ADSOF: Segundo examen parcial - 06/05/2020

Ejercicio 2 (3.5 puntos)

Necesitamos una aplicación para controlar las calorías quemadas al realizar *actividades deportivas* de varias *modalidades*.
El *total quemado* o *gastado* en cada actividad resulta de aplicar un porcentaje de *incremento por nivel de intensidad* a la suma de su *gasto estándar* más su *gasto adicional*. Los porcentajes de incremento por nivel de intensidad de la actividad son 0%, 0%, 6.25% y 12.5% respectivamente para las intensidades mínima, baja, media y máxima, respectivamente. El gasto adicional será 0 por defecto, a menos que se haya especificado otro para esa actividad. El gasto estándar se debe calcular de forma distinta según el tipo de actividad. Por ahora consideraremos solo las actividades con *pesas* y las *carreras*. El gasto estándar de las pesas se indica individualmente junto a la descripción de la actividad y la intensidad asignadas para cada actividad con pesas. El gasto estándar de las carreras depende de la duración indicada junto a la descripción y a la intensidad (asumiendo intensidad baja si no especifica ninguna): será de 10.0 Kcal por minuto, con un mínimo de 150.0 Kcal y un máximo de 800.0 Kcal por carrera.

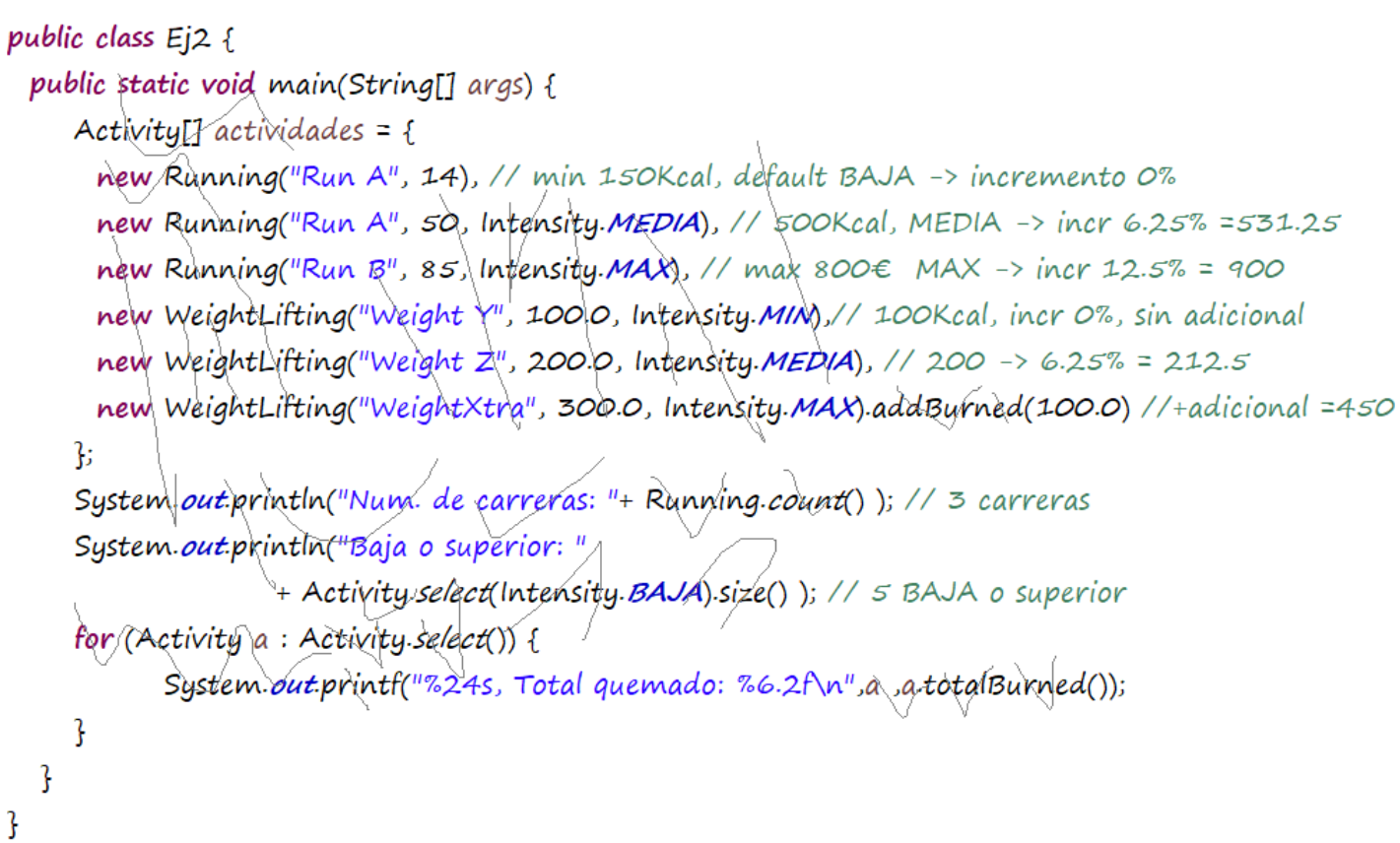
Además, queremos poder acceder fácilmente a todas las actividades de intensidad igual o superior a un nivel de intensidad indicado, o todas las actividades si no se indica ninguno. También deseamos contabilizar el número total de carreras, haciendo que la aplicación añada automáticamente dicha numeración a la descripción de cada carrera.

