

Programação Orientada a Objetos - POOS3

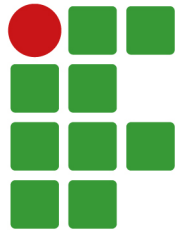
1

Tecnologia em Análise e Desenvolvimento de Sistemas

Aula 8

Foreach, varargs, collections

2º semestre de 2018



INSTITUTO FEDERAL

São Paulo

Câmpus Araraquara

Problema

- Um aluno possui quatro notas, prontuário e nome.
- Cada nota equivale a uma porcentagem. As porcentagens devem somar 100%.
- Implementar um sistema que leia as notas de um aluno e apresente a média.

```
package model;

public class Nota {

    private double nota;
    private double porcentagem;

    public Nota(double nota, double porcentagem) {
        super();
        this.nota = nota;
        this.porcentagem = porcentagem;
    }

    public double parcial(){
        return nota * (porcentagem/100.0);
    }

    public double getNota() {
        return nota;
    }

    public void setNota(double nota) {
        this.nota = nota;
    }

    public double getPorcentagem() {
        return porcentagem;
    }

    public void setPorcentagem(double porcentagem) {
        this.porcentagem = porcentagem;
    }
}
```

```
package model;
```

```
public class Aluno {
    private int prontuario;
    private String nome;
    private Nota[] notas;

    public Aluno(int prontuario, String nome) {
        this.prontuario = prontuario;
        this.nome = nome;
        notas = new Nota[4];
    }

    public void setNotas(double nota1, double nota2, double nota3,
        double nota4, double peso1, double peso2, double peso3, double peso4){
        if(peso1 + peso2 + peso3 + peso4 == 100){
            notas[0] = new Nota(nota1, peso1);
            notas[1] = new Nota(nota2, peso2);
            notas[2] = new Nota(nota3, peso3);
            notas[3] = new Nota(nota4, peso4);
        }else{
            notas[0] = new Nota(-1, -1);
            notas[1] = new Nota(-1, -1);
            notas[2] = new Nota(-1, -1);
            notas[3] = new Nota(-1, -1);
        }
    }

    public double media(){
        double soma=0;
        for(int i=0; i < 4; i++){
            soma += notas[i].parcial();
        }
        return soma;
    }
}
```

```
public Nota get(int nota){
    Nota retorno = null;
    if(nota >= 0 && nota <= 3){
        retorno = notas[nota];
    }
    return retorno;
}

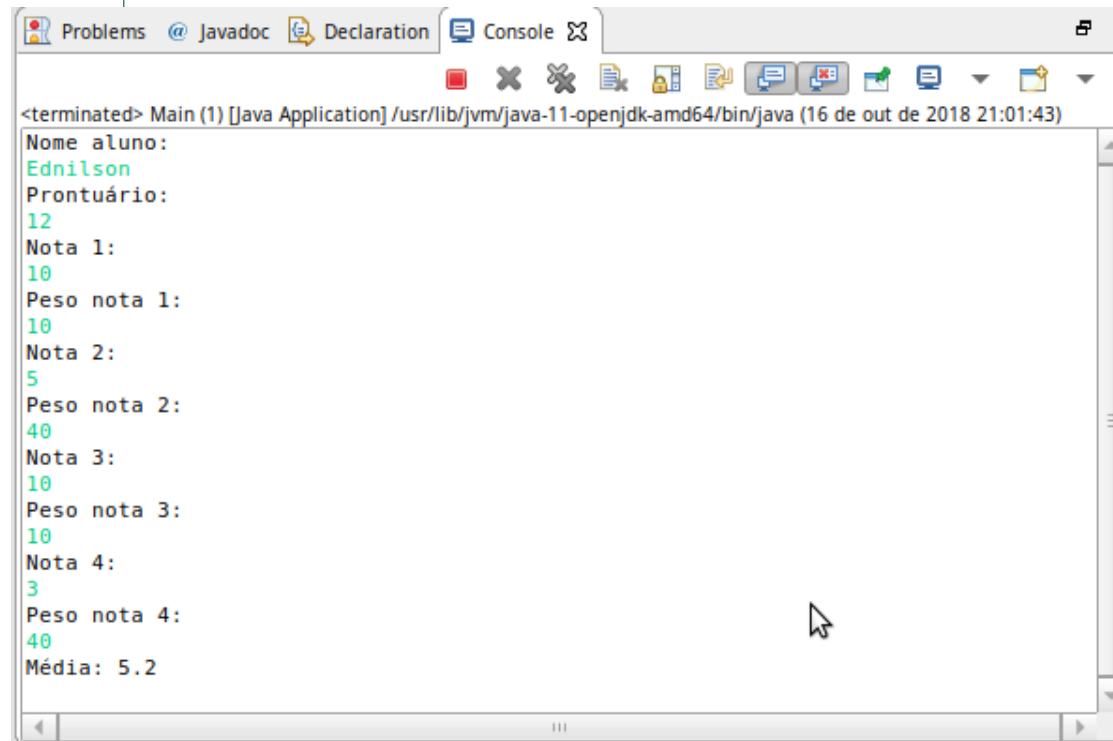
public int getProntuario() {
    return prontuario;
}

public void setProntuario(int prontuario) {
    this.prontuario = prontuario;
}

public String getNome() {
    return nome;
}

public void setNome(String nome) {
    this.nome = nome;
}
}
```

```
public class Main {  
    public static void main(String[] args) {  
        Scanner scanner = new Scanner(System.in);  
        String nome;  
        int prontuario;  
        double n1, n2, n3, n4, p1, p2, p3, p4;  
        Aluno estudante;  
        System.out.println("Nome aluno: ");  
        nome = scanner.nextLine();  
