FIGTH AGAINST MONSTERS

Vamos a implementar una aplicación en la que nos vamos a crear un personaje que va a luchar contra una serie de monstruos.
Habrá 4 tipos de monstruos, ShadowMonster, LightMonster, FireMonster y WaterMonster.
Cada uno tendrá unas habilidades especiales y también unas debilidades.
Dispondremos de unas armas, que según el tipo de elemento del arma, harán más daño o menos a los monstruos.
Al principio sólo dispondremos de un arma para atacar, conforme vayamos ganando batallas contra monstruos, tendremos la opción de ampliar nuestro repertorio de armas.

A continuación describimos las clases que vamos a implementar:

La **clase Player**, que tendrá los atributos:

```
String name;
int life, strength, speed, exp;
ArrayList<Weapon> arrayListPlayerWeapons = new ArrayList<>();
Weapon actualWeapon;
int initLife;
```

Los atributos son todos bastante autodescriptivos, initLife nos servirá para restablecer la vida del jugador después de una batalla.
ActualWeapon es el arma con el que vamos a luchar, de entre las disponibles en arrayListPlayerWeapons.

Aparte de los setters y getters necesarios, tendremos los métodos:

addWeapon(Weapon newWeapon) → para añadir un arma a nuestro repertorio.
void resetLife() → para restablecer la vida a su valor original
String takeAttack(Monster m) → este método devolverá un string que nos describirá el daño que hemos recibido. Según el tipo de monstruo que nos ataque, ShadowMonster, LightMonster, FireMonster y WaterMonster, recibiremos un daño base, y un daño exclusivo del tipo de monstruo (shadowAction, waterAction, lightAction) que se verán más abajo.

Crearemos una clase abstracta **Monster**, que tendrá las propiedades:

```
String MonsterName;
int life,strength, speed;
int initLife;
```

y el método abstracto takeAttack(Player p).

Aparte de implementar todos los métodos necesarios que veamos que van a ser comunes a todas las clases que hereden de Monster.

Crearemos la interfaz **Shadow**:
Que tendrá los atributos definidos MAXSHHOTATTACK = 20, MAXKICKATTACK = 5.
y declarará el método int shadowAction().

Crearemos la clase **ShadowMonster** que heredará de Monster e implementará la interfaz Shadow.

La interfaz Shadow nos obligará a implementar el método:
int shadowAction, que devolverá con un 50% de probabilidad uno de los siguientes ataques:

int waterShot() → devuelve un número entre 5 y MAXSHHOTATTACK
int waterKick() → devuelve un número entre 3 y MAXKICKATTACK

Por otro lado tendremos que implementar el método
String takeAttack(Player p).
Si el arma con el que ataca el personaje es del elemento “Light” nos hará 1.5 veces el daño que nos haría normalmente.
En caso contrario nos hará el daño normal, fuerza del player + fuerza del arma.

Crearemos la interfaz **Water**:
Que tendrá los atributos definidos MAXSHHOTATTACK = 15, MAXKICKATTACK = 10.
y declarará el método int shadowAction().

Crearemos la clase **WaterMonster** que heredará de Monster e implementará la interfaz Water.

La interfaz Shadow nos obligará a implementar el método:
int waterAction, que devolverá con un 50% de probabilidad uno de los siguientes ataques:

int waterShot() → devuelve un número entre 5 y MAXSHHOTATTACK
int waterKick() → devuelve un número entre 3 y MAXKICKATTACK

Por otro lado tendremos que implementar el método
String takeAttack(Player p).
Si el arma con el que ataca el personaje es del elemento “Fire” nos hará 1.5 veces el daño que nos haría normalmente.
En caso contrario nos hará el daño normal, fuerza del player + fuerza del arma.

Crearemos la interfaz **Light**:
Que tendrá los atributos definidos MAXSHHOTATTACK = 13, MAXKICKATTACK = 12.
y declarará el método int shadowAction().

Crearemos la clase **LightMonster** que heredará de Monster e implementará la interfaz Light.

La interfaz Shadow nos obligará a implementar el método:
int waterAction, que devolverá con un 50% de probabilidad uno de los siguientes ataques:

int waterShot() → devuelve un número entre 5 y MAXSHHOTATTACK
int waterKick() → devuelve un número entre 3 y MAXKICKATTACK

Por otro lado tendremos que implementar el método
String takeAttack(Player p).
Si el arma con el que ataca el personaje es del elemento “Shadow” nos hará 1.5 veces el daño que nos haría normalmente.

En caso contrario nos hará el daño normal, fuerza del player + fuerza del arma.

Crearemos la clase **Weapon**, que tendrá los siguientes atributos:

```
String name, element;
int damage, speed;
String id;
```

Crearemos nuestra clase Main, donde nos podremos crear un arrayList de Weapons y uno de monsters.

