Cátedra de Ingeniería de Software

Código Asignatura: 00825
Asignatura: Estructura de Datos
Cuatrimestre: II Cuatrimestre 2021
Proyectos y Tareas

1 Tarea #1

Objetivo:

Que el estudiante utilice recursividad para resolver problemas comunes usando el lenguaje Java.

Explicación:

La recursividad es una técnica de programación que permite que un bloque de instrucciones se ejecute n veces. En Java se puede utilizar recursividad ya que en este lenguaje es posible que un método se invoque a sí mismo. Debe quedar claro que es distinto a una iteración, como la que utilizan otras estructuras como el for, while, etc.

Desarrollo:

Se deberá crear una aplicación Java, usando el modo gráfico, donde el usuario podrá utilizar un menú con las siguientes opciones:

1. Digitar un número entero.
2. Sumar dígitos del número ingresado por el usuario (utilizando recursividad).
3. Invertir dígitos del número ingresado (utilizando recursividad).
4. Crear un arreglo de 100 enteros generados aleatoriamente y mostrarlos.
5. Mostrar el número y posición del número mayor dentro del arreglo generado.
6. Mostrar el número y posición del número menor dentro del arreglo generado.
7. Contar los números pares del arreglo generado.
8. Contar los números impares del arreglo generado.
9. Terminar

Ejemplo:

1. El usuario digita 1234
2. La suma de los dígitos: 10
3. Número invertido: 4321

Deben incluirse validaciones para que el programa no presente errores numéricos ni de otro tipo.

Notas importantes:

1. Para el despliegue del arreglo de 100 enteros es requisito poder verlos todos, por lo que se pueden presentar seguidos separados por espacios o por algún símbolo como →. Si se presentan para abajo y superan el tamaño de la pantalla, es necesario usar una barra de desplazamiento (“scroll”).

2. Los resultados de lo solicitado en los puntos 5 a 8 se puede hacer en la misma ventana del menú o por medio del despliegue de una venta de resultados, la cual debe permitir regresar al menú, una vez que se cierre.

Rúbrica:

Aspecto a calificar	**Puntos**
Uso de menú y visualización gráfica	1
Ingreso de datos de entrada y generación del arreglo con números aleatorios.	1
Punto 2 del menú correcto y usando recursividad.	1
Punto 3 del menú correcto y usando recursividad.	1
Punto 5 del menú correcto y usando recursividad.	1
Punto 6 del menú correcto y usando recursividad.	1
Punto 7 del menú correcto y usando recursividad.	1
Punto 8 del menú correcto y usando recursividad.	1
Validaciones para evitar errores	1
TOTAL	**9**

2 Tarea #2

Objetivo:

Que el estudiante aplique el concepto de pila estática utilizando Java.

Explicación:

Una pila es una estructura de datos donde el último elemento en ingresar es el primero en salir (LIFO: Last in, First out). Por ejemplo, podemos mencionar una serie de platos sucios apilados formando una torre, donde el último plato que se coloca en la pila, el de encima, será el primero en ser lavado, y así sucesivamente hasta llegar al plato que quedó al fondo, el primero en llegar, quien precisamente será el último en ser lavado.

Desarrollo:

Escribir un programa en Java, que utilice 4 arreglos de objetos. Cada objeto es del tipo Persona con los atributos Identificación (número entero positivo de 9 dígitos) y la Edad (número entero entre 1 y 100).

El primer arreglo se trata de una lista de 50 personas generada al azar, es decir, donde los atributos Identificación y Edad son generados aleatoriamente. La identificación es única (2 o más personas no puede tener el mismo ID).

Pasos del programa:

1. Como primer paso, se genera una lista aleatoria y se muestra gráficamente en pantalla. Debe verse semejante al siguiente ejemplo:

Identificación: 123456789 – Edad: 66
Identificación: 123456780 – Edad: 41
Identificación: 223456789 – Edad: 05
Identificación: 323456789 – Edad: 04
Identificación: 423456789 – Edad: 06
Identificación: 523456789 – Edad: 99
Identificación: 623456789 – Edad: 36
Identificación: 723456789 – Edad: 74
Identificación: 823456789 – Edad: 45
...

