DEPARTAMENTO LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN	**PROGRAMACIÓN ORIENTADA A OBJETOS** (Prueba realizada el 18 de junio de 2012)	
APELLIDOS, Nombre	TITULACIÓN Y GRUPO	
	MÁQUINA	

NOTAS PARA LA REALIZACIÓN DEL EJERCICIO:

- El ejercicio se almacenará en el directorio **C:\POO**. En caso de que no exista deberá crearse, y si ya existiese, deberá borrarse todo su contenido antes de comenzar.
- Al inicio del contenido de cada fichero deberá indicarse **el nombre del alumno, titulación, grupo y código del equipo** que está utilizando**.**
- La evaluación tendrá en cuenta la claridad de los algoritmos, del código y la correcta elección de las estructuras de datos, así como los criterios de diseño que favorezcan la reutilización.
- Los diferentes apartados tienen una determinada puntuación. Si un apartado no se sabe hacer, el alumno no debe pararse en él indefinidamente. Puede abordar otros.
- **Está permitido:**
  - Consultar el API.
- **No está permitido**:
  - Utilizar otra documentación electrónica o impresa.
  - Intercambiar documentación con otros compañeros.
  - Utilizar soportes de almacenamiento.
- Una vez terminado el ejercicio subir un fichero comprimido (.rar) sólo con los **fuentes** que hayáis realizado a la tarea creada en el campus virtual para ello.

Se desea llevar a cabo la gestión informatizada de la selección del equipo de fútbol ideal a partir de las votaciones y valoraciones recibidas por los diferentes equipos y jugadores participantes en la Eurocopa de selecciones que se está celebrando en Polonia y Ucrania. Para ello, se creará un proyecto `prEquipoIdeal` con las clases siguientes:

1) (***0.25 ptos.***) La clase `EurocopaException` se utilizará para tratar diferentes situaciones excepcionales tal y como se indica más adelante. Se trata de una excepción no comprobada.

2) (***1.5 ptos.***) La clase `Jugador` debe mantener información sobre un jugador. En concreto, su nombre (`String`), su demarcación (`String`) (las posibles demarcaciones son: "PORTERIA", "DEFENSA", "CENTRO", "ATAQUE"), una valoración (`int`) obtenida por el jugador a través de las puntuaciones realizadas por los comentaristas deportivos acreditados, y una indicación (`boolean`) de si ha sido elegido o no para formar parte del equipo ideal. También dispondrá del constructor y los métodos necesarios para:
   a. *(0.25 ptos.)* Construir un objeto de la clase a partir del nombre, demarcación y valoración del jugador. La representación de un jugador (`String toString()`) viene dada por esos tres datos, separados por espacios en blanco.
   b. *(0.25 ptos.)* Obtener el nombre (`String getNombre()`), obtener la demarcación (`String getDemarcacion()`), ver si ha sido elegido para el equipo ideal (`boolean getElegido()`) y elegir el jugador para el equipo ideal (`void setElegido()`).
   c. *(0.5 ptos.)* Dos objetos de la clase `Jugador` son iguales si coinciden sus demarcaciones, sus valoraciones y sus nombres. No se tendrán en cuenta las diferencias por mayúsculas o minúsculas.
   d. *(0.5 ptos.)* Un objeto de la clase `Jugador` j1 es menor que otro j2 cuando la demarcación de j1 es mayor alfabéticamente (independientemente de minúsculas y mayúsculas) que la demarcación de j2. En caso de que las demarcaciones sean iguales, el jugador j1 es menor que el j2 cuando la valoración de j1 es mayor que la valoración de j2. En caso de que las valoraciones sean iguales, se considerará menor el jugador cuyo nombre alfabéticamente sea menor.

3) (***1 pto.***) La clase `Equipo` debe mantener información sobre el nombre de un equipo de fútbol (`String`) y el número de votos (`int`) obtenidos a través de mensajes sms enviados por los aficionados. También dispondrá del constructor y los métodos necesarios para:
   a. *(0.25 ptos.)* Construir un objeto de la clase a partir del nombre y los votos del equipo. La representación de un equipo (`String toString()`) viene dada por esos dos datos, separados por un espacio en blanco.
   b. *(0.25 ptos.)* Obtener y establecer los votos (`int getVotos()` y `void setVotos(int votos)`).
   c. *(0.25 ptos.)* Dos objetos de la clase `Equipo` son iguales si coinciden sus nombres (independientemente de que estén en mayúsculas o minúsculas) y sus votos.
   d. *(0.25 ptos.)* Un objeto de la clase `Equipo` e1 es menor que otro e2 cuando el número de votos de e1 es mayor que el de e2. En caso de que el número de votos sea igual, se considerará menor el equipo cuyo nombre alfabéticamente sea menor.