        System.out.println("Prontuário: ");  
        prontuario = scanner.nextInt();  
        System.out.println("Nota 1: ");  
        n1 = scanner.nextDouble();  
        System.out.println("Peso nota 1: ");  
        p1 = scanner.nextDouble();  
        System.out.println("Nota 2: ");  
        n2 = scanner.nextDouble();  
        System.out.println("Peso nota 2: ");  
        p2 = scanner.nextDouble();  
        System.out.println("Nota 3: ");  
        n3 = scanner.nextDouble();  
        System.out.println("Peso nota 3: ");  
        p3 = scanner.nextDouble();  
        System.out.println("Nota 4: ");  
        n4 = scanner.nextDouble();  
        System.out.println("Peso nota 4: ");  
        p4 = scanner.nextDouble();  
        estudante = new Aluno(prontuario, nome);  
        estudante.setNotas(n1, n2, n3, n4, p1, p2, p3, p4);  
        System.out.println("Média: " + estudante.media());  
    }  
}
```



```
<terminated> Main (1) [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (16 de out de 2018 21:01:43)  
Nome aluno:  
Ednilson  
Prontuário:  
12  
Nota 1:  
10  
Peso nota 1:  
10  
Nota 2:  
5  
Peso nota 2:  
40  
Nota 3:  
10  
Peso nota 3:  
10  
Nota 4:  
3  
Peso nota 4:  
40  
Média: 5.2
```

```
package model;
```

```
public class Aluno {
    private int prontuario;
    private String nome;
    private Nota[] notas;

    public Aluno(int prontuario, String nome) {
        this.prontuario = prontuario;
        this.nome = nome;
        notas = new Nota[4];
    }

    public void setNotas(double nota1, double nota2, double nota3,
        double nota4, double peso1, double peso2, double peso3, double peso4){
        if(peso1 + peso2 + peso3 + peso4 == 100){
            notas[0] = new Nota(nota1, peso1);
            notas[1] = new Nota(nota2, peso2);
            notas[2] = new Nota(nota3, peso3);
            notas[3] = new Nota(nota4, peso4);
        } else {
            notas[0] = new Nota(-1, -1);
            notas[1] = new Nota(-1, -1);
            notas[2] = new Nota(-1, -1);
            notas[3] = new Nota(-1, -1);
        }
    }

    public double media(){
        double soma=0;
        for(int i=0; i < 4; i++){
            soma += notas[i].parcial();
        }
        return soma;
    }
}
```

Lembrando: Um array é uma variável composta homogênea que utiliza alocação estática de memória, ou seja, uma variável que armazena vários “objetos”, organizados sequencialmente na memória.



Foreach

- **Para cada** – Pode-se percorrer cada um dos elementos de um conjunto usando o comando “foreach”.

```
public double media(){  
    double soma=0;  
    for(int i=0; i < 4; i++){  
        soma += notas[i].parcial();  
    }  
    return soma;  
}
```

```
public double media(){  
    double soma=0;  
    for(Nota n : notas){  
        soma += n.parcial();  
    }  
    return soma;  
}
```

O resultado final é o mesmo.

