Práctica 4. El API de Java

En esta práctica se trabajará con el API de Java. En este sentido será importante acceder, leer y entender la documentación que recoge la página web de Oracle donde se consigna toda esta información. En particular, será necesario acceder a las clases Random, Character, String y ArrayList del API de Java.

4.1. Primera parte. Clase Random

Consultando la documentación del API de java, usar la clase Random del paquete java.util para que el número que hay que adivinar en los juegos de números sea un número aleatorio en lugar de un número predefinido por el programador. Para ello:

Modificación de la Clase JuegoAdivinaNumero

- Redefinir el método **reiniciaPartida** para que, además de ejecutar el código definido en la clase Juego, asigne un valor aleatorio al dato miembro que contiene el número a adivinar. Una opción para abordar la redefinición del método **reiniciaPartida** de forma que genere el número a adivinar de manera polimórfica (dependiendo del objeto que lo invoque) es crear un nuevo método denominado **numAleatorio** que es invocado desde **reiniciaPartida** y que retorna un número entero de forma aleatoria. Este método deberá ser implementado en la clase JuegoAdivinaNumero y sobreescrito en sus dos subclases, atendiendo a si el número generado aleatoriamente es par o impar.
- Para que el método **numAleatorio** genere un número entero aleatorio se usará un objeto de tipo Random (java.util.Random). A la hora de construir este objeto, es conveniente pasarle una semilla que evite que se genere siempre la misma secuencia de números pseudoaleatorios. Para ello puede usarse el método System.currentTimeMillis().
- Dadas las anteriores modificaciones el constructor ya no tendrá como parámetro el número por adivinar.

Modificación de Clase JuegoAdivinaNumeroPar

- Redefine el método **numAleatorio** en esta clase para que genere un número par aleatorio entre 0 y 10.

Modificación de la Clase JuegoAdivinaNumeroImpar

- Redefine el método **numAleatorio** en esta clase para que genere un número impar aleatorio entre 0 y 10.

4.2. Segunda parte. Las clases Character, String y Arraylist

Haciendo uso de las clases Character, String y ArrayList, junto con la clase Random, se va a implementar un nuevo juego, el Juego del Ahorcado Inglés, esta vez no basado en números sino en palabras tomadas del inglés y formadas por letras minúsculas.

Creación de la Clase JuegoAhorcadoIngles

- Se Deriva de la clase JuegoConVidas.

- Implementará la interfaz Jugable.
- Tendrá tres ArrayList como atributos:
    - Dos de caracteres, uno para almacenar la palabra a adivinar y otro, denominado guiones, donde se almacenarán los caracteres acertados.
    - El tercero es una ArrayList de String que contendrá todas las palabras de un diccionario y se utilizará para seleccionar de forma aleatoria la palabra a adivinar.
- El constructor tomará como primer parámetro el número de vidas y como segundo un ArrayList de String. Este ArrayList se utilizará para inicializar el atributo que contiene el diccionario de palabras.
- Tendrá los siguientes métodos:
    - Los métodos **muestraNombre** y **muestraInfo** reescritos para mostrar respectivamente el nombre y la información de este nuevo juego.
    - El método booleano **existeCaracter** que recibe como parámetro un carácter. El método comprueba si el carácter está presente (una o más veces) en la palabra adivinar y, si es así, incrementa un contador y lo sustituye en el ArrayList de guiones en la posición donde se produce la concordancia. Una vez finalizada la operación retorna true si el contador es mayor que cero y false en caso contrario.
    - El método **mostrarArray** muestra por pantalla el contenido del ArrayList de caracteres que se le pasa por parámetro.
    - La reescritura del método reiniciaPartida que invoca los métodos **reiniciaPartida** del supertipo, **mostrarNombre** y **mostrarInfo**. A continuación, vacía los ArrayLists que van a contener la palabra a acertar y los guiones y selecciona una palabra de manera aleatoria del diccionario de palabras. Esta palabra será utilizada para inicializar el ArrayList de la palabra a adivinar y rellenar el ArrayList de guiones. Por último, se muestra por pantalla el contenido del ArrayList de guiones, lo que revela al usuario el número de caracteres que tiene la palabra.
    - Implementa el método booleano **juega**:
        - ✓ Captura en una variable local el primer carácter de la cadena de caracteres (String) recibido como parámetro.
        - ✓ Invoca el método **existeCaracter** y, a continuación, se le pasa como parámetro el carácter obtenido con anterioridad. El retorno de este método se utiliza para controlar una estructura selectiva, tal que, si **existeCaracter** retorna true, se mostrará por pantalla el ArrayList de guiones, que revelará las posiciones que se han acertado, y se comprobará si la palabra ha sido adivinada. En caso contrario, se invocará el método **quitaVida** y se utilizará su valor de retorno para determinar si le quedan vidas al jugador para seguir jugando y devolver dicho valor como retorno del método juega. Para comprobar si se ha adivinado la palabra se utilizará otra estructura selectiva, anidada en la anterior. En el caso de que se haya adivinado la palabra se actualizará el récord y se finalizará la partida retornando el valor false. Si no se ha adivinado la palabra entonces se debe retornar el valor true.

