| Facultad de Ingeniería - Universidad Nacional de Cuyo | | | |
|---|---|---|---|
| Asignatura: | Programación II | | |
| Carrera: | Licenciatura en Ciencias de la Computación | | |
| Año: 2024 | Semestre: 2 | Trabajo Práctico n° 3 | |

**Trabajo Práctico n° 3: Gestión de Lista de Tareas utilizando ArrayList**

**Objetivos:**

- Introducir a los estudiantes al uso de ArrayList en Java.
- Aplicar diferentes operaciones con ArrayList y practicar la manipulación de listas de objetos.
- Comprender el uso de iteradores y las operaciones avanzadas sobre colecciones.

**Metodología**

- Trabajo individual.
- Tiempo de realización estimado: 1 semana.

**Contenido**

Definición y características principales:

- **ArrayList**:
  - Colección dinámica que permite almacenar y manipular objetos de manera eficiente.
  - Métodos y operaciones básicas.

Conceptos sobre operaciones de colecciones:

- **Iterator**:
  - Definición y propósito.
  - Uso de iteradores para recorrer colecciones.

Operaciones básicas con ArrayList:

- **Métodos de ArrayList**:
  - add: Añadir elementos a la lista.
  - size: Obtener el tamaño de la lista.

- get: Acceder a elementos específicos.
- contains: Verificar si un elemento está en la lista.
- indexOf: Obtener la primera posición de un elemento.
- lastIndexOf: Obtener la última posición de un elemento.
- remove: Eliminar elementos.
- clear: Vaciar la lista.
- isEmpty: Verificar si la lista está vacía.
- clone: Crear una copia de la lista.

**Actividades:**

1. **Definición de la clase Tarea**:
   - Implementar una clase **Tarea** que contenga los siguientes atributos:
     - **nombre:** Nombre de la tarea.
     - **descripcion:** Descripción de la tarea.
     - **completada:** Booleano que indica si la tarea está completada o no.
   - Agregar métodos para completar la tarea y para mostrar su información.
2. **Operaciones con ArrayList**:
   - Implementar una clase **ListaTareas** que gestione una lista de objetos Tarea utilizando ArrayList. Debe contener los siguientes métodos:
     - **agregarTarea:** Añadir nuevas tareas a la lista.
     - **mostrarTareas:** Mostrar todas las tareas utilizando un Iterator.
     - **obtenerTarea:** Obtener una tarea específica usando get.
     - **contieneTarea:** Verificar si una tarea está en la lista usando contains.
     - **indiceTarea:** Obtener la posición de una tarea usando indexOf o lastIndexOf.
     - **eliminarTarea:** Eliminar tareas de la lista, ya sea por índice o por el objeto.
     - **limpiarLista:** Vaciar toda la lista usando clear.
     - **estaVacia:** Verificar si la lista está vacía usando isEmpty.
     - **clonarLista:** Crear una copia de la lista utilizando clone.
3. **Pruebas y validaciones**:
   - Escribir una clase **Main** donde se prueben todas las operaciones anteriormente mencionadas:
     - Agregar al menos 3 tareas a la lista.
     - Mostrar todas las tareas.
     - Buscar si una tarea específica está en la lista.
     - Eliminar una tarea por su índice y luego por el objeto.
     - Verificar si la lista está vacía después de vaciarla.
     - Clonar la lista y trabajar con la copia.

**Evaluación**:

- El código debe compilar y funcionar correctamente.
- Los métodos add, size, get, contains, indexOf, lastIndexOf, remove, clear, isEmpty, clone y Iterator deben ser implementados correctamente.
- El manejo de la lista debe ser eficiente y evitar redundancias en el código.
- La presentación de este trabajo práctico será en forma personal a través de la plataforma aula abierta y deberán realizar una presentación y prueba del código frente al docente.