Desenho de personagem de desenho animado

Descrição gerada automaticamente com confiança média

Curso: CST em Análise e Desenvolvimento de Sistemas

Unidade Curricular: Programação Orientada à Objetos

Turma: GR SADS 2023/1 N1 – 2º semestre

Acadêmico: Eduardo Adão Locks

**ArrayList’s Atividade Teórica**

1. O que são ArrayList’s?
   1. Um ArrayList é uma lista flexível que você pode usar na programação para guardar várias coisas juntas. É como uma lista de compras que você pode adicionar ou retirar itens a qualquer momento, e não precisa saber quantos itens terá com antecedência. É uma maneira útil de organizar informações.
2. Quais as vantagens do uso do ArrayList em relação a arrays tradicionais?
   1. Tamanho Dinâmico: Ao contrário dos arrays tradicionais, que têm um tamanho fixo, os ArrayLists podem crescer ou diminuir conforme necessário. Isso significa que você não precisa se preocupar em definir um tamanho máximo para sua coleção de dados.
   2. Inclusão e Remoção Simples: Adicionar e remover elementos em um ArrayList é mais fácil e conveniente do que em um array tradicional. Você não precisa se preocupar em mover manualmente os elementos quando adiciona ou remove algo.
   3. Gerenciamento de Memória Automático: Os ArrayLists gerenciam automaticamente a memória para você, alocando espaço adicional quando necessário e liberando espaço quando elementos são removidos. Nos arrays tradicionais, você precisa gerenciar a alocação de memória manualmente.
   4. Flexibilidade de Tipos: Em muitas linguagens, os ArrayLists podem armazenar diferentes tipos de objetos na mesma lista, o que é útil quando você precisa trabalhar com dados heterogêneos. Arrays tradicionais geralmente requerem que todos os elementos sejam do mesmo tipo.
   5. Facilidade de Uso: Os ArrayLists oferecem uma interface de programação mais amigável com métodos convenientes para adicionar, remover e consultar elementos. Isso torna o código mais legível e fácil de manter.
   6. Redimensionamento Automático: Quando um ArrayList atinge sua capacidade máxima, ele pode ser automaticamente redimensionado para acomodar mais elementos, enquanto em um array tradicional você precisaria criar um novo array com um tamanho maior manualmente.
   7. Menos Preocupação com Índices: Como os ArrayLists lidam com a expansão e contração dinâmica, você não precisa se preocupar tanto com a gestão de índices como nos arrays tradicionais.
3. Quais conteúdos são colecionáveis?
   1. Números, Texto, Objetos, Booleanos, Caracteres, Bytes, Decimais, Octais, Hexadecimais, Binários, Float, Inteiros, Elementos Mistos, Referências de Objetos, Elementos de Dados Compostos e Dados Multidimensionais.
4. Quais os principais métodos da classe ArrayList e suas funções?
   1. add(E elemento): Adiciona o elemento especificado ao final da lista.
   2. add(int índice, E elemento): Insere o elemento especificado na posição indicada pelo índice.
   3. get(int índice): Retorna o elemento na posição especificada pelo índice.
   4. set(int índice, E elemento): Substitui o elemento na posição especificada pelo índice pelo elemento fornecido.
   5. remove(int índice): Remove o elemento na posição especificada pelo índice e reorganiza a lista.
   6. remove(Object objeto): Remove a primeira ocorrência do objeto especificado na lista, se presente.
   7. size(): Retorna o número de elementos na lista.
   8. isEmpty(): Retorna verdadeiro se a lista estiver vazia, caso contrário, retorna falso.
   9. clear(): Remove todos os elementos da lista, tornando-a vazia.
   10. contains(Object objeto): Retorna verdadeiro se a lista contiver o objeto especificado, caso contrário, retorna falso.
   11. indexOf(Object objeto): Retorna o índice da primeira ocorrência do objeto especificado na lista. Se não encontrar, retorna -1.
   12. lastIndexOf(Object objeto): Retorna o índice da última ocorrência do objeto especificado na lista. Se não encontrar, retorna -1.
   13. toArray(): Converte a lista em um array.
   14. toArray(T[] array): Converte a lista em um array do tipo especificado, se possível.
   15. addAll(Collection<? extends E> coleção): Adiciona todos os elementos da coleção especificada à lista.
   16. addAll(int índice, Collection<? extends E> coleção): Insere todos os elementos da coleção especificada na posição indicada pelo índice.
   17. removeAll(Collection<?> coleção): Remove todos os elementos da lista que estão contidos na coleção especificada.
   18. retainAll(Collection<?> coleção): Mantém na lista apenas os elementos que estão contidos na coleção especificada, removendo os outros.
   19. subList(int doInicio, int doFim): Retorna uma sublista da lista original que vai do índice doInicio (inclusive) até o índice doFim (exclusive).
   20. iterator(): Retorna um iterador para percorrer os elementos da lista.
   21. forEach(Consumer<? super E> ação): Executa uma ação em cada elemento da lista usando uma expressão lambda ou referência a método.