***­­Importante***: Para cada item abaixo deve ser copiado trechos do código que cumprem o requisito e explicado, se não for aparente, o porquê o requisito é cumprido. Sejam bem explícitos. Deve ser indicado também o arquivo da classe em que está o trecho do código. Eu avaliarei o código do Github a partir desse documento para confirmá-lo e também para detectar possíveis erros. **Quem não seguir o que está indicado aqui, não terá o projeto avaliado e perderá a atividade.**

**Usar um novo repositório!**

**Entrega final: 26 de novembro via GitHub. Prova oral sobre o código ainda a ser definida.**

**Entrega parcial: 19 de novembro via GitHub. Apenas Diagrama de Classes, interfaces e classes abstratas no branch general.**

**Requisitos Gerais**

***Todos os atributos e funções membros devem estar relacionados a classe***

1. Pelo menos 4 atributos
   1. Livros:
      1. private float preco;
      2. private String nome;
      3. private Genero tiposdelivros ;
      4. private int numerodepaginas;
      5. private int numerodeexemplares;
   2. Filmes:
      1. private float preco;
      2. private String nome;
      3. private Genero tiposdeflme;
      4. private float duracao;
   3. Usuario:
      1. private String nome;
      2. private String endereco;
      3. private String Loguin,senha;
   4. SistemaDeLogin:
      1. private ArrayList<Usuario> usuarios;
      2. private ArrayList<Amazon> amazon;
      3. private static int numerodecadastros;
   5. LojaOnline
      1. protected Tipodecompra formadepagamento;
      2. protected static int quantidadedelivros = 0;
      3. protected int livrosccomprados;
      4. protected ArrayList<Produto> livros;
   6. InstantVideo:
      1. protected Tipodenegocio negocio;
      2. protected static ArrayList<Filmes> filmes;
      3. protected ArrayList<Filmes> filmesalugados;
      4. protected ArrayList<Filmes> filmescomprados;
   7. Amazon:
      1. protected final static String CNPJ = "15.436.940/0001-03";
      2. protected Nota satisfacaodocliente;
   8. Pessoa:
      1. protected enum Entrada{USUARIO,ADMINISTRADOR};
      2. protected String nome;
      3. protected String cpf;
   9. ServocoDeVendas:
      1. protected String lembretesDeCompras;
   10. Obs: Nas classes Amazon e SistemaDeLogin não encontrei necessidade de adicionar mais variáveis.
2. Pelo menos 4 funções membros sem incluir get e set ok
   1. Amazon:
      1. public void menu();
      2. public void darNota();
   2. InstantVideo:
      1. private boolean verificar(Filmes filme);
      2. private void mostrar();
      3. private void add();
      4. public void alugar();
      5. public void comprar();
      6. public void comprar(int i);
      7. public void menu();
   3. LojaOnline:
      1. public void addLivro();
      2. public void atualizarLivro();
      3. public void mostrarLivros();
      4. private int codigo();
      5. public void Comprar();
      6. private int totalDeExemplares();
      7. public void verDetalhes();
      8. public void menu();
   4. SistemaDeLogin:
      1. public void fazerLogin();
      2. public static void cadastrar(SistemaDeLogin sistema ,Usuario usuario);
      3. public static void deletar(SistemaDeLogin sistema);
      4. public static void menu(SistemaDeLogin sistema);
   5. Usuario
      1. public void Cadastrar();
      2. public boolean equals(Usuario obj);
   6. Filmes:
      1. public final void detalhes();
   7. Livros:
      1. public boolean comprarlivro();
      2. public void atualizar();
      3. public void detalhes();
      4. public void Menu();
   8. Pessoa
      1. public abstract void menu();
   9. Produto:
      1. public abstract boolean comprar();
      2. public abstract void atualizar();
   10. Servico:
       1. public abstract void realizarServico();
   11. Site:
       1. public abstract void conectar();
   12. Obs: nas classes Usuario e Filmes não encontrei motivos de adicionar mais funções ao projeto.
3. Diagrama UML completo (obrigatório salvar também o png do diagrama no gitHub) ok