2. El segundo paso es clasificar cada una de las personas generadas en la lista, en 3 arreglos tipo PILA, donde la primera pila contendrá a las personas menores, es decir las que tienen una edad inferior a 18. La segunda pila contendrá a las personas adultas, es decir las que tienen una edad entre 18 y 64 inclusive. La tercera pila contendrá a los adultos mayores, es decir las personas con la edad superior a 64. La clasificación de las personas, que consiste en sacar cada uno de los elementos de la lista inicial e insertarlos en la pila correspondiente, es un proceso que se realiza automáticamente, de modo que una vez realizado la pantalla mostrará gráficamente la lista y las pilas de la siguiente manera:

TODOS	MENORES	ADULTOS	ADULTOS MAYORES
Identificación: 123456789 – Edad: 66			
Identificación: 123456780 – Edad: 41			
Identificación: 223456789 – Edad: 05			
Identificación: 323456789 – Edad: 04			
Identificación: 423456789 – Edad: 06			
Identificación: 523456789 – Edad: 99			
Identificación: 623456789 – Edad: 36			**Identificación: 723456789 – Edad: 67**
Identificación: 723456789 – Edad: 74	**Identificación: 423456789 – Edad: 06**		**Identificación: 723456789 – Edad: 74**
Identificación: 823456789 – Edad: 67	**Identificación: 323456789 – Edad: 04**	**Identificación: 623456789 – Edad: 36**	**Identificación: 523456789 – Edad: 99**
…	**Identificación: 223456789 – Edad: 05**	**Identificación: 123456780 – Edad: 41**	**Identificación: 123456789 – Edad: 66**

3. Una vez realizado este proceso de generación y clasificación automática, el usuario podrá:

a. Extraer un elemento de cualquiera de las 3 pilas (menores, adultos, adultos mayores). Extraer significa que el usuario selecciona una pila y el programa debe mostrar la información del siguiente elemento de la pila seleccionada y eliminarlo de ella. Debe "refrescarse" la pila y mostrarse sin el elemento extraído.
b. Mostrar el tamaño de cualquiera de las 3 pilas, es decir, el número de elementos que tiene ingresados.
c. Vaciar cualquiera de las 3 pilas, es decir, eliminar todos sus elementos.
d. Reiniciar el proceso, es decir, limpiar la lista y las pilas y volver a generar y clasificar personas.

El programa se mostrará en ambiente gráfico (modo GUI), es decir, con un menú gráfico, pantallas, ventanas, botones y barras de desplazamiento. No se permitirá el despliegue del menú o resultados en consola.

Utilice métodos que realicen las funciones específicas que se solicitan.

Deben incluirse validaciones para que el programa siempre se ejecute correctamente y sin errores.

Nota importante:

1. El manejo de las pilas se realizará con **arreglos básicos estáticos** (cada uno de un tamaño de 50). No se permite el uso de colecciones Stack, Vector, ArrayList, LinkedList, ni ninguna otra colección dinámica de Java. Si el estudiante no cumple con esta regla su nota será de cero.

2. Los resultados de lo solicitado en los puntos 1, 2 y 3 se puede hacer en la misma ventana del menú o por medio del despliegue de una venta de resultados, la cual debe permitir regresar al menú, una vez que se cierre.

Rúbrica:

Aspecto a calificar	Puntos
Generación aleatoria de personas y visualización gráfica de la lista.	1
Clasificación de personas y visualización gráfica de las 3 pilas con sus elementos (personas).	1
Extracción de cualquiera de las pilas y visualización de la misma sin el elemento extraído.	1
Mostrar tamaño de cualquiera de las pilas.	1
Vaciar cualquiera de las pilas.	1
El programa se ejecuta sin errores	1
TOTAL	**6**

3 Proyecto #1

Objetivo:

Que el estudiante aplique al menos una técnica de ordenamiento utilizando el lenguaje Java.

Explicación:

Para mejorar el procesamiento de cualquier aplicación, se requiere en la mayoría de los casos de algoritmos de ordenamiento de los elementos involucrados en los procesos.

Existen diferentes algoritmos para esto, entre los que están:

- Ordenación por inserción.
- Shellsort.
- Mergesort.
- Quicksort.
- Burbuja.
- Selección rápida.

El ordenamiento se realiza en memoria siempre y cuando la cantidad de elementos a ordenar no sea tan extenso. Si se trata de un volumen de datos muy amplio, ya se tendría que trabajar sobre disco, bases de datos u otros medios de almacenamiento masivo.

Desarrollo:

Se debe desarrollar un programa en Java, utilizando arreglos y que realice las siguientes tareas:

1. Generar aleatoriamente, una lista de 1000 números reales en el rango de 0 a 10,000.
2. Ordenar la lista ascendentemente utilizando los siguientes 3 métodos de ordenamiento: Burbuja, Shellsort y QuickSort. Posteriormente mostrar los tiempos que tuvieron esos algoritmos para ordenarse de modo que se pueda visualizar una comparación real. Utilice los decimales que sean necesarios para mostrar las diferencias.
3. Terminar.

Notas importantes:

1. El programa se debe mostrar en modo gráfico (GUI), es decir, con menú y pantallas gráficas, no desde la consola.
2. Deben incluirse validaciones para que el programa siempre se ejecute correctamente y sin errores.
3. No se permite el uso de librerías, funciones, objetos (como arraylist) o métodos especiales propios de alguna librería o del lenguaje para realizar el ordenamiento, sino que el estudiante debe realizar las operaciones (ciclos, comparaciones, etc.) necesarias para lograr ordenar los elementos en cada algoritmo.
4. Todos los elementos a ordenar y los elementos ordenados en cada uno de los tres algoritmos se deben mostrar en una misma pantalla.
5. Se debe ofrecer la posibilidad de visualizar todos los elementos ordenados. Para esto se pueden utilizar controles de desplazamiento (“scroll”) o similares.