Observe que para cada uma das notas no array será feita uma referência nota, e essa pode ser utilizada da forma que for necessário.



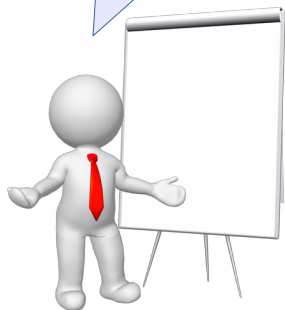
Varargs

- Lista de Argumentos de Comprimento Variável
- Um tipo de parâmetro seguido por reticências(...) na lista de parâmetros indica que o método recebe um número variável de argumentos desse tipo particular.
- No corpo do método, a lista de parâmetros de tamanho variável é vista como um array (disponibilizando para cada argumento uma posição de armazenamento).

Definiu-se que argumentos é um array do tipo double, mas não sabe-se o tamanho desse array.

O método pode verificar o tamanho do array e tomar sua ação.

Esse exemplo não foi o mais apropriado, veremos outro adiante.



```
public void setNotas(double nota1, double nota2, double nota3,
    double nota4, double peso1, double peso2, double peso3, double peso4){
    if(peso1 + peso2 + peso3 + peso4 == 100){
        notas[0] = new Nota(nota1, peso1);
        notas[1] = new Nota(nota2, peso2);
        notas[2] = new Nota(nota3, peso3);
        notas[3] = new Nota(nota4, peso4);
    }else{
        notas[0] = new Nota(-1, -1);
        notas[1] = new Nota(-1, -1);
        notas[2] = new Nota(-1, -1);
        notas[3] = new Nota(-1, -1);
    }
}
```

```
public void setNotas(double... argumentos) {
    int soma = 0;
    notas[0] = new Nota(-1, -1);
    notas[1] = new Nota(-1, -1);
    notas[2] = new Nota(-1, -1);
    notas[3] = new Nota(-1, -1);
    if (argumentos.length == 8) {
        for (int i = 4; i < argumentos.length; i++) {
            soma += argumentos[i];
        }
        if (soma == 100) {
            for (int i = 0; i < 4; i++) {
                notas[i].setNota(argumentos[i]);
                notas[i].setPorcentagem(argumentos[i + 4]);
            }
        }
    }
}
```

Varargs – outro exemplo

- Calcular a média aritmética de vários números.'

```
public static final double mediaAritmetica(int... numeros){  
    int soma=0;  
    for(int i:numeros){  
        soma += i;  
    }  
    return soma / numeros.length;  
}
```

Legal!!!

Na chamada é
passado um array?



```
int nros[] = {10, 5, 9, 14, 100, 200};  
System.out.println("Média: " + Aluno.mediaAritmetica(nros));
```

```
System.out.println("Média: " + Aluno.mediaAritmetica(10, 5, 9, 14, 100, 200));
```

A chamada pode ser por meio de um array ou por uma lista de argumentos separados por virgula. O exemplo estão os valores definidos, porém é possível utilizar variáveis do tipo inteiro.



