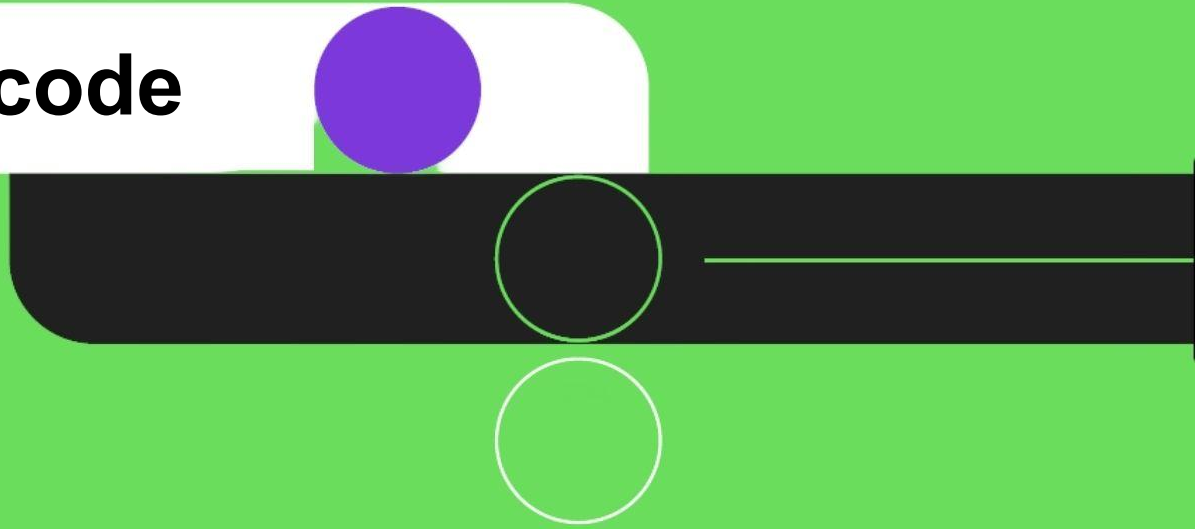
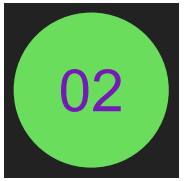
GAMA ACADEMY

Coleções

Equals e Hashcode



Equals



Collections

SET

LIST

Equals

O método equals é utilizado para comparações.

Classes por referência sobrescrevem equals() para garantir que dois objetos analisados, com o mesmo conteúdo, possam ser considerados iguais. E, quando a classe a qual os objetos em questão pertencem não sobrescreve o método Equals(), o método Object.Equals() será chamado.

Quando comparamos objetos, é considerada uma boa prática utilizarmos o método Equals() para fazer a comparação de igualdade.

Collections

A Java API fornece várias estruturas de dados pré definidas, chamadas **coleções**, usadas para armazenar grupos de objetos relacionados na memória.

Essas classes fornecem métodos eficientes que organizam, armazenam e recuperam seus dados sem a necessidade de conhecer como os dados são armazenados. Isso reduz o tempo de desenvolvimento de aplicativos.

Collections

SET	Não ordenado (por padrão) Não indexado Não aceita repetição
LIST	Indexada Aceita repetição
MAP	Chave/Valor Chave não aceita repetição Valor aceita repetição
QUEUE	Implementa fila Fist in/ Fist out(FIFO)
STACK	Implementa pilha(stack) Last in/ Fist out(LIFO)

Collections

Interface	Descrição
Collection	A interface-raiz na hierarquia de coleções a partir da qual as interfaces Set, Queue e List são derivadas.
Set	Uma coleção que não contém duplicatas.
List	Uma coleção ordenada que pode conter elementos duplicados.
Queue(fila)	Em geral, uma coleção primeiro a entrar, primeiro a sair que modela uma fila de espera; outras ordens podem ser especificadas.
Map	Uma coleção que associa chaves a valores e que não pode conter chaves duplicadas. Map não deriva de Collection.