**Requisitos de implementação**

1. Todas as classes concretas devem vir de interfaces ou classes abstratas. Pelo menos três hierarquias de classes. Uma das hierarquias deve ter três níveis. Exemplo: Personagem >> Ciborgue >> Robocop; Class Arma (interface) >> Beretta93R
   1. Interface: Servico >> (classe abstrata) ServicoDeVendas >> (classe concreta) Amazon >> (Subclasse de Amazon) LojaOnline, (Subclasse de Amazon) InstatVideo, (Subclasse de Amazon) Webservice.
2. Ao menos três interfaces. A terceira interface deve ser uma interface que liga duas hierarquias como no exemplo da interface **corredor** (Figura 1).
   1. Interfaces :
      1. Produto.
      2. Servico.
      3. Site.
   2. A interface site liga a Hierarquia Servico >> (classe abstrata) ServicoDeVendas >> (classe concreta) Amazon >> (Subclasse de Amazon) LojaOnline na classe LojaOnline e Saraiva.
      1. public final class LojaOnline extends Amazon implements Site
      2. public class Saraiva extends ServicoDeVendas implements Site
3. Usar a interface **Comparable** e sobrescrever o método **compareTo** em pelo menos uma hierarquia
   1. public abstract class Pessoa implements Comparable<Pessoa>
      1. public abstract int compareTo(Pessoa o);
4. Sobrescrever **equals** para de Object
5. Todas as hierarquias devem ter classes Concretas, e em uma das hierarquias, três classes Concretas relacionadas: Exemplo Servico >> ServicoStream >> (Netflix, HBOStream, AmazonPrime, NowTv). Em uma pesquisa de 10 segundos: <http://www.tomsguide.com/us/pictures-story/620-top-online-streaming-video.html>
   1. Amazon >> ( LojaOnline, InstatVideo, Webservice ).
      1. public class InstantVideo extends Amazon implements Site
      2. public final class LojaOnline extends Amazon implements Site
      3. public class Webservices extends Amazon
6. Sempre usar o super para o máximo de reaproveitamento de código
   1. LojaOnline:
      1. public String toString() {
      2. String out, out1, out2, out3, out4, out5;
      3. out = super.toString();
      4. ...
      5. public LojaOnline() {
      6. super();
      7. ...
   2. Webservice
      1. public String toString() {
      2. return super.toString()+
      3. ...
      4. public Webservices(String descreicao, float espacodearmazenamento, float valor, Nota satisfacaodocliente) {
      5. super(satisfacaodocliente);
      6. ...
   3. InstanteVideo
      1. public String toString() {
      2. return super.toString()+
      3. ...
      4. public InstantVideo() {
      5. super();
      6. ...
7. Atributos static e const static
   1. Amazon:
      1. protected final static String CNPJ = "15.436.940/0001-03";
   2. InstantVideo
      1. protected static ArrayList<Filmes> filmes;
   3. LojaOnline:
      1. protected static int quantidadedelivros = 0;
   4. SistemaDeLogin:
      1. private static int numerodecadastros=0;
   5. Saraiva:
      1. private static ArrayList<Produto> produtos;
      2. private static int quantidadedeprodutos;
      3. private static final String CNPJ = "xxxxxxx-xxxxxxx-xxxxxx-xx";
8. Método static
   1. Classe Principal:
      1. public static void menuAmazon(Amazon a)
      2. public static void menuIntantVideo(InstantVideo a)
      3. public static void menulojaonline(LojaOnline a)
      4. public static void menuWebservices(Webservices a)
      5. public static void menuSaraiva(Saraiva a)
   2. Casse SistemaDeLogin:
      1. public static void cadastrar(SistemaDeLogin sistema, Pessoa usuario)
9. Construtores em todas as classes, e dois na hierarquia principal. Sempre validar os dados em todas as classes
   1. Amazon:
      1. public void setSatisfacaodocliente() {
      2. Scanner in = new Scanner(System.in);
      3. int escolha;
      4. do{
      5. System.out.println("digite sua satisfação com nossos sistemas");
      6. System.out.println("\n1- Ruim\n2- Bom\n3- Exelente");
      7. escolha=in.nextInt();
      8. switch(escolha){
      9. case 1:
      10. this.satisfacaodocliente = Nota.Ruim;
      11. break;
      12. case 2:
      13. this.satisfacaodocliente = Nota.Bom;
      14. break;
      15. case 3:
      16. this.satisfacaodocliente = Nota.Exelente;