ArrayList

- É um tipo de objeto que permite armazenar uma **coleção** de dados. Esses dados podem ser acessados por sua posição no ArrayList.
- Muito semelhante a um Array, porém possui algumas vantagens:
 - Não tem tamanho fixo;
 - Gerenciamento da lista (inserção, remoção, etc) é controlado pelo objeto;
 - Suporte ao generics;
 - Outros.

```

package model;

import java.util.ArrayList;

public class Estudante {
    private int prontuario;
    private String nome;
    private ArrayList<Nota> notas;

    public Estudante(int prontuario, String nome) {
        this.prontuario = prontuario;
        this.nome = nome;
        notas = new ArrayList<>();
    }

    public void addNota(double valor, double porcentagem){
        notas.add(new Nota(valor, porcentagem));
    }

    public double media(){
        double soma=0;
        for(Nota n : notas){
            soma += n.parcial();
        }
        return soma;
    }

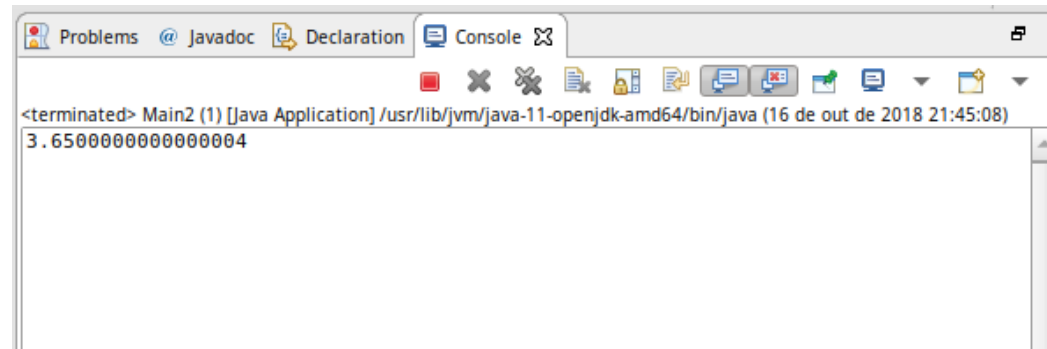
    /*gets e sets*/
}

```

```

public class Main2 {
    public static void main(String args[]) {
        Estudante estudante;
        estudante = new Estudante(123, "Ednilson");
        estudante.addNota(5, 15);
        estudante.addNota(10, 5);
        estudante.addNota(3, 80);
        System.out.println(estudante.media());
    }
}

```



The screenshot shows an IDE window with a console tab selected. The console output displays the result of the program execution: 3.6500000000000004. The title bar of the window indicates it is a Java application running on a specific JVM.

```

<terminated> Main2 (1) [Java Application] /usr/lib/jvm/java-11-openjdk-amd64/bin/java (16 de out de 2018 21:45:08)
3.6500000000000004

```

```
package model;

import java.util.ArrayList;

public class Estudante {
    private int prontuario;
    private String nome;
    private ArrayList<Nota> notas;

    public Estudante(int prontuario, String nome) {
        this.prontuario = prontuario;
        this.nome = nome;
        notas = new ArrayList<>();
    }

    public void addNota(double valor, double porcentagem){
        notas.add(new Nota(valor, porcentagem));
    }

    public double media(){
        double soma=0;
        for(Nota n : notas){
            soma += n.parcial();
        }
        return soma;
    }

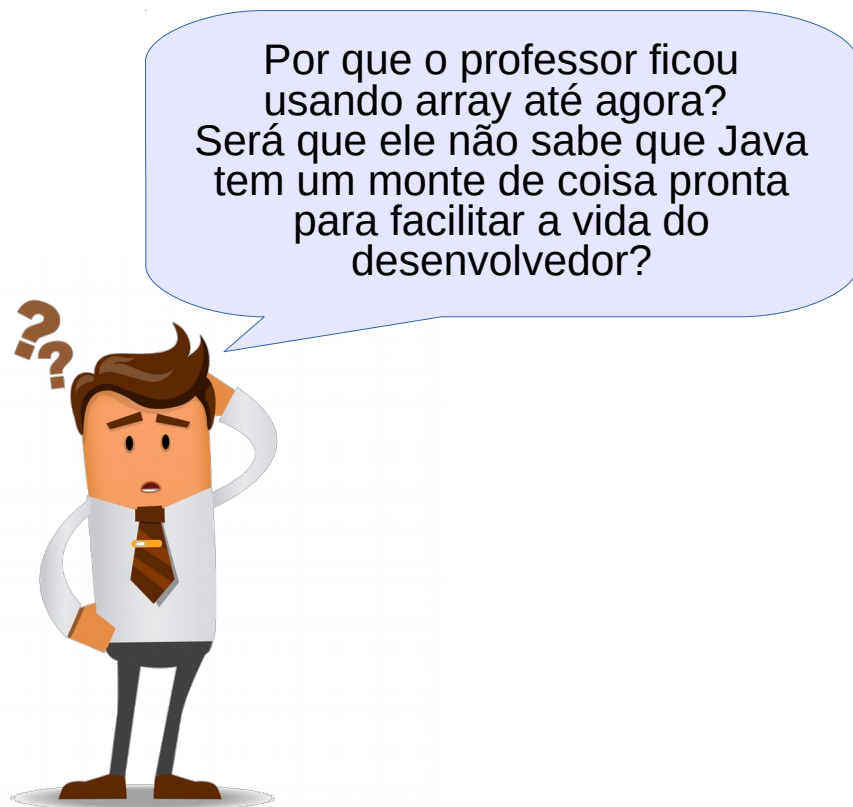
    /*gets e sets*/
}
```

Definiu-se um
ArrayList de Nota

Instância do objeto,
notas é uma lista de
vários objetos Nota.

