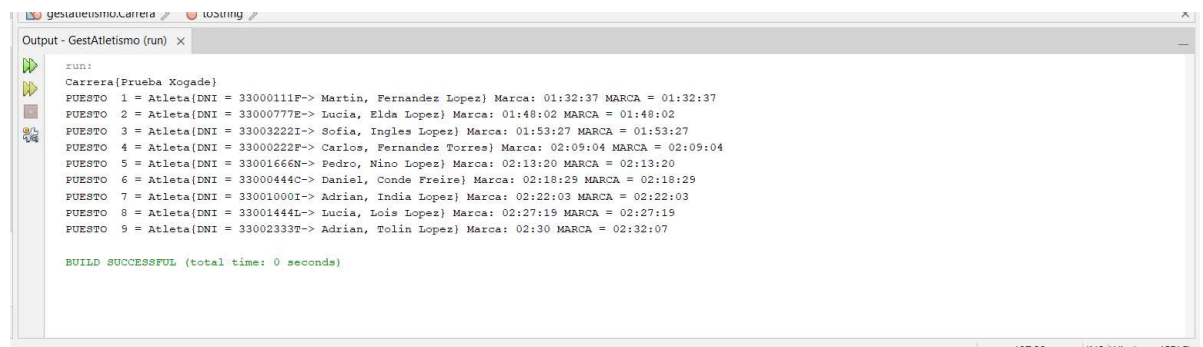


Examen II Recuperación - 2ª Evaluación 04/04/2024

Ejercicio:

Se pide realizar una aplicación en Java que permita gestionar una competición de atletismo. El proyecto se llamara **GestAtletismo**. Las acciones que debemos poder realizar (mediante un menú) son:

1. Crear una **carrera**. Para ello se pedirán los datos necesarios por consola.
2. Insertar **participantes**:
 1. Automáticamente (Se pedirá un número y se agrará ese número de participantes usando la librería **atletismo**)
 2. Manualmente (Se pedirá por consola los datos necesarios para crear un participante y se añadirá utilizando los datos aportados por la librería **atletismo**)
3. Simular la carrera. (Cada corredor obtendrá una marca para la carrera)
4. Mostrar los resultados ordenados (de menor a mayor tiempo) por pantalla. (Para ordenarlos es obligatorio usar la clase Collections). Los campos a mostrar son:
 1. Puesto obtenido en la carrera
 2. Marca obtenida en la carreta
 3. Datos personales (dni, nombre y apellidos).
 4. Marca perrsonal actualizada.



```
run:
Carrera[Prueba Xogade]
PUESTO 1 = Atleta[DNI = 33000111F-> Martin, Fernandez Lopez] Marca: 01:32:37 MARCA = 01:32:37
PUESTO 2 = Atleta[DNI = 33000777E-> Lucia, Elda Lopez] Marca: 01:48:02 MARCA = 01:48:02
PUESTO 3 = Atleta[DNI = 33003222I-> Sofia, Ingles Lopez] Marca: 01:53:27 MARCA = 01:53:27
PUESTO 4 = Atleta[DNI = 33000222F-> Carlos, Fernandez Torres] Marca: 02:09:04 MARCA = 02:09:04
PUESTO 5 = Atleta[DNI = 33001666N-> Pedro, Nino Lopez] Marca: 02:13:20 MARCA = 02:13:20
PUESTO 6 = Atleta[DNI = 33000444C-> Daniel, Conde Freire] Marca: 02:18:29 MARCA = 02:18:29
PUESTO 7 = Atleta[DNI = 33001000I-> Adrian, India Lopez] Marca: 02:22:03 MARCA = 02:22:03
PUESTO 8 = Atleta[DNI = 33001444L-> Lucia, Lois Lopez] Marca: 02:27:19 MARCA = 02:27:19
PUESTO 9 = Atleta[DNI = 33002333T-> Adrian, Tolin Lopez] Marca: 02:30 MARCA = 02:32:07

BUILD SUCCESSFUL (total time: 0 seconds)
```

5. Salir.

La **carrera** consta de los siguientes datos:

- Un nombre
- Fecha y hora de realización.
- Número máximo de participantes.
- Edad mínima.
- Los participantes.

