Escuela de Arquitectura y Tecnología Grado en Ingeniería Informática

PROGRAMACIÓN ORIENTADA A OBJETOS II

Examen parcial 2

Curso:	Segundo	**Curso académico:**	2021/2022
Fecha examen:	03/06/22	**Duración examen:**	100 minutos
Convocatoria:	Primera	**Material permitido:**	TODO menos RTC.
Instrucciones:	Crear un proyecto llamado nombreApellidoParcial2. Utilizar un paquete con vuestro nombre y almacenar cada pregunta en un subpaquete diferente. Para entregarlo exportar el proyecto desde el IDE como un .zip y subirlo a la PDU.		

1. **(2 puntos)** Crea una serie de clases capaz de almacenar la información relacionada con la población de un País:
   1.1. Crea la clase **País**, que contenga un nombre de país, y una **colección (no un array [])** de Provincias. Haz un método añadirProvincia que reciba como parámetro una Provincia y la añada a la colección.
   1.2. Crea la clase **Provincia**, que contenga un nombre de provincia y una **colección** de Personas. Haz un método añadirPersona que reciba como parámetro una Persona y lo añada a la colección.
   1.3. Crea la clase **Persona**, que contenga lo necesario para representar a una persona: nombre, apellido1, apellido2, edad, id. Garantiza por construcción que las personas reciban ids diferentes (consecutivos es lo más fácil) conforme son creados y asegurando que **nunca** se repite ninguno.
   1.4. Al **imprimir un objeto** de tipo País debe verse toda la información del País, las provincias y las personas que contiene.

2. **(3 puntos)** Comprueba que las clases anteriores funcionan, creando una clase HelperPais que contenga una serie de **métodos estáticos**:
   2.1. **createPersona**: recibe como parámetro de entrada dos **sets**, uno de nombres y otro de apellidos y genera una Persona con un nombre y dos apellidos elegidos de forma aleatoria entre los de los sets recibidos. El apellido puede ser repetido y todos los nombres y apellidos deben tener la misma probabilidad de ser elegidos. La edad será un entero aleatorio entre 1 y 100.
   2.2. **createPais**: recibe como parámetro de entrada el nombre del país, el número de provincias del país, el número máximo y mínimo de personas que puede haber en cada provincia, un set de nombres y un set de apellidos. Determina el número de personas **de cada provincia** de forma aleatoria entre el máximo y mínimo dado, créalas y añádelas a la provincia. Como resultado devuelve el País generado.
   2.3. **buscaPersonas**: recibe como parámetro de entrada un País, un nombre y dos apellidos y devuelve la colección de Personas cuyo nombre y apellidos coinciden con los dados. Si no encuentra ninguno en el Pais lanza una Excepción checked de tipo PersonaNotFound (tendrás que crearla primero).
   2.4. Crea una clase **Ejercicio2** con un método main y comprueba que los métodos anteriores funcionan creando un País llamado España con 50 provincias que tienen entre 20 y 50 personas cada una y un set de nombres con [“Pepe”,”Juan”,”Marcos”,”Lucia”,”Maria”] y un set de apellidos con

["Gomez","Perez","Sanchez","Garcia","Lopez"]. Busca una persona que exista e imprime cuantas personas has encontrado con ese nombre y apellidos. Busca una que no exista, captura la excepción e imprime un mensaje por pantalla.

3. **(4 puntos)** Crea una clase Indice que tenga una serie de métodos estáticos:
   3.1. **generarExtensiones** que reciba como parámetro de entrada un número de extensiones a crear y devuelva un array de Strings ([] un **array "clásico",** no una colección) que contenga extensiones aleatorias. Las extensiones se crean uniendo un punto y tres letras aleatorias, pero debes garantizar que **no hay duplicados** en el array de extensiones devuelto.
   3.2. **generarArchivos** que reciba como parámetro de entrada un **Path** (no un String) a una carpeta, un número de archivos y un array de extensiones. La función generará el número de archivos pedido (archivos vacíos) en la carpeta indicada. Cada archivo tendrá como nombre un numero consecutivo (para evitar colisiones) y una extensión elegida de forma aleatoria de entre las recibidas como parámetro. Asegúrate de que la carpeta existe antes de crear los archivos y si no existe créala tu.
   3.3. **generarIndice** que reciba como parámetro de entrada un Path a una carpeta y devuelva un Map<String, Integer>. La función debe recorrer la carpeta dada e ir generando un índice que indique el número de archivos que existen en la carpeta para cada extensión.
   3.4. Crea una clase **Ejercicio3** con un método main y comprueba que los métodos anteriores funcionan creando un array de 1000 extensiones, generando 5000 archivos en una carpeta llamada p3 dentro del proyecto (**que funcione cuando lo pruebe en mi ordenador**), generando el índice e imprimiéndolo por pantalla de forma legible.

4. **(1 puntos)** Haz una versión recursiva de las funciones anteriores que funcione para varios niveles de anidamiento y comprobar que funciona:
   4.1. La función generarExtensiones no hace falta modificarla.
   4.2. La función generarArchivosRecursivo es igual que generarArchivos pero recibe también la probabilidad de crear una carpeta y la va reduciendo cada vez que crea una carpeta anidada (para garantizar que converge).
   4.3. La función generarIndiceRecursivo es igual a generarIndice pero recibe también como parámetro de entrada el Map<String,Integer> que esa construyendo para incluir los archivos de la carpeta actual (y las que contenga).
   4.4. Crea una clase **Ejercicio4** con un método main donde compruebes que los métodos anteriores funcionan, elige los valores que consideres para probarlo adecuadamente.