestra que esta carpeta está configurada para que su contenido sea compilado (note que el ícono es diferente al de las carpetas `data` y `docs` que se encuentran más abajo. Es decir que la carpeta `src` hace parte de lo que Eclipse llama "Build path": todos los elementos que se utilizan durante el proceso de compilación.
4. `uniandes.dpoo.taller0.consola`: el ícono de esta carpeta también es diferente y nos indica que es un paquete, es decir una unidad de Java dentro de la cual se encuentran clases. El nombre completo del paquete es el que aparece en esta carpeta, pero en el diagrama de clases que mostramos anteriormente sólo incluimos la última parte (consola) para hacer que el diagrama fuera un poco más sencillo.
5. ConsolaOlimpicos.java: este es un archivo y tanto el ícono como la extensión nos indican que se trata de un archivo de Java. Lo normal es que dentro de este archivo encontremos una clase llamada `ConsolaOlimpicos`.
6. A continuación, encontraremos otros dos paquetes (modelo y procesamiento) con los archivos java que cada uno contiene. Note que esta estructura corresponde a la del diagrama de clases que vimos antes.
7. data: en esta carpeta tenemos la información de los juegos olímpicos que vamos a cargar (el archivo `atletas.csv`).
8. docs: en esta carpeta tenemos documentación adicional sobre la aplicación.

Ahora bien, si desplegamos el contenido de uno de los archivos .java, podremos ver la estructura que tiene por dentro. Veamos, a modo de ejemplo, el archivo Pais.java

En esta estructura vemos que dentro del archivo Pais.java hay una clase llamada Pais (está marcada con una C verde) y que esa clase tiene algunos atributos (marcados con cuadros rojos) y métodos (marcados con círculos verdes). El color indica la *visibilidad*: los elementos públicos son verdes mientras que los privados son rojos. Más adelante estudiaremos en detalle lo que esto significa. En el caso de los métodos, podemos ver los tipos de los parámetros y los tipos de los resultados de cada uno (`void` significa que el método no tiene un retorno).

Nota importante: El nombre de la clase es Pais y el nombre del archivo es Pais.java. Esto no es una casualidad. Una clase pública **DEBE** declararse en un archivo que tenga exactamente el mismo nombre de la clase y la extensión .java. Una consecuencia de esto es que no se pueda tener un archivo en el cual se declaren dos clases públicas.

Finalmente, abra el Explorador de Archivos, Finder o una herramienta equivalente en su computador para observar la estructura de la carpeta src de su proyecto. Debería encontrarse con algo similar a lo siguiente:



Esto muestra que los paquetes en Java son equivalentes a las carpetas y que una clase se encuentra en un paquete dependiendo de dónde esté ubicado el archivo dentro de esta estructura.

Estudiar la estructura de un archivo Java: Pais.java

Estudiemos ahora los elementos que hacen parte de un archivo Java para identificar sus principales componentes. Note que los archivos del proyecto están documentados de forma extensiva para que usted entienda con facilidad el rol de cada elemento.

Estudiaremos el archivo Pais.java, descomponiéndolo para ir explicando algunos de los bloques de instrucciones que lo componen.

```
package uniandes.dpoo.taller0.modelo;
```

Esta línea declara que el archivo Pais.java está ubicado dentro de la carpeta uniandes/dpoo/taller0/modelo. Note que los nombres de las carpetas se separan usando el carácter '.'

¡Tenga cuidado con los nombres de carpetas! El nombre de un paquete no puede tener ni espacios ni casi ningún símbolo, así que limítese a letras y números.

¡Tenga cuidado con los Punto y Coma! Este caracter se usa en Java para terminar cualquier instrucción, así que lo encontraremos al final de casi cualquier línea.

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.HashSet;
import java.util.List;
import java.util.Map;
import java.util.Set;
```

Estas líneas declaran que en este archivo se utilizarán todas estas clases definidas dentro del paquete '`java.util`'. El mecanismo de importar una clase es muy similar al mecanismo de Python.

```
/**
 * Esta clase tiene la información sobre un país y sus atletas.
 */
```

Estas líneas sólo tienen un comentario, que en este caso describe el objetivo de la clase Pais. En Java, un comentario largo puede incluirse usando las cadenas `/*` y `*/` para delimitar de dónde a dónde debe considerarse que va el comentario.