Rúbrica:

Aspecto a calificar	Puntos
Ordenar ascendentemente utilizando el método de la Burbuja, mostrar el tiempo empleado y brindar opción para visualizar todos los elementos utilizados en el método.	1
Ordenar ascendentemente utilizando el método Shellsort, mostrar el tiempo empleado y brindar opción para visualizar todos los elementos utilizados en el método.	1
Ordenar ascendentemente utilizando el método Quicksort, mostrar el tiempo empleado y brindar opción para visualizar todos los elementos utilizados en el método.	1
La comparación refleja las diferencias en los tiempos de ordenamiento.	1
Usa pantallas en modo gráfico (no usa consola) y muestra todos los elementos en una misma pantalla.	1
El programa se ejecuta sin errores.	1
TOTAL	**6**

4 Proyecto #2

Objetivo:

Que el estudiante aplique sus conocimientos en el uso de listas enlazadas y colas, utilizando el lenguaje java.

Explicación:

Las listas son estructuras simples de datos, que se utilizan para almacenar elementos de diferentes tipos. Las listas son ordenadas con longitud arbitraria, donde se pueden añadir o excluir elementos en cualquier ubicación.

Las listas pueden implementarse utilizando arreglos o por medio de listas enlazadas sencillas, donde cada enlace contiene el elemento de la lista así como el puntero al siguiente elemento. Cuando no se tiene claro cuántos elementos se va a requerir almacenar, es cuando se piensa que lo mejor es utilizar listas enlazadas.

Desarrollo:

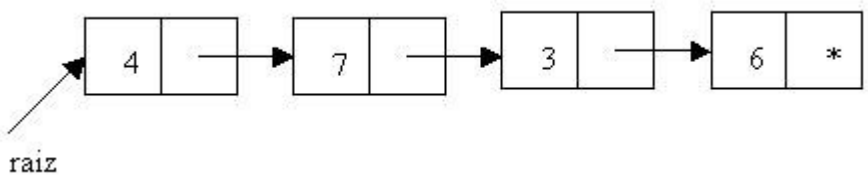
Se requiere crear una cola utilizando una lista enlazada, es decir, con nodos donde cada uno tendrá un elemento con la información y otro con un puntero al siguiente elemento. La información de cada elemento será un número entero positivo o negativo.

Es necesario crear un menú con las siguientes opciones:

1. Insertar elemento en la cola.
2. Extraer un elemento de la cola. En este caso, se muestra el número y se elimina.
3. Obtener el elemento de la cola en la posición que indique el usuario. Se muestra su valor, sin extraerlo.
4. ¿La cola está vacía?. En esta opción solo se indica SI o NO.
5. Tamaño actual de la cola. Para esto es necesario contar el número de elementos en la cola o de nodos en la lista enlazada.
6. Mostrar todos los elementos de la cola. Desde el primer elemento hasta el último.
7. Vaciar la cola, es decir, eliminar todos los nodos.
8. Terminar.

Notas importantes:

1. Este programa debe ser creado utilizando el modo gráfico o GUI, donde el menú y las pantallas son gráficos, por lo que no se permite utilizar la consola para el despliegue del menú o los resultados.
2. La opción *6. Mostrar todos los elementos de la cola*, mostrará en pantalla la cola con formas gráficas, es decir, podrá utilizar librerías que dibujan rectángulos y líneas o flechas. Si se muestra usando texto, o con un objeto lista o grid, perderá los puntos correspondientes según la rúbrica. El siguiente ejemplo muestra cómo se debe presentar la cola (ejemplo de cola con 3 elementos).



3. Deben incluirse validaciones para que el programa siempre se ejecute correctamente y sin errores.
4. No se debe utilizar ArrayList, LinkedList ni ninguna colección de Java ya implementada, sino implementar listas enlazadas usando nodos.
5. Los resultados de lo solicitado en los puntos 1 a 7 del menú se pueden presentar en la misma ventana del menú o por medio del despliegue de una venta de resultados, la cual debe permitir regresar al menú, una vez que se cierre.

Rúbrica:

Aspecto a calificar	Puntos
Insertar elemento en cola	1
Extrae elemento de la cola	1
Obtener elemento en determinada posición	1
Indicador de cola vacía	1
Tamaño de cola	1
Mostrar elementos de la cola	1
Mostrar cola gráfica según enunciado	2
Vaciar cola	1
El programa se ejecuta sin errores	1
TOTAL	**10**