**Examen UF5-PE1. Estructura de dades**

**Indicacions per al lliurament**

La prova es realitzarà de forma individual i amb accés a apunts, però sense accés a Internet.

El lliurament es realitzarà al curs de Moodle en un únic document comprimit en format *zip*.

El nom del fitxer a lliurar tindrà el format:

**m03-uf5-pe1-cognom1\_cognom2\_nom.zip**

Cal lliurar només els fitxers de codi font *.java*.

No s’accepten lliurament fora de termini o per altres vies.

**Criteris de correcció**

· Es requereix per puntuar que el programa compili i no generi cap error. Cas contrari es puntuarà amb un 0.

· Es valoraran únicament els apartats que realitzin la funció exacta que sol·licita l’enunciat.

· S’han d’utilitzar les estructures demanades o més adients per realitzar cada funcionalitat del programa.

· El tractament d'errors ha de ser adequat, de manera que si es produeixen hauria d'informar-se a l'usuari del tipus d'error produït.

· Es tindrà en compte la validació de dades entrades per l’usuari i el tractament adequat dels errors.

· Cal proporcionar a l’usuari els missatges adequats a cada situació.

· Cal documentar i indentar adequadament el codi font. La documentació ha d’incloure just abans de la declaració de la classe l’explicació de l’objectiu o funcionalitat del codi i el nom i cognoms de l’autor, així com la data de realització. Cal incloure també la documentació interna del codi per explicar què fa cada fragment de codi que sigui significatiu.

**EXERCICI 1 (7,5 p)**



Es proposa fer el torneig definitiu de Karate All-Valley!! Programarem un programa que contindrà el un llistat de tots els lluitadors inscrits amb la seva força i la seva resistencia.

**Class KarateTournee**

Hi haurà un llistat de tots els apuntats,amb els atributs següents:

* listMembers (Hi ha un mètode per generar-lo)
* Season (Es pot especificar al crear-se, i vol dir el any en què es celebra el Torneig “2024”)
* Pot tenir algun atribut més si us fa falta per algún requisit dels mètodes.

Mínim té que tenir implementat aquests mètodes:

**public KarateMember addFighter(KarateMember kf)** (amb total 2 pts)

ha d'afegir el lluitador.

* El nom i el dojo el demanarà l’usuari.
* El dorsal, estaria bé el calcules abans de crearlo amb funció del número d’inscrits a la listMembers, si no sabeu com, també podeu fer que ho introdueixi l’usuari, però contarà tota la puntuació**(0,5)**
* Sol ha de permetre 16 inscrits (per provar pots fer 3, però al final ha de ser 16), en cas de que ja hi siguin tots ha de llençar una excepcion (no MoreRegistrationsAllowedException). **(0,5 pt)**
* No es pot afegir cap lluitador repetit si ja existeix ha de saltar l’excepció FighterDuplicateException. **(0,5 pt)**
* Afegir-lo i avisar que s’ha afegit **(0,75 pt)**

**public boolean removeFighter(KarateMember kf)** ha de poder eliminar de la llista el membre que rep com a paràmetre.El membre ja ha de venir informat amb els camps clau de KarateMember **(0,75 pt)**

**public boolean LostFigther(int Dorsal)** a partir de sols el id, que no és camp clau, l’ha de borrar **(0,75 pt)**

**getters i setters** necessàris (getListMembers() .. )

**Class KarateMember (1,5 pts)**

Hi haurà una classe KarateMember que ha de tenir els atributs següents:

- Dorsal(el rep al crearse)

- Power(Random entre 50 i 100)

- name (Es pot especificar al crear-se)

- Dojo al qual pertany (Es pot especificar al crear-se)

*Un lluitador és igual a un altre si coincideix per nom i pel Dojo.*

Haurà d’implementar els mètodes:

* **getters and setters** necessàris
* **2 Constructors.** 1 què rep el nom , el dojo i el dorsal, i un segon on sol rep el nom i el dojo.
* **mètode per imprimir per pantalla**  Ha de printar de l’estil:   
  “Fighter {Name} has {Power} power”

**Class SortByPower** (que ha d’implementar Comparator<T>) **(1 pts)**

Ens ha de permetre ordenar els lluitadors per la seva potencia d’atac.

**KarateArea (MainClass) (1,75 correcte comunicació main )**

Al inici s’ha de carregar 5 lluitadors d’entrada

Llavors ha de mostrar-se el llistat amb les opcions

1.- Afegir lluitador

2.- Borrar Lluitador

3.- Lluitador perd

4.- Mostrar lluitadors ordenats per més forts.

5.- Mostrar lluitadors d’un dojo específic.

Sorted by Power

{name 3} {id 3} {fight3}

{name 2} {id 2} {fight2}

{name 4} {id 4} {fight4}

{name 1} {id 1} {fight1}

{name 2} {id 2} {fight2}

Number members listed …. 5

El llistat per dojo ha de ser no és necessari que estiguin ordenats per força

Dojo Escollit: {NomDojo}

“Fighter {Name} has {Power} power”

“Fighter {Name} has {Power} power”

“Fighter {Name} has {Power} power”

Number members listed …. 3

**Exercici 2 (2,5p)**

Resulta que al acabar el torneig la gent vol saber quants diners cobra en funció de la ronda on han quedat eliminats, per això van a un robot que té els diners, aquest robot està programat amb un map amb la següent taula de valors:

* Campió→ 4000$
* SotsCampió→ 2000$
* Semifinal→ 1000$
* Quarts→ 300$
* Vuitens → 0$

Crea un Map que s’anomeni premis on la clau ha de ser de tipus String i el valor de tipus enter.

Has de implementar els següent mètodes:

* **loadAwards()** ha de carregar els elements proposats (0,2 ps)
* **showAwards()** ha de mostrar la carta amb el format proposat (0,4 pt)
* **requestAwards()** l'usuari indica a quina ronda ha arribat i li diu el que ha guanyat, si fica alguna posició no coneguda, ho té que indicar(1 pt).
* **requestAwardsbyMoney()** és demana quina quantitat s’ha guanyat i t’ha de dir en quina posició s’ha quedat (0,4 pt)