4) (**3.5 *ptos.***) La clase `Eurocopa` almacenará la información de los equipos y jugadores que han obtenido votos y valoraciones en una estructura `puntuaciones` de tipo `SortedMap<Equipo,List<Jugador>>`. También dispondrá del constructor y los métodos necesarios para:

   a. *(1.5 ptos.)* Construir un objeto de la clase a partir del nombre del fichero (`String`) que contiene la información necesaria para crear la aplicación `puntuaciones`. El formato de los datos es el que aparece en el fichero `equipos.txt` proporcionado en el campus virtual. Cualquier error de formato detectado (falta algún dato de equipo o jugador, o bien el dato no es del tipo correcto), provocará el lanzamiento de una excepción del tipo `EurocopaException`.
   b. *(1.5 ptos.)* Formar el equipo ideal (`SortedSet<Jugador> formarEquipoIdeal(int defensa, int centro, int ataque)`. Este método recibe como parámetros el número de defensas, centrocampistas y atacantes requeridos para construir un conjunto de jugadores que devolverá como resultado (también se seleccionará un portero). Los jugadores se almacenarán en este conjunto ordenados atendiendo a su orden natural. Si el número de jugadores disponibles en cada demarcación es menor que el exigido, se seleccionarán todos los posibles. El proceso para seleccionar a los jugadores y almacenarlos en el conjunto resultado es el siguiente (para cada demarcación): se selecciona el jugador mejor valorado de esa demarcación del equipo más votado. Si hay varios jugadores con la misma valoración máxima en un equipo dado, se elegirá el jugador cuyo nombre alfabéticamente sea menor. El jugador se marcará como elegido. Después se pasa al siguiente equipo más votado para seleccionar otro jugador de su lista, y así sucesivamente, hasta seleccionar todos los exigidos (o bien determinar que ya no hay más jugadores disponibles para seleccionar). Cuando se termina de tratar el equipo menos votado, de nuevo se comienza por el más votado.
      Para ello se deben construir dos métodos auxiliares: `SortedSet<Jugador> seleccionar(int numJugadores, String demarcacion)` para recorrer los diferentes equipos y `Jugador seleccionarJugador(Equipo eq, String demarcacion)` para recorrer la lista de jugadores de un determinado equipo. Hay que tener en cuenta que el método `seleccionarJugador` devolverá

`null` si no hay ningún jugador posible para seleccionar en el equipo dado. Por su parte, el método `seleccionar` devolverá un conjunto vacío si no ha encontrado jugadores para seleccionar entre los diferentes equipos.

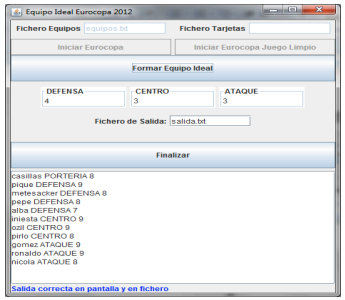
c. *(0.5 ptos.)* Mostrar el equipo ideal. Dos métodos estáticos, uno para mostrar el equipo ideal por pantalla (`static void mostrarEquipoIdeal (SortedSet<Jugador> ideal)`) y otro para almacenarlo en un fichero (`static void mostrarEquipoIdeal(String nombreFichero, SortedSet<Jugador> ideal)`). El formato de salida es como el mostrado en el fichero `equipoIdeal.txt` proporcionado en el campus virtual.

