

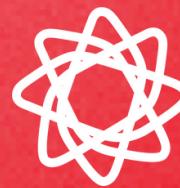


expouna

# Controle de Estoque em uma Loja de Informática

# Introdução

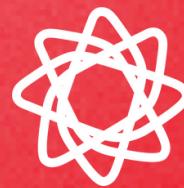
UC - Programação de Soluções Computacionais | PROFESSORES: Julio Vilela e Rafaela Priscila | INTEGRANTES: Amanda Cardoso, Felipe, Araujo, Henrique Oliveira, João Vitor e Pedro Henrique



## Introdução

O controle de estoque é fundamental em qualquer ramo, pois garante a disponibilidade de produtos, otimiza os custos operacionais e melhora a satisfação do cliente ao assegurar que os itens necessários estejam sempre à disposição no momento certo.

# Motivação

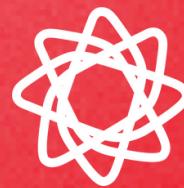


## Motivação

Visando melhorar a gestão de estoque de uma loja de Informática, elaboramos um programa para a realização de funções básicas de um sistema de controle , como:

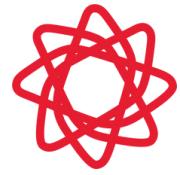
- Menu interativo para o usuário;
- Gerenciamento de Itens (Cadastro, inclusão e exclusão);
- Controle da quantidade de cada produto;
- Cálculo do valor total do estoque.

# Apresentação das Classes do Programa



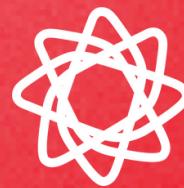
## Classe Main

Classe principal que contém o método main, responsável por exibir um menu interativo para o usuário. Permite adicionar novos itens ao estoque, listar os itens no estoque e sair do programa.



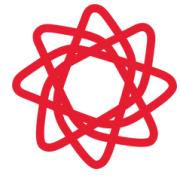
# Classe Main

```
Main.java X Celular.java X Item.java X PC.java X Consoles.java X Gerenciador.java X
Source History | I H G S M F D P C U L R E Z
7 |     Gerenciador gerenciador = new Gerenciador();
8 |     Scanner scanner = new Scanner(System.in);
9 |
10| -     while (true) {
11|       System.out.println("\nMenu:");
12|       System.out.println("1. Adicionar novo item ao estoque");
13|       System.out.println("2. Exibir lista de itens no estoque");
14|       System.out.println("3. Calcular valor total do estoque");
15|       System.out.println("4. Sair");
16|       System.out.print("Escolha uma opção: ");
17|
18|       int opcao = scanner.nextInt();
19|       scanner.nextLine(); // consumir a nova linha
20|
21|       // switch case para selecionar a opção do menu
22|       switch (opcao) {
23|         -     case 1 -> {
24|             System.out.println("Escolha a categoria do item:");
25|             System.out.println("1. Celular");
26|             System.out.println("2. PC");
27|             System.out.println("3. Consoles");
28|             int categoria = scanner.nextInt();
29|             scanner.nextLine(); // consumir a nova linha
30|         }
31|       }
32|     }
33|   }
34| }
```



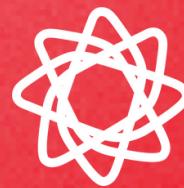
## Classe Item

Classe Item: Classe base abstrata para todos os itens.  
Contém atributos comuns como Nome, Quantidade, Preço, Descrição e Tamanho.



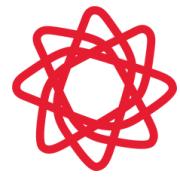
# Classe Item

```
6  private string name;
7  private int amount;
8  private double price;
9  String description;
10 String size;
11
12 public Item(String name, int amount, double price, String description, String size) {
13     this.name = name;
14     this.amount = amount;
15     this.price = price;
16     this.description = description;
17     this.size = size;
18 }
19
20     /*get e set
21 Set atribui um valor
22 Get retorna o valor
23 */
24 public String getName() {
25     return name;
26 }
27
28 public void setName(String name) {
29     this.name = name;
30 }
31
32 public int getAmount() {
33     return amount;
34 }
35
36 public void setAmount(int amount) {
37     this.amount = amount;
38 }
39
40 public double getPrice() {
41     return price;
42 }
```



## Sub-Classes Celular, PC e Consoles:

Estas são subclasses de Item e representam as categorias específicas dos itens. Cada classe tem atributos adicionais específicos para a categoria.