Uma das principais vantagens é que
não é necessário se preocupar com
qual posição armazenar o novo
objeto, essa funcionalidade é
resolvida pelo ArrayList.

O foreach também resolver o problema com ArrayList. Na verdade o
foreach foi criado para ArrayList (Collections) e depois adaptado para
array.
Contudo, o método tradicional também funciona!



Como funciona um ArrayList

- Este tipo de lista é implementado como um Array que é dimensionado dinamicamente, ou seja, sempre que é necessário o seu tamanho aumenta em 50% do tamanho da lista.
- Significa que se você tiver uma lista de tamanho igual a 10 e ela “encher”, seu tamanho aumentará para 15 automaticamente.
- O custo para aumentar o tamanho de um ArrayList é alto, pois é feita uma cópia do array atual para um novo array com um novo tamanho.
- Imagine um array com 10mil elementos que será copiado para um novo array para criação de mais 5 mil elementos? De fato é um alto custo.
- É altamente aconselhável que você já inicie seu Array com uma quantidade de elementos que atenda ao seu objetivo atual, sem a necessidade de criação dinâmica de novos espaços, ou seja, se você souber que terá que armazenar de 300 a 400 objetos em um Array, defina 500, pois é melhor sobrar espaço do que utilizar recurso do processador sem necessidade.

Exemplo

```
public Estudante(int prontuario, String nome) {  
    this.prontuario = prontuario;  
    this.nome = nome;  
    notas = new ArrayList<>(4);  
}
```

Ainda é vantagem
usar ArrayList !

Existe alguma outra
restrição?

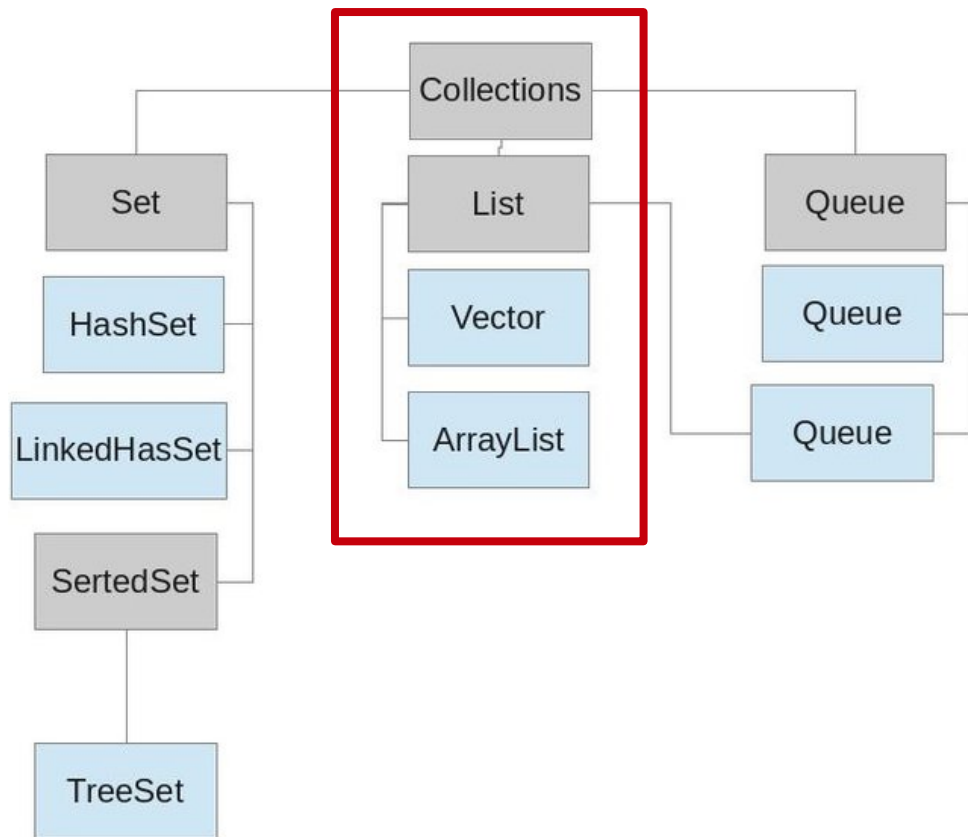
Alternativas?

