1. **Descargar este archivo**

https://pastebin.com/G6UyHYt7

2. **Copiar** estos archivos a su proyecto.

Enunciado:

1. Crear las clases y metodos necesarios para poder leer el archivo json. (Recomendación: Usar Jackson).

2. De la lista anterior de **aeropuertos** generar otra lista pasando los aeropuertos cuya capacidad sea mayor a una enviada por parámetro. Cuando la capacidad sea **menor o igual** a la enviada, lanzar una excepción **custom** almacenando la capacidad del aeropuerto. **20 pts.**

3. Crear una **clase genérica** para almacenar tanto los pilotos como los tags en un **arraylist**. Las opciones de esta clase genérica son agregar, eliminar y recuperar según posición. **20 pts.**

4. Desde la **lista genérica** pasar a un **hashset** todos los tags. Implementar el **equals** en la **etiqueta. 20 pts.**

Implementar un main para probar todas las funcionalidades. **10 pts.**

1. Copiar el contenido de este enlace a un archivo .json (es texto plano) y colocarlo en la raíz del proyecto creado para el parcial

https://drive.google.com/file/d/1j6RgAe0YFA-xXxQPlEiJMReKAmAlMtQD/view?usp=sharing

Enunciado:

Crear las clases Java necesarias para guardar toda la información del archivo. Aquí deben procesar el JSON y convertirlo en clases java. **20 pts**

2. Agregar a una lista las personas que solo sean mayor o igual a 18 años. Cuando encuentren una persona menor, lanzar una excepción (creada por ustedes, no las genéricas). Dicha excepción deberá tener la edad de la persona y al mostrar la excepción, mostrar esa edad. **20 pts**

3. La lista del punto 2 deberá ser creada en una clase genérica donde deberán aplicar restricción de genericidad para acceder a los métodos pertinentes. **20 pts**

4. Desde la lista genérica pasar a un hashset todos los autores. Implementar el equals. **14 pts**

5. Desde la lista genérica pasar a un hashmap todas las etiquetas. La clave será el id de la persona y el valor un arraylist con las etiquetas. **13 pts**

Hacer un función que devuelva la cantidad de etiquetas según un dato enviado por parámetro. También se envía el id del usuario. Si la clave no existe, lanzar una excepción propia. **13 pts**

Un supermercado necesita un módulo de simulación de las filas de las cajas. Necesita calcular el tiempo medio de espera de cada cliente. Cada cliente tiene un medio de pago, un tipo de cliente y la cantidad de artículos. Cada atributo influye en el tiempo que permanecerá en caja.

- Cada artículo consume 30 segundos en la caja.
- Tipo de cliente
  - Común: 15%+
  - Jubilado: 25%+
  - Embarazada: 20%+
- Tipo de pago
  - Efectivo: 10%+
  - Tarjeta s/problemas: 15%+
  - Tarjeta c/problemas: 50%+

Debe generar los métodos y clases necesarios para contemplar nuevos tipos de pago y cliente. Los clientes generados tienen que tener variedad utilizando la función random provista por Java.

Los clientes se agrupan en una colección de cajas y cada caja tiene un comportamiento de fila. Se agrega al final, se extrae al principio. Desarrolle los métodos de la colección y de cada caja para brindar las funciones necesarias para el comportamiento básico.

Los clientes deben pasar desde una colección genérica llamada “fila_espera” hacia el arreglo de cajas. Para agrupar los clientes en las cajas se debe buscar un equilibrio tanto en la cantidad de clientes como en los tiempos de espera de cada uno.

Desarrolle un método atender() que elimine a todos los clientes de todas las cajas calculando los tiempos. Tenga en cuenta que el tiempo del segundo cliente es igual a su tiempo + el tiempo del primero. y así sucesivamente. arrojar resultados por caja y en general.

Si una caja tiene más de 3 clientes con tarjeta c/problemas lanzará una excepción creada por el programador.

Exportar el contenido de cada una de las cajas en un jsonarray.

Diseño de Clases/Herencia/Funcionamiento (30 pts)
Excepciones (diseño e implementación) (15 pts)
Genéricos (20 pts)
JSON (20 pts)
Main, comentarios, modularización de las funciones, paquetes (15 pts)