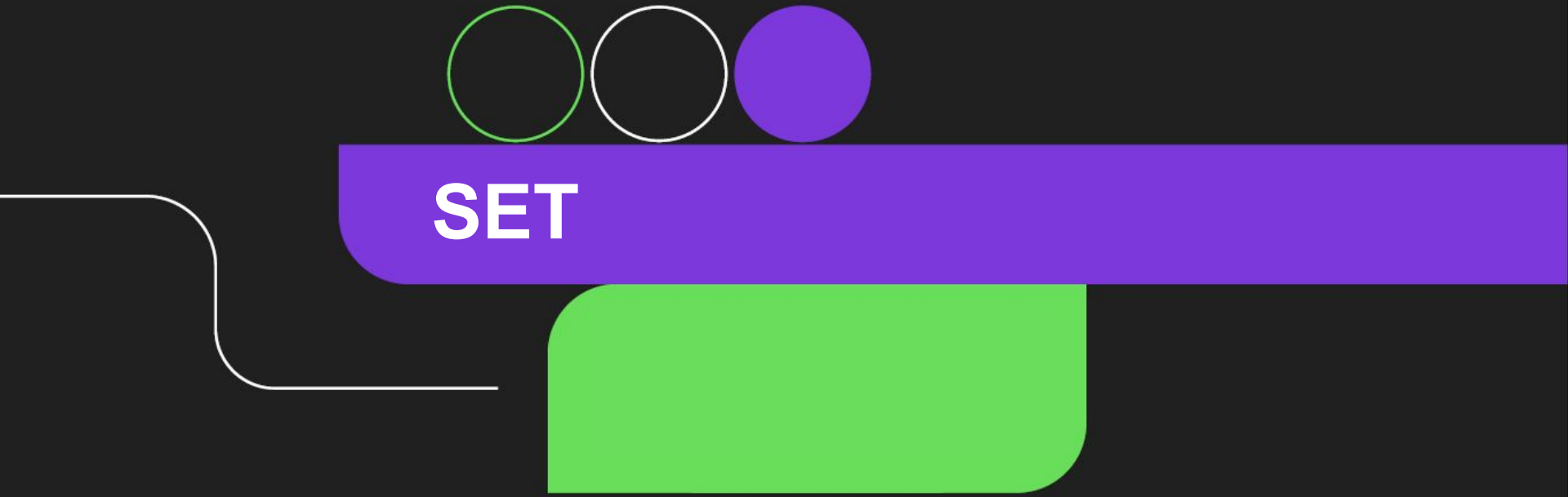
Collections

Escolhendo uma coleção

A documentação para cada coleção discute os requisitos de memória e as características de desempenho dos métodos para operações como adição e remoção de elementos, pesquisa de elementos, classificação de elementos etc.

Antes de escolher uma coleção, revise a documentação on-line para a categoria da coleção que você está considerando (Set, List, Map, Queue etc.), então selecione a implementação que melhor atende às necessidades de seu aplicativo.

documentação (acesso em 10/2021)


SET

SET

Características - Set

- Velocidade na pesquisa de dados, sendo mais rápida que um objeto do tipo List;
- A inserção de dados é mais lenta;
- Permite trabalhar com conjuntos e pode ser implementado como instâncias das classes HashSet ou TreeSet;
- Não precisa especificar a posição para adicionar um elemento;
- Não aceita valores duplicados. Se caso inserir um registro que já tenha no Set não será adicionado.
- Podem ser implementados como instâncias das classes HashSet ou TreeSet;

SET

Declaração - Set

Quando está construindo objetos na classe Set é necessário informar que tipo de coleção será implementada.

Sintaxe: **Set set = new Type()**;

E - é o objeto declarado, podendo ser classes Wrappers ou tipo de coleção.
Type - é o tipo de objeto da coleção a ser usado;

LIST

LIST

Características - List

Uma coleção ordenada (também conhecida como *sequência*). O usuário desta interface tem controle preciso sobre onde na lista cada elemento é inserido.
O usuário pode acessar os elementos por seu índice inteiro (posição na lista) e pesquisar os elementos na lista.
A interface List coloca estipulações adicionais, além das especificadas na interface Collection , nos contratos dos métodos iterador , add , remove , equals e hashCode .

Declaração - List

```
ArrayList<Usuario> lista = new ArrayList<>();
```

GAMA ACADEMY

VAMOS PRATICAR?