Não é adequado para
variáveis de tipo primitivo (int,
boolean, double). Sempre que
é usamos existe uma
conversão para um objeto
equivalente. O programador
não vê, mas o custo também
é alto.

ArrayList não é sincronizado,
quando estudar Multi-Thread
entenderemos melhor isso.



API Java Collection



List é INTERFACE!!!



Quero instanciar List !
Quero instanciar List !



Vector

- A classe Vector é muito similar a ArrayList, porem é preciso esta atendo em algumas diferenças entre Ambas:
 - Vector é sincronizada (Podemos implementa Thread-Safe utilizando-a).
 - ArrayList cresce automaticamente nossa lista em 50%, já classe Vector aumenta o dobro, ou seja se temos uma lista com tamanho 20 utilizando Vector e caso queiramos aumentá-la esta lista vai fica com tamanho 40.

Vector

```
public class Estudante {
    private int prontuario;
    private String nome;

    private Vector<Nota> notas;

    public Estudante(int prontuario, String nome) {
        this.prontuario = prontuario;
        this.nome = nome;

        notas = new Vector<>(4);
    }

    public void limparNotas(){
        notas.clear();
    }

    public void addNota(double valor, double porcentagem){
        notas.add(new Nota(valor, porcentagem));
    }

    public void addNota(int nota, double valor, double porcentagem){
        notas.add(nota, new Nota(valor, porcentagem));
    }
}
```

```
@Override
public String toString(){
    StringBuilder sb = new StringBuilder();
    sb.append("Aluno: ");
    sb.append(getNome());
    sb.append("\nProntuário: ");
    sb.append(getProntuario());
    sb.append("\nNotas:");
    for (Nota n: notas) {
        sb.append("\nValor: ");
        sb.append(n.getNota());
        sb.append("\tPorcentagem: ");
        sb.append(n.getPorcentagem());
        sb.append("%");
    }
    return sb.toString();
}

public double media(){
    double soma=0;
    for(Nota n : notas){
        soma += n.parcial();
    }
    return soma;
}...
```

LinkedList

Alguém, explicar o motivo, deve!



20

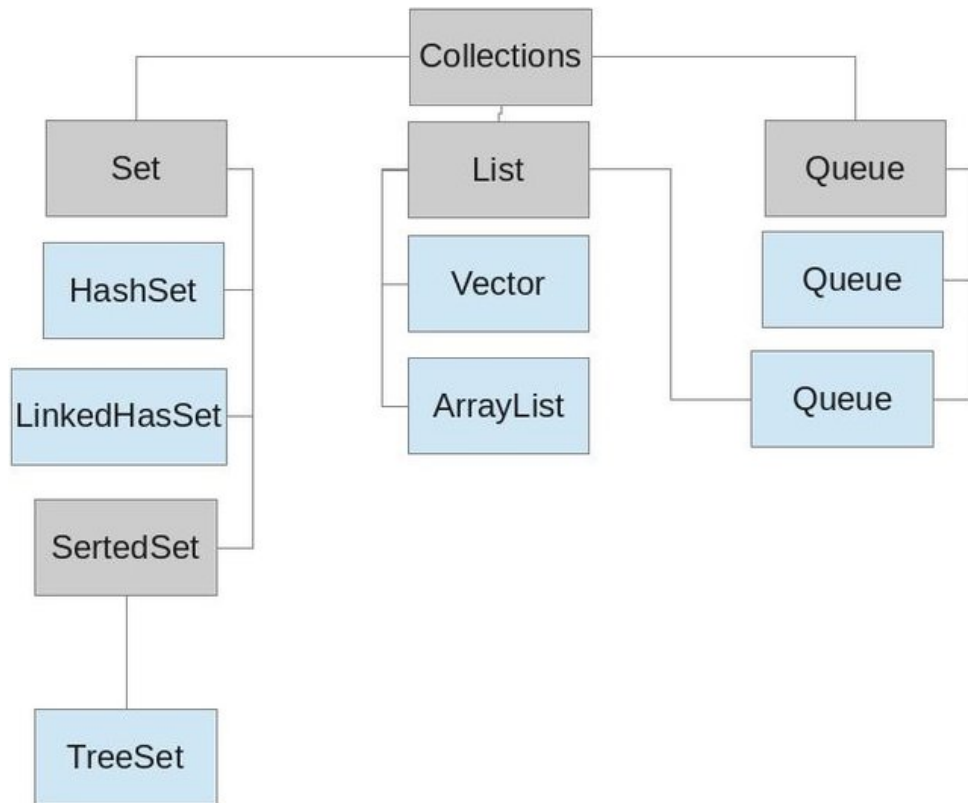
- Implementa uma “double linked list”, ou seja, uma lista duplamente encadeada.
- A sua principal diferença entre o ArrayList é na performance entre os métodos add, remove, get e set.
- Possui melhor performance nos métodos add e remove, do que os métodos add e remove do ArrayList, em compensação seus métodos get e set possuem uma performance pior do que os do ArrayList.
- Vamos fazer uma comparação entre a complexidade apresentada de cada método do ArrayList e o da LinkedList.
 - get(int index): LinkedList possui $O(n)$ e ArrayList possui $O(1)$
 - add(E element): LinkedList possui $O(1)$ e ArrayList possui $O(n)$ no pior caso, visto que o array será redimensionado e copiado para um novo array.
 - add(int index, E element): LinkedList possui $O(n)$ e ArrayList possui $O(n)$ no pior caso
 - remove(int index): LinkedList possui $O(n)$ e ArrayList possui $O(n - \text{index})$, se remover o último elemento então fica $O(1)$