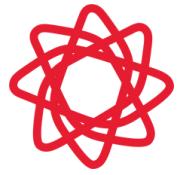


# Sub-Classes Celular, PC e Consoles:



## Celular:

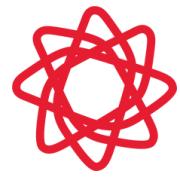
```
4  public class Celular extends Item{
5      private String marca;
6
7      public Celular(String name, String description, String size, int amount, double price, String marca) {
8          super (name, amount, price, description, size);
9          this.marca = marca;
10     }
11     /*get e set
12     Set atribui um valor
13     Get retorna o valor
14     */
15     public void setMarca (String marca){
16         this.marca = marca;
17     }
18     public String getMarca(){
19         return marca;
20     }
21     // vamos retornar os atributos
22     @Override
23     public String toString(){
24         return "Celular{" +
25             "nome='"+ getName() + '\'' +
26             ",quantidade=" + getAmount() +
27             ",preço=" + getPrice() +
28             ", marca='"+ marca + '\'' +
29             '}';
30     }
31
32 }
```



# Sub-Classes Celular, PC e Consoles:

## PCs:

```
2 // Sub classe PC (computador)
3 public class PC extends Item {
4     private String marca;
5
6     public PC(String name, int amount, double price, String marca, String description, String size) {
7         super(name, amount, price, description, size);
8         this.marca = marca;
9     }
10
11    /*get e set
12     Set atribui um valor
13     Get retorna o valor
14     */
15    public void setMarca(String marca) {
16        this.marca = marca;
17    }
18
19    public String getMarca() {
20        return marca;
21    }
22
23    // Retornando os atributos
24    @Override
25    public String toString() {
26        return "Computador{" +
27            "nome='" + getName() + '\'' +
28            ", quantidade=" + getAmount() +
29            ", preço=" + getPrice() +
30            ", descrição='" + getDescription() + '\'' +
31            ", tamanho='" + getSize() + '\'' +
32            ", marca='" + marca + '\'' +
33            '}';
34    }
35}
```

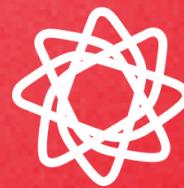


# Sub-Classes Celular, PC e Consoles:



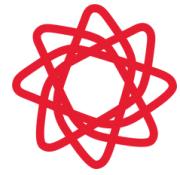
## Consoles:

```
4  public class Consoles extends Item {
5      private String marca;
6      private String jogos;
7
8      public Consoles(String name, String description, String size, int amount, String marca, String jogos, double price) {
9          super(name, amount, price, description, size);
10         this.marca = marca;
11         this.jogos = jogos;
12     }
13
14     // Getters e Setters
15     public void setMarca(String marca) {
16         this.marca = marca;
17     }
18
19     public String getMarca() {
20         return marca;
21     }
22
23     public void setJogos(String jogos) {
24         this.jogos = jogos;
25     }
26
27     public String getJogos() {
28         return jogos;
29     }
30
31     // Retornando os atributos
32     @Override
33     public String toString() {
34         return "Video Game{" +
35             "nome='" + getName() + '\'' +
36             ", quantidade=" + getAmount() +
37             ", preço=" + getPrice() +
38             ", marca='" + marca + '\'' +
39             ", jogos='" + jogos + '\'' +
40             '}';
41     }
```



## Classe Gerenciador

Classe responsável por gerenciar a lista de itens no estoque. Possui métodos para adicionar itens e listar todos os itens no estoque.



# Classe Gerenciador

```
3  [-] import java.util.ArrayList;
4  [-] import java.util.List;
5
6  public class Gerenciador {
7      private final List<Item> itens;
8
9  [-]     public Gerenciador() {
10         this.itens = new ArrayList<>();
11     }
12
13     // Método para adicionar um item ao estoque
14     [-] public void adicionarItem(Item item) {
15         itens.add(item);
16     }
17
18     // Método para listar todos os itens no estoque
19     [-] public void listarItens() {
20         [-]     for (Item item : itens) {
21             System.out.println(item);
22         }
23     }
24
25     // Método para calcular o valor total do estoque
26     [-] public double calcularValorTotalEstoque() {
27         double total = 0;
28         [-]     for (Item item : itens) {
29             total += item.getPrice() * item.getAmount();
30         }
31         return total;
32     }
33 }
```

# Execução do Programa

# Considerações Finais



expouna

**Agradecemos pela  
sua atenção!**