Modificación de la clase JuegosReunidos:

- Se sustituye el array de objetos de tipo Jugable por un ArrayList de objetos de tipo Jugable (ArrayList<Jugable>).
- Se cambia el constructor para que inicialice el ArrayList a través de los objetos de los distintos juegos creados.
- se añade un método, denominado agregar, que no devuelve nada y recibe como parámetro un objeto de tipo Jugable. El método añadirá el objeto que se ha pasado por parámetro al atributo de la clase JuegosReunidos (ArrayList<Jugable>).

Extensión de la clase JuegosReunidos:

- Se crea la clase JuegosReunidosExtendidos que se deriva de la clase JuegoReunidos.
- Su constructor tiene como parámetro una referencia de tipo ArrayList<String> que representa el diccionario de palabras. En su cuerpo de definición, primero se invoca al constructor del supertipo y, a continuación, se invoca el método agregar al que se le pasa por parámetro un objeto de tipo Juego AhorcadoIngles, al que se inicializa con 5 vidas y el diccionario de palabras.

Extensión de la clase Menú

- Se crea la clase MenuExtendido que se deriva de la clase Menu.
- Su constructor recibe como parámetro un objeto de tipo JuegosReunidos. A continuación, inicializa el constructor del supertipo con dicho objeto.
- Se reescriben los siguientes métodos:
    - **mostrarOpciones**: invoca al método mostrarOpciones del supertipo y, a continuación, muestra por pantalla la nueva opción.
    - **chequeoOpcion**: retorna un valor booleano resultado de concatenar, mediante un operador AND, la invocación del método **chequeoOpcion** del supertipo y la comprobación de si la opción se corresponde con la cuarta opción (juego del ahorcado en inglés).

Modificación de la clase MyInput

- Se incluirá un nuevo método denominado **leeFichero** (se incluye el código) que recibe como parámetro una cadena de caracteres (String) que representa el camino, o “path”, donde se encuentra el fichero de texto plano y retorna como resultado un ArrayList que contiene en cada nodo todas las palabras contenidas en dicho fichero. Por esta razón, el fichero de texto debe estar configurado de tal forma que cada una de sus líneas solo contiene una palabra del diccionario. El método devuelve su contenido en un ArrayList que contiene en cada nodo una palabra.
- El fichero de texto se debe almacenar en la carpeta raíz del proyecto. De esta forma no es necesario pasar toda la ruta para su correcta localización.

```
public static ArrayList <String> leeFichero(String pathFichero){
   ArrayList <String> v = new ArrayList <String>();
   File fichero=null;
   FileReader fr=null;
```

```
BufferedReader br=null;
try{
   fichero=new File(nombreFichero);
   fr=new FileReader(fichero);
   br=new BufferedReader(fr);
   String linea;
   while ((linea=br.readLine())!=null){
      v.add(linea);}
}
catch (Exception e){
   e.printStackTrace();
}
finally {
   try {
      if (null!= fr){
         fr.close();
         br.close();}
      }
   catch (Exception e1){
        e1.printStackTrace();
      }
   }
return v;
}
```

Modificación de la clase principal

Por último, se deberá modificar la clase Aplicación para que desde el método principal se cree:

1) Un objeto de tipo JuegosReunidosExtendidos, al cual habrá que pasarle por parámetro el ArrayList que contiene el diccionario de palabras, que se encuentra almacenado en un fichero externo.
2) Un objeto de tipo MenuExtendido al que se le pasará por parámetro la referencia que apunta al objeto de tipo JuegosReunidosExtendidos.
3) Por último, la invocación a través del objeto de tipo MenuExtendido del método ejecuta.