Condiciones:

- Si se intenta añadir más corredores que el número máximo de participantes se debe lanzar una excepción del tipo **NumeroMaximoException**. mostrando por consola un mensaje de que se ha lanzado la excepción (utilizando las sentencias y métodos adecuados para el control de errores). La excepción debe controlarse y permitir la ejecución normal del programa.
- En la inscripción automática, se solicitará un número de atletas mediante la librería. Si el número de atletas es superior al número máximo de corredores que admite la carrera, se inscribirán según el orden en que se han obtenido hasta llegar al número máximo de corredores de la carrera.

- No se puede inscribir corredores que no tengan la edad mínima en el momento de la carrera.
- Dos corredores inscritos en la carrera no pueden tener el mismo DNI.
- Si un corredor consigue mejorar su marca personal, está debe actualizarse.
- Para la simulación de la carrera cada corredor obtendrá un tiempo aleatorio independiente.
- Para el recorrido de los elementos de la colección o mapa que uséis, es obligatorio, usar como mínimo en una ocasión la clase Iterator.
- El código debe seguir los paradigmas de la POO, por lo que toda la gestión de los datos se realizará en una clase llamada Control.
- Para el almacenamiento podéis utilizar la estructura de almacenamiento que creáis conveniente (excepto arrays).
-

Libreria atletismo:

◆ Atleta

ATRIBUTOS

```
private String dni;
private String nombre;
private String apellido1;
private String apellido2;
private LocalDate fechaNacimiento;
private LocalTime marcaPersonal;
```

CONSTRUCTORES

```
public Atleta(String dni, String nombre, String apellido1, String apellido2, String
    fechaNacimiento)
public Atleta(Atleta p)
```

GETTERS Y SETTERS

```
public LocalDate getFechaNacimiento()
public String getDni()
public String getNombre()
public String getApellido1()
public String getApellido2()
public LocalTime getMarcaPersonal()
public void setMarcaPersonal(LocalTime marcaPersonal)
```

METODOS

```
public boolean equals(Object obj)           Devuelve true si dos atletas tienen el mismo DNI o
                                              son el mismo atleta.

public String toString() {
    return "{" + "DNI = " + dni + "-> " + nombre + ", " + apellido1 + " " + apellido2 + "}';
}
```

◆ Datos

CONSTRUCTORES

```
public Datos()
```

METODOS

```
public ArrayList<Atleta> getRandomAtletas(int cantidad)
```

Devuelve una colección de objetos del tipo Atleta. El número de objetos en la colección coincide con el parametro cantidad. No hay garantía de que no se repitan los objetos.

◆ Herramientas

METODOS

```
public static LocalTime getTiempoCarrera()
```

Devuelve un LocalTime aleatorio.

Entrega:

Un archivo .zip en el que se incluya el proyecto completo en formato Netbeans. Dicho proyecto, debe ser exportable a otras instancias de Netbeans, sin necesidad de modificar ningún archivo, para poder realizar la corrección. Si la estructura de archivos no es correcta o se requieren modificaciones para poder ejecutar el código, se descontará un 20 % de la nota final del examen. Todos los archivos java deben llevar una línea comentada (//Nombre_Apellido1_Apellido2) con el nombre y apellidos del alumno.

Para poder calificar el proyecto, debe poderse ejecutar el programa y probar la validez de cada una de las partes del mismo.

Calificación:

En esta tabla se muestran las puntuaciones máximas que se pueden obtener en cada uno de los bloques.

Apartado 1: Cargar librería y uso de todas sus clases 0,75

Apartado 2: Crear carrera 1

Apartado 3: Inscribir corredores 2,5

Apartado 4: Simular carrera 0,75

Apartado 5: Mostrar resultado ordenado de la carrera simulada 1

Apartado 6: Uso de Iterator ,75

Apartado 7: Control de errores 0,5

Apartado 8: Correcto funcionamiento del programa y ajustado a las condiciones solicitadas 2