Nota: por simplicidad, puedes ignorar el control de errores derivados de cantidades negativas y strings vacíos o nulos.

Se pide diseñar y codificar en Java, siguiendo buenos principios de orientación a objetos, las clases necesarias para resolver los requisitos anteriores haciendo que el programa dado más abajo **produzca la siguiente salida:**

```
Num. de carreras: 3
Baja o superior: 5
          Run A#1, BAJA, Total quemado: 150,00
         Run A#2, MEDIA, Total quemado: 531,25
           Run B#3, MAX, Total quemado: 900,00
          Weight Y, MIN, Total quemado: 100,00
        Weight Z, MEDIA, Total quemado: 212,50
        WeightXtra, MAX, Total quemado: 450,00
```

Tester en forma de imagen para evitar cortar y pegar de texto del enunciado a Eclipse. Más abajo va en forma de texto.



```
public class Ej2 {
  public static void main(String[] args) {
     Activity[] actividades = {
       new Running("Run A", 14), // min 150Kcal, default BAJA -> incremento 0%
       new Running("Run A", 50, Intensity.MEDIA), // 500Kcal, MEDIA -> incr 6.25% =531.25
       new Running("Run B", 85, Intensity.MAX), // max 800€  MAX -> incr 12.5% = 900
       new WeightLifting("Weight Y", 100.0, Intensity.MIN),// 100Kcal, incr 0%, sin adicional
       new WeightLifting("Weight Z", 200.0, Intensity.MEDIA), // 200 -> 6.25% = 212.5
       new WeightLifting("WeightXtra", 300.0, Intensity.MAX).addBurned(100.0) //+adicional =450
     };
     System.out.println("Num. de carreras: "+ Running.count() ); // 3 carreras
     System.out.println("Baja o superior: "
                        + Activity.select(Intensity.BAJA).size() ); // 5 BAJA o superior
     for (Activity a : Activity.select()) {
           System.out.printf("%24s, Total quemado: %6.2f\n",a ,a.totalBurned());
     }
   }
}
```

Solución: Indica asignación orientativa de n décimas de puntos en forma de `[n]` sobre total de 35. Otros fallos penalizados aparte.

```
public enum Intensity {
	// MIN(0.0), LOW(0.0), AVERAGE(0.0625), MAX(0.125);
	MIN(0.0), BAJA(0.0), MEDIA(0.0625), MAX(0.125);          // [2]
	private double value;
	private Intensity(double value) { this.value = value; }     // [2]
	public double getValue() { return this.value; }
}
public abstract class Activity {    // abstract [1]
	private static List<Activity> activities = new ArrayList<>();
	public static List<Activity> select(Intensity en) {    // [3]
		List<Activity> result = new ArrayList<>();
		for (Activity a : Activity.activities)
			if (a.priority.compareTo(en)>=0) result.add(a);
		return result;
	}
	public static List<Activity> select() {              // [2]
		// return select( Intensity.values()[0]);
		return Collections.unmodifiableList(Activity.activities);
	}
	private Intensity priority;
	private String description;
	public Activity(String d, Intensity e) {     // [3]
		description = d; priority = e; activities.add(this); }

	public double totalBurned()  { return (this.standardBurn()    // [4]
					+ this.addBurned() )
					* (1.0 + this.priority.getValue()); }
	public abstract double standardBurn();        // [1]
	public  double addBurned() { return 0.0; }    // [1]
	public String toString() {return description + ", "   + this.priority;}  // [1]
}
import java.util.*;
public class Running extends  Activity {    //  extends [1]
	public static final double MAX_STD_BURN = 800.0;
	public static final double MIN_STD_BURN = 150.0;
	public static final double BURN_PER_MINUTE = 10.0;
	 private static int lastId = 0;                 // [1]
	 public static int count() { return lastId; }   // [1]

	private int minutes;
	public Running(String descripcion, int c1, Intensity enumera) {
		super(descripcion + "#"+  ++lastId, enumera);    // [2]
		this.minutes = c1;
	}
	public Running(String descripcion, int c1) {
		this(descripcion, c1, Intensity.BAJA); // LOW);   // [2]
	}
	@Override public double  standardBurn() {   // [1]
		return Math.min(MAX_STD_BURN,
				Math.max(MIN_STD_BURN,
					BURN_PER_MINUTE * this.minutes));
	}
}
public class WeightLifting extends Activity {   // extends [1]
	private double standardBurn;
	private double extra = 0.0;
	public WeightLifting(String descripcion, double burned, Intensity enumera) {
		super(descripcion, enumera);    // [2]
		standardBurn = burned;
	}
	@Override public double standardBurn() { return this.standardBurn; } // [1]
	@Override public double addBurned()  { return this.extra;}           // [1]
	public WeightLifting addBurned(double extra)  { this.extra = extra; return this;}  // [2]
}
```