```
public class Pais
{
```

Estas líneas inician la declaración de la clase Pais, la cual termina al final del archivo con el carácter { . Es importante notar que, en este caso, como en la mayoría, indicamos que la clase Pais va a ser pública.

En Java, todos los bloques de instrucciones se delimitan con la pareja de caracteres { y }, por lo cual la **indentación** ni es obligatoria ni significa nada. La estructura que en Python se construye a través de la indentación, en Java se logra a través del uso de llaves. Sin embargo, es una muy buena práctica indentar el código para facilitar su lectura.

```
        // ***************************************************************************
        // Atributos
        // ***************************************************************************
```

```
Estas líneas ilustran la otra forma que ofrece Java para introducir un comentario en un programa: con
los caracteres // se indica que el resto de la línea es un comentario.
```

```
        /**
         * El nombre del país
         */
        private String nombre;

        /**
         * La lista de los atletas que han representado al país
         */
        private List<Atleta> atletas;
```

Estas líneas muestran la definición de dos atributos en la clase Pais llamados `nombre` y `atletas`. Acá debemos resaltar, en primer lugar, que los atributos se definen como privados. Esto es muy importante en relación con el concepto de encapsulamiento que estudiaremos más adelante.

En segundo lugar, observe que estamos especificando el tipo de cada uno de estos atributos: el nombre de un país decimos que es un `String`, es decir una cadena de caracteres (note la S mayúscula); en cambio decimos que `atletas` es una lista de elementos de tipo `Atleta`. Si miramos con atención, veremos que `List` es el nombre de algo que importamos del paquete `java.util` al principio del archivo. `Atleta` no la encontramos importada al inicio, pero como se encuentra en el mismo paquete que nuestra clase `Pais`, podemos utilizarla sin problema.

```
        /**
         * Construye un nuevo país con el nombre indicado y con una lista vacía de
         * atletas
         *
         * @param nombre El nombre del país
         */
        public Pais(String nombre)
        {
                this.nombre = nombre;
                atletas = new ArrayList<Atleta>();
        }
```

Este bloque de instrucciones corresponde a la declaración, implementación y documentación del método constructor de la clase `Pais`. Más abajo encontraremos otros métodos, pero este es especial porque es el que permite construir instancias de un `Pais`. Es decir que, cuando queramos construir un nuevo Pais, tendremos que invocar este método. Sabemos que este método es un método constructor por dos razones: se llama igual que la clase y no indica el tipo de retorno.

La implementación de este método se encuentra entre los caracteres { y } y en este caso se compone de dos instrucciones.

La documentación de este método se encuentra justo antes del método y tiene una estructura que estudiaremos más adelante.

```
        /**
         * Consulta el nombre del país
         *
         * @return nombre
         */
        public String darNombre()
        {
                return nombre;
        }
```

Ahora encontramos la declaración, implementación y documentación del método `darNombre`. A diferencia del método anterior, este método no es un método constructor (no se puede usar para crear nuevos países) y en cambio debe utilizarse para preguntarle el nombre a un país. La signatura del método (`public String darNombre()`) nos indica varias coas importantes:

- El método es público, para que pueda ser invocado desde otras clases.
- El método retorna algo de tipo `String`.
- El método se llama darNombre.
- El método no espera ningún parámetro.

```
/**
 * Consulta cuáles han sido los medallistas del país de un determinado género.
 *
 * @param generoAtleta El género de interés.
 * @return Retorna un mapa donde las llaves son los nombres de los atletas del
 *         país y del género que han sido medallistas y los valores son una
```

```
 *          lista con información de sus medallas. La información de cada medalla
 *          también es un mapa que tiene tres llaves: "evento", que tiene
 *          asociado el nombre del evento; "anio", que tiene asociado el año en
 *          el que el atleta ganó la medalla; y "medalla" que tiene asociado el
 *          tipo de medalla.
 */
public Map<String, List<Map<String, Object>>> consultarMedallistasGenero(Genero generoAtleta)
{
  Map<String, List<Map<String, Object>>> resultado = new HashMap<String, List<Map<String, Object>>>();