Podemos añadir a modo de ejemplo los siguientes weapons

```
add(new Weapon("Fire Sword", "Fire", 20, 12,"ST12"));
add(new Weapon("Emo sword", "Shadow", 14, 24,"TR16"));
add(new Weapon("Light gravestone", "Light", 22, 13,"NT15"));
add(new Weapon("Sea Slayer", "Water", 14, 25,"TG23"));
add(new Weapon("Life Sword", "Grass", 18, 22,"AM27"));
```

Y los siguientes monsters

```
add( new WaterMonster("Oceanid", 120, 15, 5));
add(new ShadowMonster("Mama grande", 130, 25, 2));
add(new LightMonster("Sun Dragon", 200, 10, 10));
```

La clase Main dispondrá del siguiente menú:

```
1.Create your character
2.Fight against a monster
3.Sort by...
4.Exit
Option:
```

Con la primera opción, nos pedirá un nombre para nuestro personaje, y creará un player con vida 100, fuerza un numero aleatorio entre 5 y 10, velocidad un número aleatorio entre 5 y 10, y experiencia 0.

y nos dará la opción de escoger una de las armas disponibles añadidas al arrayLists.

```
Whats your name:
pepe
Pick one of this weapons to Start the Fight:
---------------------------------
0
Name = Fire Sword
Element = Fire
Damage = 20
Speed = 12

---------------------------------
1
Name = Emo sword
Element = Shadow
Damage = 14
Speed = 24

---------------------------------
2
```

```
Name = Light gravestone
Element = Light
Damage = 22
Speed = 13

---------------------------------
3
Name = Sea Slayer
Element = Water
Damage = 14
Speed = 25

---------------------------------
4
Name = Life Sword
Element = Grass
Damage = 18
Speed = 22


Write the number of the weapon: 2
Light gravestoneacquired


Created Character:


Jugador:
Name = pepe
Strength = 7
Speed = 9
Life = 100
Exp = 0

Weapons:
 Name: Light gravestone, Element: Light, Damage: 22, Speed: 13
```

De esta manera podremos empezar a luchar con monstruos.
Se tendrá que comprobar que se escoge correctamente un arma de los posibles. (no valido caracteres, número de arma correcto ….)

En la segunda opción si hemos creado un personaje podremos luchar contra un monstruo aleatorio escogido del arrayList contenedor de monstruos.

En caso de no haber creado el personaje todavía, nos mostrará un mensaje de error y volveremos al menú inicial.

En caso que tengamos nuestro personaje creado, empezaremos la lucha que durará mientras player o monster tengan vida.
Será una lucha por turno, empezando uno de los dos aleatoriamente.
En cada turno nos mostrará el estado de los dos combatientes

```
THE FIGHT WILL START

THOSE ARE THE CURRENT STANDINGS
pepe has 100 remaining Life
Oceanid has 120 remaining Life
New round
```

y los turnos se irán sucediendo conforme vayamos tecleando “enter” y mostrando quien ataca.

```
THOSE ARE THE CURRENT STANDINGS
pepe has 13 remaining Life
Oceanid has 120 remaining Life
New round
```

```
Turn of Player!
Monster take 14 of damage

THOSE ARE THE CURRENT STANDINGS
pepe has 13 remaining Life
Oceanid has 120 remaining Life
New round

Turn of the Monster!
A water shot is coming!
monster attack does 15 of basic damage
monster attack does 12 damage of WaterAction


End of the combat

Oceanid won!


Player vs Monsters
1.Create your character
2.Fight against a monster
3.Sort by...
4.Exit
Option:
```

Mostrando el ganador al final. Y devolviéndonos al menú principal.
En caso de ganar el jugador, subirá un punto su experiencia.
Al final de la lucha se restablecerán las vidas del monstruo y del jugador.

La mecánica de la lucha es la siguiente:

El jugador tiene una probabilidad de esquivar el ataque igual a 100-velocidad del monstruo atacante.
El monstruo tiene una probabilidad de esquivar el ataque de 100 -(velocidad del jugador + velocidad del arma que está utilizando)
En caso de esquivar ataque, se nos mostrará un mensaje informativo.

En caso de no esquivar el ataque, el jugador o el monstruo recibirán el daño definido en el método takeAttack y cambiaremos de turno.

Por último, la tercera opción del menú, nos dara las siguientes opciones:

```
1)Show Player Weapons ordered by damage
2)Show Player Weapons ordered by Speed
3)Show Monsters ordered by Strength
4.Back
```

Que son auto descriptivas. Tendremos que hacer uso de las interfaces Comparable y Comparator.