UNIVERSIDAD ICESI

Departamento de TIC
Algoritmos y Programación II
Tarea Integradora 1

Objetivos

Unidad 1: Pruebas Automáticas y Tipos de Excepción

OE1.1 Reconocer el mecanismo de manejo de excepciones señalando las implicaciones de la propagación versus el control.

OE1.2 Usar e implementar distintos tipos de excepción como parte de un programa, de manera que sea posible clasificar los tipos de error que se pueden presentar y asociarles en el programa distintas maneras de recuperarse ante el problema.

OE1.3 Diseñar pruebas unitarias automáticas que permitan validar el adecuado funcionamiento de las operaciones del sistema desarrolladas para soportar los requerimientos funcionales.

OE1.4 Desarrollar las clases y los métodos necesarios para implementar las pruebas unitarias automáticas, que ayudan a comprobar el correcto funcionamiento de un programa.

Unidad 2: Persistencia, Manejo de Archivos de Texto y Algoritmos de Ordenamiento y Búsqueda

OE2.1 Manipular archivos de texto para implementar requerimientos del cliente relacionados con persistencia.

OE2.2 Leer entradas e imprimir salidas de programas que interactúan directamente con archivos de texto y no con usuarios finales.

OE2.3 Hacer persistir el estado del modelo de solución del problema durante la ejecución de un programa y restaurarlo cuando se requiera usando la técnica de serialización.

OE2.4 Implementar algoritmos clásicos de ordenamiento de datos en estructuras de datos lineales y aplicarlos en la solución de un problema.

OE2.5 Implementar algoritmos clásicos de búsqueda de información en estructuras de datos lineales y aplicarlos en la solución de un problema.

OE2.6 Hacer uso de las interfaces Comparable y Comparator para definir relaciones de orden total sobre objetos.

OE2.7 Calcular el tiempo de ejecución de un algoritmo por medio de las operaciones de tiempo del sistema

OE2.8 Implementar métodos que permitan generar muestras con datos aleatorios.

Enunciado

Los administradores de restaurantes de la ciudad se han asociado con el objetivo de enfrentar juntos esta difícil situación que están enfrentando. El gobierno ha decretado restricciones para el ingreso físico a los restaurantes y muchas personas prefieren no hacerlo como una medida de prevención. Sin embargo, los pedidos a domicilio se han incrementado sustancialmente y se hace necesario el desarrollo de un sistema que permita gestionarlos apropiadamente. La asociación de restaurantes entiende que si tienen una plataforma común, dado que las necesidades son las mismas en este sentido, pueden hacer economía de escala.

A usted se le pide que desarrolle un prototipo de sistema de pedidos que permita el registro de restaurantes, todo restaurante tiene un nombre, nit y el nombre del administrador. Debe permitir el registro de un productos, todo producto tiene un código, nombre, una descripción, costo y nit del restaurante que ofrece dicho producto. Debe permitir el registro de un cliente, todo cliente tiene un tipo de identificación, un número de identificación, nombre completo, teléfono y dirección. Debe permitir también el registro de un pedido, todo pedido tiene un código de pedido (autogenerado), una fecha y hora (tomada del sistema y de tipo Date), el código del cliente que está haciendo el pedido, el nit del restaurante, y una lista de productos a pedir, cada producto a pedir tiene el código del producto y la cantidad.

El programa debe tener la posibilidad de actualizar los datos de un restaurante dado el nit, actualizar los datos de un producto dado su código, los datos de un cliente dado su número de documento, y los datos de un pedido dado su código. La lista de clientes debe estar ordenada siempre alfabéticamente descendente por apellido y nombre, por tanto, cada vez que se agrega un nuevo cliente, este debe insertarse de forma ordenada (tenga en cuenta que insertar de forma ordenada no es lo mismo que agregar y luego ordenar).

El programa debe permitir cambiar el estado de un pedido entre SOLICITADO, EN PROCESO, ENVIADO y ENTREGADO. Así mismo debe verificar que todos los productos de un pedido pertenezcan al restaurante que el cliente eligió para hacer el pedido. También es importante tener en cuenta que se puede cambiar el estado del pedido hacia adelante (por ejemplo de SOLICITADO a EN PROCESO, o de SOLICITADO a ENVIADO) pero no hacia atrás.

El programa debe guardar toda su información a través de la serialización de sus objetos en archivos. Este guardado debe ser transparente para el usuario del programa, que es un operario del restaurante (cada restaurante tendrá una persona encargada de ingresar información al sistema), es decir, cada vez que se registre o actualice información, esta se guardará en los archivos serializados. También se debe generar un archivo csv de pedidos, incluyendo para cada registro

de producto solicitado, los datos del restaurante, del cliente y del producto pedido. El listado debe estar ordenado por los siguientes criterios en este orden: nit del restaurante ascendente, documento del cliente descendente, fecha del pedido ascendente y código del producto ascendente. En el momento en que el usuario elija la opción de exportar esta información, debe preguntársele cuál es el separador que se utilizará. La primera línea del archivo debe tener los nombres de las columnas separadas también por dicho separador.

El programa debe tener una opción que permita listar en pantalla todos los restaurantes en orden alfabético ascendente, otra opción que permita listar en pantalla todos los clientes en orden de su número telefónico descendente.

El programa debe tener una opción que permita buscar eficientemente un cliente dado un nombre e indicar el tiempo que tardó la búsqueda.

El programa debe tener al menos 2 pruebas unitarias automáticas por cada clase del modelo.

Su programa debe implementar al menos 2 algoritmos de ordenamiento de los 3 vistos en clase (burbuja, selección e inserción). Debe hacer por lo menos 1 ordenamiento utilizando Comparable y 1 ordenamiento utilizando Comparator, utilizando en ambos casos el sort de Collections o de Arrays.

El programa debe permitir importar datos de un archivo csv con información de restaurantes, otro con información de clientes, otro con información de productos y otro con información de pedidos. Usted debe generar un archivo csv con al menos 1000 datos para probar cada uno de las opciones anteriores. Estos archivos pueden ser generados en sitios como https://www.generatedata.com/, https://www.mockaroo.com/, u otros.

Usted debe entregar los siguientes artefactos y condiciones:

1. **[10%]** Especificación de Requerimientos Funcionales.
2. **[20%]** Diseño completo del diagrama de clases, incluyendo el paquete del modelo, de la interfaz con el usuario, de excepciones y pruebas.
3. **[10%]** Diseño de todos los casos de la única prueba, incluyendo los escenarios.
4. **[50%]** Implementación completa y correcta del modelo, las excepciones, la ui y las pruebas.
5. **[10%]** Usted debe entregar el enlace del repositorio en GitHub o GitLab con los elementos anteriores. Su repositorio debe corresponder con un proyecto de eclipse. Debe tener al menos 10 commits con diferencia de 1 hora entre cada uno de ellos. En el repositorio/proyecto de eclipse debe haber un directorio llamado **docs/** en el cual deberán ir cada uno de los documentos del diseño.

Los requerimientos funcionales, el diagrama de clases y el diseño de casos de prueba deben entregarse en un mismo archivo en formato **pdf**, bien organizado por secciones y títulos, con hoja de portada.

Importante: su repositorio debe ser privado hasta la hora y fecha de entrega, después de la cual usted debe hacerlo público para que pueda ser revisado.

La rúbrica con la que se evaluará esta tarea se encuentra en el listado de notas de seguimientos y tareas integradoras en la pestaña TI1.