SESIÓN 1: Documentación de proyectos. Programación guiada por pruebas (Junit 4)

Objetivos

- Crear la estructura adecuada de proyectos Java del alumno y la asignatura y vincularlos con los repositorios que correspondan.
- Saber crear y generar la documentación en una clase utilizando la herramienta javadoc.
- Utilizar la programación guiada por pruebas con JUnit para el diseño de las clases.
- Aprender a utilizar la documentación de la API del JDK.
- Aplicar las anteriores herramientas a estructuras de datos aprendidas en la asignatura Introducción a la programación.

Material proporcionado

- Guía rápida de aprendizaje de javadoc.
- Guía rápida de aprendizaje de JUnit.
- Las clases **Mayor, Carta** y **Baraja.**
- Los test de pruebas de los distintos ejercicios propuestos, **TestMayor** y **TestBaraja.**

Requisitos

- Seguir el esquema de nombrado de paquetes, es decir: **org.mp.sesion01**. En ese paquete se crearán todos los programas que se proponen en la sesión dándoles un nombre alusivo a lo que realiza el programa y que se indica en cada ejercicio en negrita.
- Documentar todas las clases.
- Generar de todas las clases los diagramas **UML**.
- Todas las clases deberán superar con éxito los test de pruebas.
- Al final de la sesión, el alumno deberá cargar el trabajo realizado a su repositorio indicando la clave correspondiente a la sesión.
- Las sesiones se han diseñado para cubrir una semana de trabajo.

Ejercicios propuestos

1. Crear el espacio de trabajo **nombreunidad:\WORKSPACES\DOCENCIA2017**. En ese mismo espacio de trabajo puedes situar los proyectos de otros años u otras asignaturas para cualquier consulta.

En el anterior espacio de trabajo, crear un proyecto Java, donde se realizarán todos los ejercicios de las sesiones, tanto propuestos como los del trabajo autónomo. El nombre del proyecto Java seguirá el esquema de nombrado que se indica, primero los apellidos, después el nombre del alumno añadiendo los primeros caracteres del nombre de la asignatura al final, MP y el año de la asignatura, 2017. Por ejemplo: **GomezPerezLuisMP2017**. En el proyecto, en la carpeta de fuentes **src,** se irán creando paquetes para cada una de las sesiones de trabajo. Los nombres de los **paquetes** deben mantener las normas de estilo propias del lenguaje Java (sólo usar minúsculas para los paquetes, **org.mp.sesion01**). Además se creará otra carpeta de código, **test**,

en la que situaremos los test unitarios con la misma estructura de paquetes que tenemos en en la carpeta src.

En ese mismo espacio de trabajo, tendremos el proyecto Java de la asignatura con el material tanto del grupo docente como del grupo de trabajo que irá proporcionando el profesor. El nombre de este proyecto es **MetodologiaProgramacion2017**.

El alumno trabajara con dos repositorios, uno estará vinculado a su proyecto y el otro al proyecto de la asignatura.

Repositorio de la asignatura (solo lectura para el alumno)

URL: **http://svndoc.ual.es:9090/svn/mp2017**

Usuario y contraseña: la del aula virtual.

Repositorio del alumno (lectura/escritura)

URL: **http://svndoc.ual.es:9090/svn/apellido1apellido2nombremp2017**

Ejemplo**: http://svndoc.ual.es:9090/svn/gomezperezluismp2017**

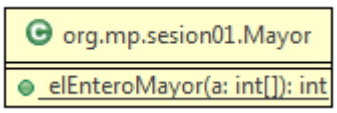
Usuario y contraseña: la del aula virtual.

Importante:

No olvides que si vas a trabajar con los repositorios fuera de la UAL, será necesario estar conectados a la VPN de la universidad. Puedes consultar como hacerlo en:

http://vpn.ual.es/

2. La clase **Mayor** tiene el método **elEnteroMayor** que devuelve el mayor entero de una lista que se le pasa como parámetro. La lista está soportada por un array.
   Se debe ejecutar el juego de pruebas que se proporciona, comprobar su funcionamiento y corregir a través de las pruebas los errores encontrados en el código.



3. Consulta las siguientes clases de la API:

   String, Integer, Long, Short, Float, Double, Byte, Boolean, Character

Trabajo autónomo

4. Documenta la clase **Fraccion** del paquete **org.ip.sesion06** de la asignatura IP.

5. Queremos ordenar una baraja española de 40 cartas en el orden natural: primero palo y en el mismo palo en orden ascendente numérico, es decir:

O1, O2,03,…,O12, C1,C2,C3,…,C12, E1,E2,E3,…,E12, B1,B2,B3,…B12

O →Oros, C →Copas, E →Espadas, B →Bastos

Termina de implementar las clases **Carta** y **Baraja**. Utiliza el método de ordenación por inserción haciendo uso del método **compareTo**.