5) (***1.75 ptos.***) La clase `EurocopaJuegoLimpio` se comporta como la clase `Eurocopa`. La diferencia radica en que a la hora de formar el equipo ideal, no se tendrán en cuenta aquellos jugadores que hayan recibido un número determinado de tarjetas. Para ello, esta clase almacenará en la lista `jugadoresNoPermitidos` (de tipo `List<String>`) los nombres de los jugadores no permitidos. Estos nombres se leerán de un fichero. Dispondrá del constructor y el método necesarios para:

a. *(1 ptos.)* Construir un objeto de la clase a partir del nombre del fichero (`String`) que contiene la información necesaria para crear la aplicación `puntuaciones` y el nombre del fichero (`String`) que contiene los nombres de los jugadores que han recibido tarjetas (el formato de los datos es el que aparece en el fichero `tarjetas.txt` proporcionado en el campus virtual).

b. *(0.75 ptos.)* Una nueva implementación del método `Jugador seleccionarJugador(Equipo eq, String demarcacion)` para no considerar como seleccionables aquellos jugadores que estén en la lista `jugadoresNoPermitidos.`

6) (***2 ptos.***) La clase `ControladorEurocopa` controla e interactúa con el modelo (compuesto por las clases anteriormente descritas) y la vista (se proporcionan en el campus virtual la interfaz `VistaEurocopa` y la clase `PanelEurocopa`). El constructor debe habilitar la parte de inicialización de la vista (introducción de ficheros de equipos y tarjetas, y los botones de iniciar Eurocopa) y mostrar un mensaje en la parte baja de la misma indicando al usuario que introduzca nombre de ficheros y pulse el botón iniciar deseado. El resto de la vista estará deshabilitado. La pulsación de alguno de los botones iniciar hará que se cree un objeto de la clase `Eurocopa` o de la clase `EurocopaJuegoLimpio`, se deshabilite la zona de inicialización, se habilite el resto de la vista y se muestre un mensaje al usuario invitándolo a formar el equipo ideal. Cada vez que el usuario pulse el botón "Formar Equipo Ideal" se procederá a formar el equipo ideal con los jugadores disponibles. El resultado se mostrará en el área de texto y se escribirá en el fichero de salida especificado. El botón "Finalizar" deshabilita esta zona de proceso, limpia las zonas de texto y habilita la parte de inicialización de la vista para poder empezar de nuevo con otros ficheros de entrada. Si al pulsar un determinado botón, no se han introducido los datos necesarios en los campos de texto, se debe mostrar un mensaje de error en la parte baja de la vista (para ello se deben capturar excepciones de tipo `IOException` y `RuntimeException`)



La siguiente clase principal muestra un ejemplo del uso de las clases anteriormente descritas (SIN USO DE GUIs). Está disponible en el campus virtual.

```
import java.io.IOException;
import java.util.SortedSet;

public class Principal {
	public static void main(String[] args) {
		try {
			Eurocopa e = new Eurocopa("equipos.txt");

			// primer equipo ideal
			SortedSet<Jugador> ideal1 = e.formarEquipoIdeal(4,3,3);
			Eurocopa.mostrarEquipoIdeal(ideal1);
			Eurocopa.mostrarEquipoIdeal("equipoIdeal1.txt",ideal1);

			// segundo equipo ideal
			SortedSet<Jugador> ideal2 = e.formarEquipoIdeal(4,4,2);
			Eurocopa.mostrarEquipoIdeal(ideal2);
			Eurocopa.mostrarEquipoIdeal("equipoIdeal2.txt",ideal2);

			EurocopaJuegoLimpio ejl =
				new EurocopaJuegoLimpio("equipos.txt", "tarjetas.txt");

			// equipo ideal juego limpio
			SortedSet<Jugador> idealjl = ejl.formarEquipoIdeal(4,3,3);
			EurocopaJuegoLimpio.mostrarEquipoIdeal(idealjl);
			EurocopaJuegoLimpio.mostrarEquipoIdeal("equipoIdealJuegoLimpio.txt",
								idealjl);
		} catch (EurocopaException e) {
			System.out.println("ERROR: " + e.getMessage());
		} catch (IOException e) {
			System.out.println("ERROR de Entrada/Salida: " + e.getMessage());
		}
	}
}
```

La siguiente clase principal hace uso de GUIs. Está disponible en el campus virtual.

```
import javax.swing.*;

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

		VistaEurocopa panel = new PanelEurocopa();
		ControladorEurocopa ctr = new ControladorEurocopa(panel);
			// el modelo se crea dentro del controlador
		panel.controlador(ctr);

		JFrame ventana = new JFrame("Equipo Ideal Eurocopa 2012");
		ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		ventana.setContentPane((JPanel) panel);
		ventana.pack();
		ventana.setVisible(true);
	}
}
```