  for (Atleta atleta : atletas)
  {
       if (atleta.darGenero().equals(generoAtleta) && atleta.esMedallista())
       {
           List<Map<String, Object>> medallasAtleta = atleta.consultarMedallas();
           resultado.put(atleta.darNombre(), medallasAtleta);
       }
  }
  return resultado;
}
```

Finalmente encontramos el método `consultarMetallistasGenero`, el cual tiene varios elementos muy interesantes que estudiaremos a lo largo del curso:

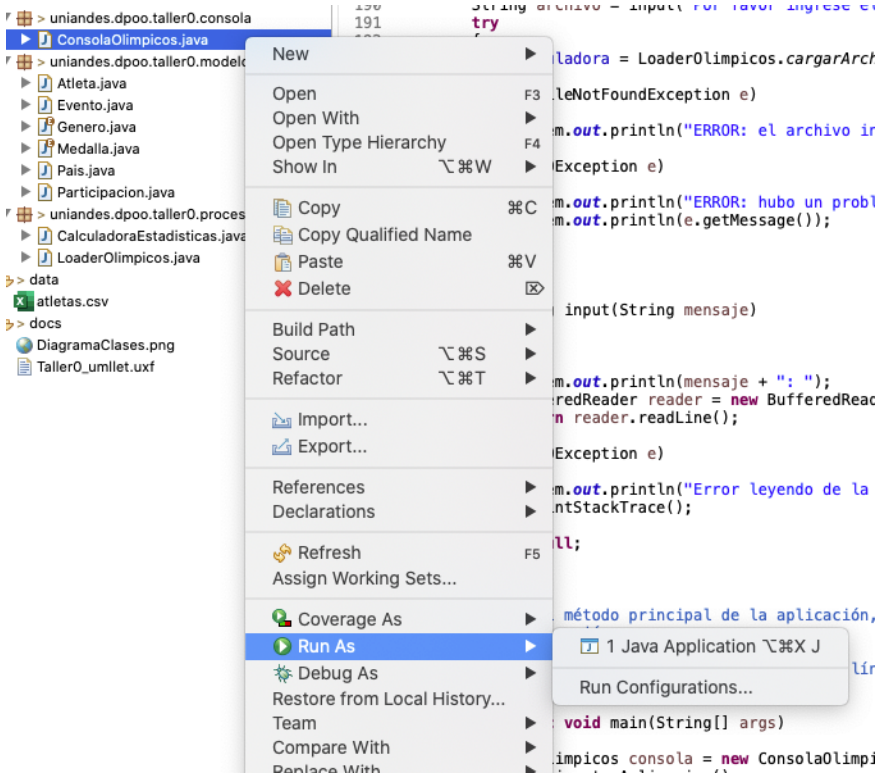
- Este método retorna un mapa (Map, el equivalente de los diccionarios de Python) cuyas llaves son cadenas de caracteres y cuyos valores son listas. A su vez, estas listas contienen mapas cuyas llaves son cadenas de caracteres y sus valores son objetos.
- La primera instrucción del método crea una variable llamada `resultado`, del tipo que debe retornar el método.
- La variable `resultado` se inicializa con un nuevo elemento de tipo `HashMap`: la palabra **`new`** nos indica que estamos invocando el método constuctor de la clase `HashMap`. Más adelante veremos por qué acá usamos `HashMap` y no simplemente `Map`.
- Las siguientes líneas implementan un ciclo sobre el atributo `atletas` de un país. A este tipo de ciclo se le conoce en Java como un for-each, porque está recorriendo cada elemento de `atletas` y lo está guardando en la variable `atleta`. Esta es la estructura más cercana en Java a la estructura `for-in` de Python.
- Dentro del ciclo, encontramos un condicional: tiene la misma estructura que en Python, pero la sintaxis es ligeramente diferente (no hay :, los paréntesis son obligatorios, el cuerpo del condicional se delimita con llaves).

Correr la aplicación

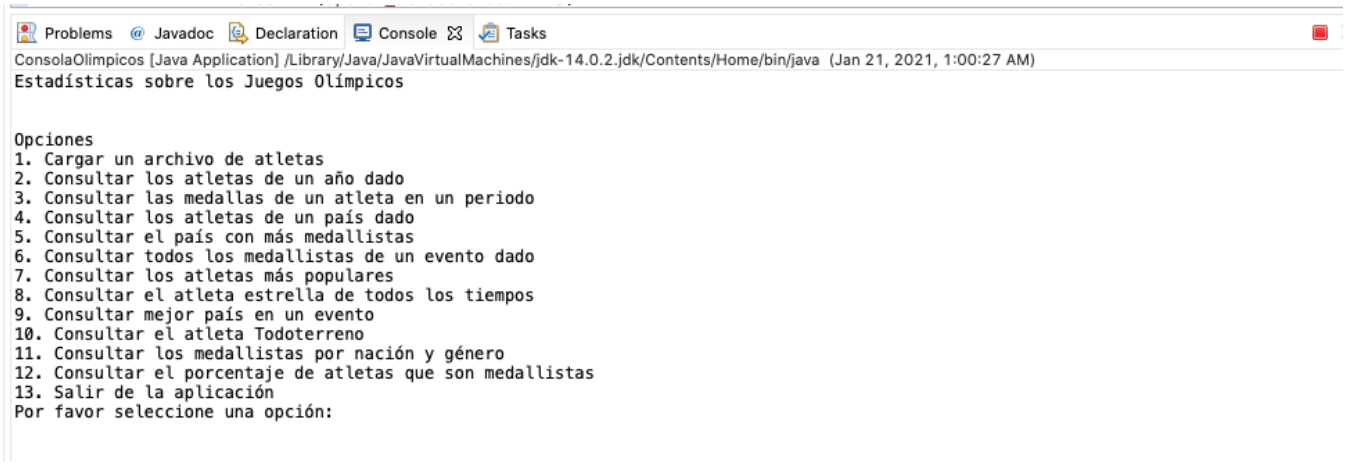
Ubique el archivo con la clase ConsolaOlimpicos y ábralo. Al final del archivo encontrará un método con la signatura:

```
public static void main(String[] args)
```

Esta es la signatura que **DEBE** tener el método `main` de una aplicación Java, es decir el método que se puede utilizar para lanzar una aplicación. Ahora haga click derecho sobre el archivo que contiene la clase y seleccione la opción para correrlo como una aplicación Java.



La aplicación debería empezar a correr y mostrar los siguientes mensajes en la consola:



Pruebe algunas de las opciones y finalmente seleccione la opción 13 para salir, o haga click en el cuadro rojo de la parte superior derecha de la consola para terminar la ejecución del programa.

Note que si hace click derecho sobre una clase que no tenga un método `main`, no le aparecerá la opción para correr la clase como una aplicación Java.

Puede volver a correr la clase ConsolaOlimpicos haciendo click sobre el botón que se encuentra en el menú superior, el cual correrá el último programa ejecutado.

Parte 3: Modificar el programa

En esta parte ahora usted tendrá que modificar el programa utilizando como base los programas que ya tiene.

Modificación 1: Agregar una nueva opción

En este caso usted debe agregar una opción adicional al menú de la aplicación, la cual debe hacer lo siguiente:

1. Preguntarle al usuario el nombre de un atleta.
2. Buscar el atleta con ese nombre.
3. Informarle al usuario el nombre del país al cual representa el atleta.

Para lograr esto, usted debe:

1. Modificar la clase `ConsolaOlimpicos`. Siga la guía de las opciones que ya existen para agregar su opción. No olvide modificar el menú que se le muestra al usuario.
2. Modificar la clase `CalculadoraEstadisticas`. También acá le servirá guiarse por los métodos que ya existen. Recuerde que, al igual que en Python, dentro de un método puede llamar a métodos que ya existan dentro de la misma clase.

Modificación 2: Agregar un nuevo programa

En este caso usted creará un nuevo programa en el que usará los elementos que ya están implementadas en las clases del esqueleto.

1. Agregue un nuevo paquete al proyecto llamado `uniandes.dpoo.taller0.modificacion`.
2. Dentro del nuevo paquete agregue una clase llamada `Modificacion`.
3. Agregue un método `main` a la nueva clase, el cual debe tener únicamente el siguiente contenido:

```
System.out.println(“Hola, mundo!”);
```

4. Ejecute su nueva clase y asegúrese de que en la consola aparece el mensaje.
5. Modifique su método `main` para que cargue la información de los atletas y luego imprima el nombre del país con más medallistas. Para eso puede utilizar las siguientes dos instrucciones dentro de la implementación del método:

```
CalculadoraEstadisticas calc = LoaderOlimpicos.cargarArchivo("./data/atletas.csv");
System.out.println(calc.paisConMasMedallistas());
```

6. Si se produce un error, analice el mensaje de error que aparece en la vista “Problems”. Revise las opciones de corrección automática que le da Eclipse: corrija el error.
7. Cuando no aparezcan errores en la vista de problemas (no aparece nada marcado en rojo dentro de los archivos .java), ejecute la aplicación.

Entrega

GitHub es una plataforma en la nube que permite administrar un proyecto de software, bien sea que se trabaje de forma individual o en grupo. En este curso, las entregas de talleres y proyectos deben realizarse a través de esta plataforma.

Para continuar con el laboratorio deben tener una cuenta en GitHub, en caso de no poseerla deben registrarse **con su correo de Uniandes** en la plataforma utilizando el enlace `https://github.com/join`. Si ya tienen una

cuenta personal con otro correo, debe crear una nueva cuenta con el correo de la Universidad o cambiar el correo de registro de que ya existe.

Construya un repositorio **público** nuevo para el laboratorio y deje en este todo el trabajo que haya realizado para el taller.

Entregue a través de Bloque Neón el URL del repositorio en la actividad designada como "**Taller 1**".