      17. break;
      18. default:
      19. System.out.println("digite um valor valido!!");
      20. }
      21. }while(escolha < 1 && escolha > 3);
      22. }
      23. A entrada é feita apenas pelo setSatisfacaodocliente, pois a outra variável (CNPJ) é static final;
   2. InstantVideo:
      1. public InstantVideo(ArrayList<Filmes> filmes, ArrayList<Filmes> filmesalugados) {
      2. super();
      3. setNegocio();
      4. InstantVideo.filmes = filmes;
      5. this.filmesalugados = filmesalugados;
      6. }
      7. Como Filmes e filmes alugados são apenas arrays a validação é feita pela classe array e pelos respectivos tipos.
      8. public final void setNegocio() {
      9. Scanner in = new Scanner(System.in);
      10. int escolha;
      11. do{
      12. System.out.println("digite o tipo de compra");
      13. System.out.println("\n1- Alugar\n2- Comprar");
      14. escolha=in.nextInt();
      15. switch(escolha){
      16. case 1:
      17. this.negocio = Tipodenegocio.ALUGAR;
      18. break;
      19. case 2:
      20. this.negocio = Tipodenegocio.COMPRAR;
      21. break;
      22. default:
      23. System.out.println("digite um valor valido!!");
      24. }
      25. }while(escolha < 1 && escolha > 2);
      26. }
      27. No setNegocio e dado na verdade um menu de escolha já que o dado é do tipo enum.
   3. LojaOnline:
      1. public LojaOnline(int tamanhoinicial) {
      2. super();
      3. setFormadepagamento();
      4. setLivros(tamanhoinicial);
      5. setLivrosccomprados(0);
      6. }
      7. Toda a validação foi feita no setLivros().
      8. public void setLivros(int tamanhoinicial) {
      9. if (tamanhoinicial > 0)
      10. this.livros = new ArrayList<>(tamanhoinicial);
      11. else
      12. this.livros = new ArrayList<>(tamanhoinicial);
      13. }
      14. Verificar se o tamanho o vetor é menor ou igual a 0;
   4. SistemaDeLogin:
      1. Como se trata apenas de um sistema para entrada de dados e possui apenas vetores não é necessário a validação de dados.
   5. Usuario:
      1. public Usuario(String nome, String Login,String senha,String endereco) {
      2. setNome(nome);
      3. setLoguin(Login);
      4. setSenha(senha);
      5. this.endereco = endereco;
      6. }
      7. Como a classe pode receber todo o tipo de entrada já que é String fica para o meu nível de conhecimento difícil de validar.
   6. Livros:
      1. public Livros(float preco, String nome, int numerodepaginas,int numerodeexemplares) {
      2. setPreco(preco);
      3. setNome(nome);
      4. setTiposdelivros();
      5. setNumerodepaginas(numerodepaginas);
      6. setNumerodeexemplares(numerodeexemplares);
      7. }
      8. Validação feita nos sets.
         1. public final void setPreco(float preco) {
         2. if (preco < 0){
         3. this.preco = 00;
         4. }else
         5. this.preco = preco;
         6. }
         7. public final void setTiposdelivros() {
         8. Scanner in =new Scanner(System.in);
         9. int escolha;
         10. System.out.println("ESCOLHA O GENERO DE FILME: ");
         11. do{
         12. System.out.println("1- ROMANCE\n2- AVENTURA\n3FICCAO\_CIENTIFICA");
         13. escolha = in.nextInt();
         14. switch(escolha){
         15. case 1:
         16. this.tiposdelivros=Genero.ROMANCE;
         17. break;
         18. case 2:
         19. this.tiposdelivros=Genero.AVENTURA;
         20. break;
         21. case 3:
         22. this.tiposdelivros=Genero.FICCAO\_CIENTIFICA;
         23. break;
         24. case 4:
         25. this.tiposdelivros=Genero.OUTROS;
         26. break;
         27. default :
         28. System.out.println("Digite um valor valido");
         29. }
         30. }while(escolha < 4 || escolha < 1);
         31. }
         32. public final void setNumerodepaginas(int numerodepaginas) {
         33. if (numerodepaginas < 0){
         34. this.numerodepaginas=0;
         35. }else
         36. this.numerodepaginas = numerodepaginas;
         37. }
         38. public final void setNumerodeexemplares(int numerodeexemplares) {
         39. if (numerodeexemplares < 0){