LinkedList

```
public class Estudante {  
    private int prontuario;  
    private String nome;  
    private LinkedList<Nota> notas;  
    public Estudante(int prontuario, String nome) {  
        this.prontuario = prontuario;  
        this.nome = nome;  
  
        notas = new LinkedList<>();  
    }  
    public void limparNotas(){  
        notas.clear();  
    }  
    public void addNota(double valor, double percentagem){  
        notas.add(new Nota(valor, percentagem));  
    }  
    public void addNota(int nota, double valor, double percentagem){  
        notas.add(nota, new Nota(valor, percentagem));  
    }  
}
```

Única diferença é que não se define o tamanho inicial. Não teria sentido fazer isso para uma lista encadeada.

API Java Collection



Quero instanciar List !
Quero instanciar List !



```
public class Estudante {  
    private int prontuario;  
    private String nome;  
    private List<Nota> notas;  
    public Estudante(int prontuario, String nome) {  
        this.prontuario = prontuario;  
        this.nome = nome;  
        notas = new ArrayList<>(4);  
    }  
}
```

```
public class Estudante {  
    private int prontuario;  
    private String nome;  
    private List<Nota> notas;  
    public Estudante(int prontuario, String nome) {  
        this.prontuario = prontuario;  
        this.nome = nome;  
        notas = new Vector<>(4);  
    }  
}
```

```
public class Estudante {  
    private int prontuario;  
    private String nome;  
    private List<Nota> notas;  
    public Estudante(int prontuario, String nome) {  
        this.prontuario = prontuario;  
        this.nome = nome;  
        notas = new LinkedList<>();  
    }  
}
```

Agora que sei
POLIMORFISMO, entendo
que List é apenas uma
INTERFACE e conheço as
CLASSES CONCRETAS que
implementam a interface List
posso dominar o mundo!



Agora quero
usar Array!



Trabalhando

- **Exercício de Fixação**
 - 4 e 5





- **Varargs**

- <https://www.devmedia.com.br/listas-de-argumentos-de-comprimento-variavel-em-java/25559>
- <https://www.youtube.com/watch?v=vlthjvYNf08>

- **ArrayList**

- <https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html>
- <https://www.devmedia.com.br/visao-geral-da-interface-collection-em-java/25822>
- <https://www.devmedia.com.br/api-collections-em-java-fundamentos-e-implementacao-basica/28445>
- <https://www.devmedia.com.br/diferenca-entre-arraylist-vector-e-linkedlist-em-java/29162>
- <https://digaotutoriais.wordpress.com/2016/04/11/interface-set-list-map-e-queue/>