         40. this.numerodeexemplares=0;
         41. }else
         42. this.numerodeexemplares = numerodeexemplares;
         43. }
   7. Filmes:
      * 1. public Filmes(float preco, String nome, int duracao) {
        2. setPreco(preco);
        3. setNome(nome);
        4. setTiposdeflme();
        5. setDuracao(duracao);
        6. }
        7. public final void setPreco(float preco) {
        8. if (preco < 0){
        9. this.preco=0;
        10. }else
        11. this.preco = preco;
        12. }
        13. public final void setTiposdeflme() {
        14. Scanner in =new Scanner(System.in);
        15. int escolha;
        16. System.out.println("ESCOLHA O GENERO DE FILME: ");
        17. do{
        18. System.out.println("1- ROMANCE\n2- AVENTURA\n3FICCAO\_CIENTIFICA");
        19. escolha = in.nextInt();
        20. switch(escolha){
        21. case 1:
        22. this.tiposdeflme=Genero.ROMANCE;
        23. break;
        24. case 2:
        25. this.tiposdeflme=Genero.AVENTURA;
        26. break;
        27. case 3:
        28. this.tiposdeflme=Genero.FICCAO\_CIENTIFICA;
        29. break;
        30. case 4:
        31. this.tiposdeflme=Genero.OUTROS;
        32. break;
        33. default :
        34. System.out.println("Digite um valor valido");
        35. }
        36. }while(escolha < 4 || escolha < 1);
        37. }
        38. public final void setDuracao(float duracao) {
        39. if (duracao > 0)
        40. this.duracao = duracao;
        41. else
        42. this.duracao=1;
        43. }
   8. 2 Cosntrutores
      1. Amazon:
         * 1. public Amazon(){
           2. this.satisfacaodocliente = Nota.Indefinida;
           3. }
           4. public Amazon(Nota satisfacaodocliente) {
           5. this.satisfacaodocliente=satisfacaodocliente;
           6. }
      2. InstantVideo:
         * 1. public InstantVideo(ArrayList<Filmes> filmes, ArrayList<Filmes> filmesalugados) {
           2. super();
           3. setNegocio();
           4. InstantVideo.filmes = filmes;
           5. this.filmesalugados = filmesalugados;
           6. }
           7. public InstantVideo() {
           8. super();
           9. this.negocio= Tipodenegocio.INDEFINIDOO;
           10. InstantVideo.filmes = new ArrayList<>();
           11. this.filmesalugados= new ArrayList<>(3);
           12. this.filmescomprados = new ArrayList<>();
           13. }
      3. LojaOnline:
         * 1. public LojaOnline(int tamanhoinicial) {
           2. super();
           3. setFormadepagamento();
           4. setLivros(tamanhoinicial);
           5. setLivrosccomprados(0);
           6. }
           7. public LojaOnline(){
           8. super();
           9. this.formadepagamento=Tipodecompra.CARTAO\_DE\_CREDITO;
           10. setLivros(1);
           11. setLivrosccomprados(0);
           12. }
      4. Webservice
         * 1. public Webservices(String descreicao, float espacodearmazenamento, float valor) {
           2. this.descricao = descreicao;
           3. this.espacodearmazenamento = espacodearmazenamento;
           4. this.valor = valor;
           5. }
           6. public Webservices(String descreicao, float espacodearmazenamento, float valor, Nota satisfacaodocliente) {
           7. super(satisfacaodocliente);
           8. this.descricao = descreicao;
           9. this.espacodearmazenamento = espacodearmazenamento;
           10. this.valor = valor;
           11. }
10. Construtor cópia em uma das hierarquias
    1. Saraiva:
       1. public Saraiva(Saraiva s){
       2. this.lembretesDeCompras=s.lembretesDeCompras;
       3. this.numerodecompras=s.numerodecompras;
       4. this.premio=s.premio;
       5. }
11. ArrayList
    1. InstantVideo:
       1. protected static ArrayList<Produto> filmes;
       2. protected ArrayList<Produto> filmesalugados;
       3. protected ArrayList<Produto> filmescomprados;
    2. LojaOnline:
       1. protected ArrayList<Produto> livros;
    3. SistemaDeLogin:
       1. private ArrayList<Produto> usuarios;
       2. private ArrayList<ServicoDeVendas> amazon;
    4. Saraiva:
       1. private static ArrayList<Produto> produtos;
12. ENUM
13. Amazon:

protected enum Nota { Ruim, Bom, Exelente, Indefinida};

1. InstantVideo:

protected enum Tipodenegocio {ALUGAR , COMPRAR, INDEFINIDOO};

1. LojaOnline:

public enum Tipodecompra { BOLETO\_BAMCARIO, CARTAO\_DE\_CREDITO };

1. Usar o **instanceof** no main junto com as classes concretas. Para uma da classe concreta identificada, chamar um método dessa classe e fazer uma ação;
   * 1. for (int i = 0; i < 2; i++) {
     2. if (p[i] instanceof Usuario){
     3. System.out.println(" este é um usuario comum ");
     4. SistemaDeLogin.cadastrar(sistema, p[i]);
     5. }else{
     6. System.out.println("este é um administrador ");
     7. SistemaDeLogin.cadastrar(sistema, p[i]);
     8. }
     9. }
     10. for (int i = 0; i < 2; i++) {
     11. if (site[i] instanceof Saraiva ){
     12. System.out.println("bem vindo a Saraiva");
     13. }else{
     14. System.out.println("bem vindo a loja online da amazon ");
     15. }
     16. site[i].conectar();
     17. }
     18. for (int i = 0; i < 5; i++) {
     19. if (serv[i] instanceof Saraiva)
     20. menuSaraiva(((Saraiva) serv[i]));
     21. else if (serv[i] instanceof Amazon)
     22. menuAmazon((Amazon) serv[i]);
     23. else if (serv[i] instanceof LojaOnline)
     24. menulojaonline((LojaOnline) serv[i]);
     25. else if (serv[i] instanceof InstantVideo)
     26. menuIntantVideo((InstantVideo) serv[i]);
     27. else
     28. menuWebservices((Webservices) serv[i]);
     29. }
2. Dividir o projeto em pacotes
3. Trabalhar com pacotes
   1. Pacote Amazon que possui as classes Amazon, IntantVideo e LojaOnline.
   2. Pacote Usuario que possui as classes Usuarios, Administrador e SistemaDeLogin.
   3. Pacote Produtos que possui as classes Filmes e Livros.
   4. Pacote Saraiva que possui a classe Saraiva.
   5. Pacote abstratas que possui as classes Pessoa e ServicoDeVendas.
   6. Pacote intercaces que possui as classes Produto,Servico,Site.
   7. Pacote Principal que possui a classe principal (única que possui o método main).
4. Sobrescrever para todas as classes o método toString
   1. Amazon:
      1. @Override
      2. public String toString() {
      3. String out1, out2, out3;
      4. out1 = ("##############AMAZOM.COM##############");
      5. out2 = ("com o CNPJ: " + CNPJ);
      6. out3 = ("E com a nota definida por você: " + this.satisfacaodocliente.name());
      7. return out1 + "\n" + out2 + "\n" + out3 + "\n";
      8. }
   2. InstantVideo:
      1. @Override
      2. public String toString() {
      3. return super.toString()+"InstantVideo{" + "negocio=" + negocio + ", filmesalugados=" + filmesalugados + ", filmescomprados=" + filmescomprados + '}';
      4. }
   3. LojaOnline:
      1. @Override
      2. public String toString() {
      3. String out, out1, out2, out3, out4, out5;
      4. out = super.toString();
      5. out1 = ("-------Loja Online-------");
      6. out2 = ("voce escolheu pagar com: " + this.formadepagamento.name() + ".");
      7. out3 = ("voce ja comprou: " + this.livrosccomprados + " livros.");
      8. out4 = ("a loja possui: " + quantidadedelivros + " Livros diferemtes.");
      9. out5 = ("E um total de: " + totalDeExemplares());
      10. return out + "\n" + out1 + "\n" + out2 + "\n" + out3 + "\n" + out4 + "\n" + out5 + "\n";
      11. }
   4. Webservice:
      1. @Override
      2. public String toString() {
      3. return super.toString()+"\nWebservices{" + "descreicao=" + descricao + ", espacodearmazenamento=" + espacodearmazenamento + ", valor=" + valor + '}';
      4. }
   5. Usuario:
      1. @Override
      2. public String toString() {
      3. return"Loguin=" + Login + ", senha=" + senha;
      4. }
   6. Administrador
      1. @Override
      2. public String toString() {
      3. return "cargo=" + cargo + ", login=" + login + ", senha=" + senha;
      4. }
   7. Livros:
      1. public String toString(){
      2. String out,out1,out2,out3,out4;
      3. out = "################## "+this.nome+" ##################";
      4. out1 = "com: "+this.numerodepaginas+" paginas.";
      5. out2 = "do genero: "+this.tiposdelivros.name()+".";
      6. out3 = "com: "+this.numerodeexemplares+" exemplares no estoque";
      7. out4 = "No valor de "+this.preco+" R$";
      8. return out +"\n"+ out1+"\n"+out2+"\n"+out3+"\n"+out4;
      9. }
   8. Filmes:
      1. @Override
      2. public String toString(){
      3. String out,out1,out2,out3;
      4. out = ("################## "+this.nome+" ##################");
      5. out1 = ("com: "+this.duracao+" minutos de duração.");
      6. out2 = ("do genero: "+this.tiposdeflme.name()+".");
      7. out3 = ("com: "+this.preco+" RS");
      8. return out +"\n"+ out1+"\n"+out2+"\n"+out3;
      9. }
   9. Saraiva
      1. @Override
      2. public String toString() {
      3. return "Saraiva{" + "premio=" + premio + ", numerodecompras=" + numerodecompras + '}';
      4. }
5. Usar a classe Random do pacote java.util (java.util.Random). Nota: deve ser usado conforme o contexto do projeto. Se for usado em um método genérico sem relação com a classe e apenas para cumpri-lo, esse requisito será desconsiderado.
   1. Classe Saraiva:
      1. public void sorteio() {
      2. Random rand = new Random();
      3. switch (rand.nextInt(5)) {
      4. case 0:
      5. this.premio = PREMIOS.CARRO;
      6. break;
      7. case 1:
      8. this.premio = PREMIOS.DESCONTO;
      9. break;
      10. case 2:
      11. this.premio = PREMIOS.LIVRO;
      12. break;
      13. case 3:
      14. this.premio = PREMIOS.VIAGEM;
      15. break;
      16. case 4:
      17. this.premio = PREMIOS.NAO\_FOI\_DESTA\_VEZ;
      18. break;
      19. }
      20. System.out.println("parabens seu premio é :" + this.premio.name());
      21. }
6. No main o usuário deve fazer entrada via teclado e interagir com a aplicação. Opcional de bônus: pode ser usada a classe JOptionPane do pacote javax.swing. Vejam: showInputDialog e showMessageDialog.
   1. JOptionPane telinha = new JOptionPane();
   3. JOptionPane.showMessageDialog(telinha,"Tipos de usuario: ");
   4. for (int i = 0; i < 2; i++) {
   5. if (p[i] instanceof Usuario){
   6. System.out.println(" este é um usuario comum ");
   7. SistemaDeLogin.cadastrar(sistema, p[i]);
   8. }else{
   9. System.out.println("este é um administrador ");
   10. SistemaDeLogin.cadastrar(sistema, p[i]);
   11. }
   12. }
   14. JOptionPane.showMessageDialog(telinha,"sites para acessar: ");
   16. for (int i = 0; i < 2; i++) {...}
   18. JOptionPane.showMessageDialog(telinha,"Tipos de Serviços: ");
   20. for (int i = 0; i < 5; i++) {...}



Figura 1 – Interface Corredor conectando duas hierarquias de classe