***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.**

**Criar repositórios desde o início do desenvolvimento. Primeiro commit deve estar com todas as classes declaradas sem implementação.**

**Primeira versão do código completa dia 01 de dezembro.**

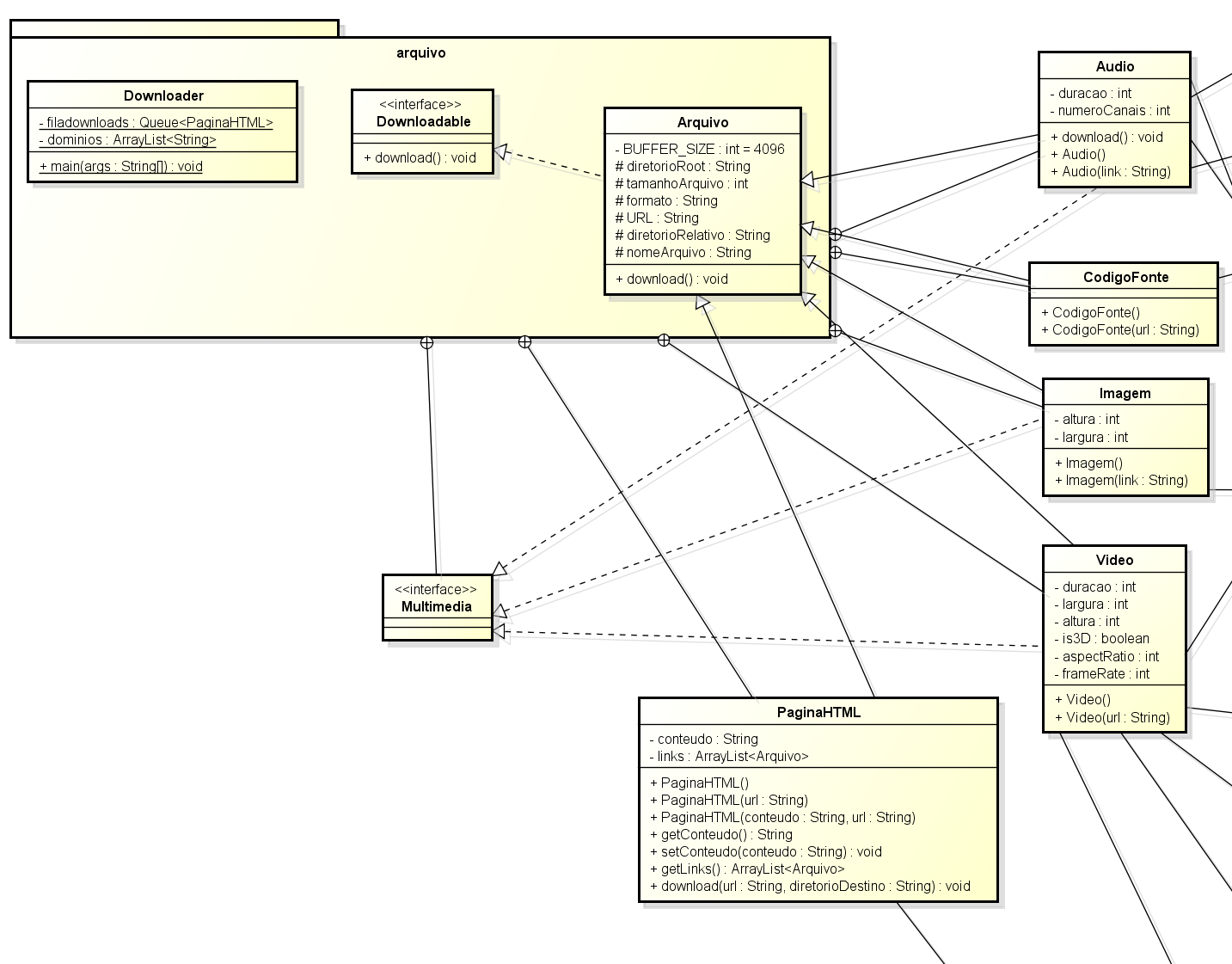
**Versão final do código dia 08 de dezembro.**

**No dia da apresentação deve ser mostrado o código rodando para vários casos de teste.**

**Será avaliado o quanto o projeto está elaborado. Trabalhos muito simples com métodos apenas mostrando mensagens, serão penalizados.**

**Requisitos de implementação para Java**

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

Não consegui fazer 3 hierarquias, apenas uma.

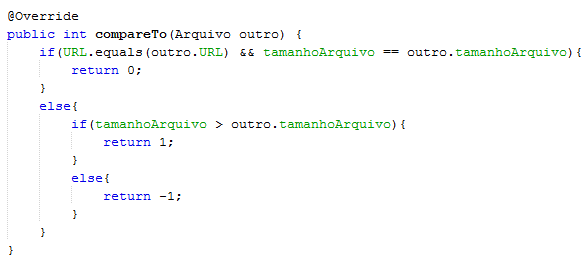
1. Ao menos três interfaces. A terceira interface deve ser uma interface que liga duas hierarquias como no exemplo da interface **corredor** (Figura 1).

Implementei apenas duas:



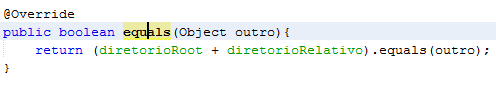
1. Usar a interface **Comparable** e sobrescrever o método **compareTo** em pelo menos uma hierarquia

(Arquivo)

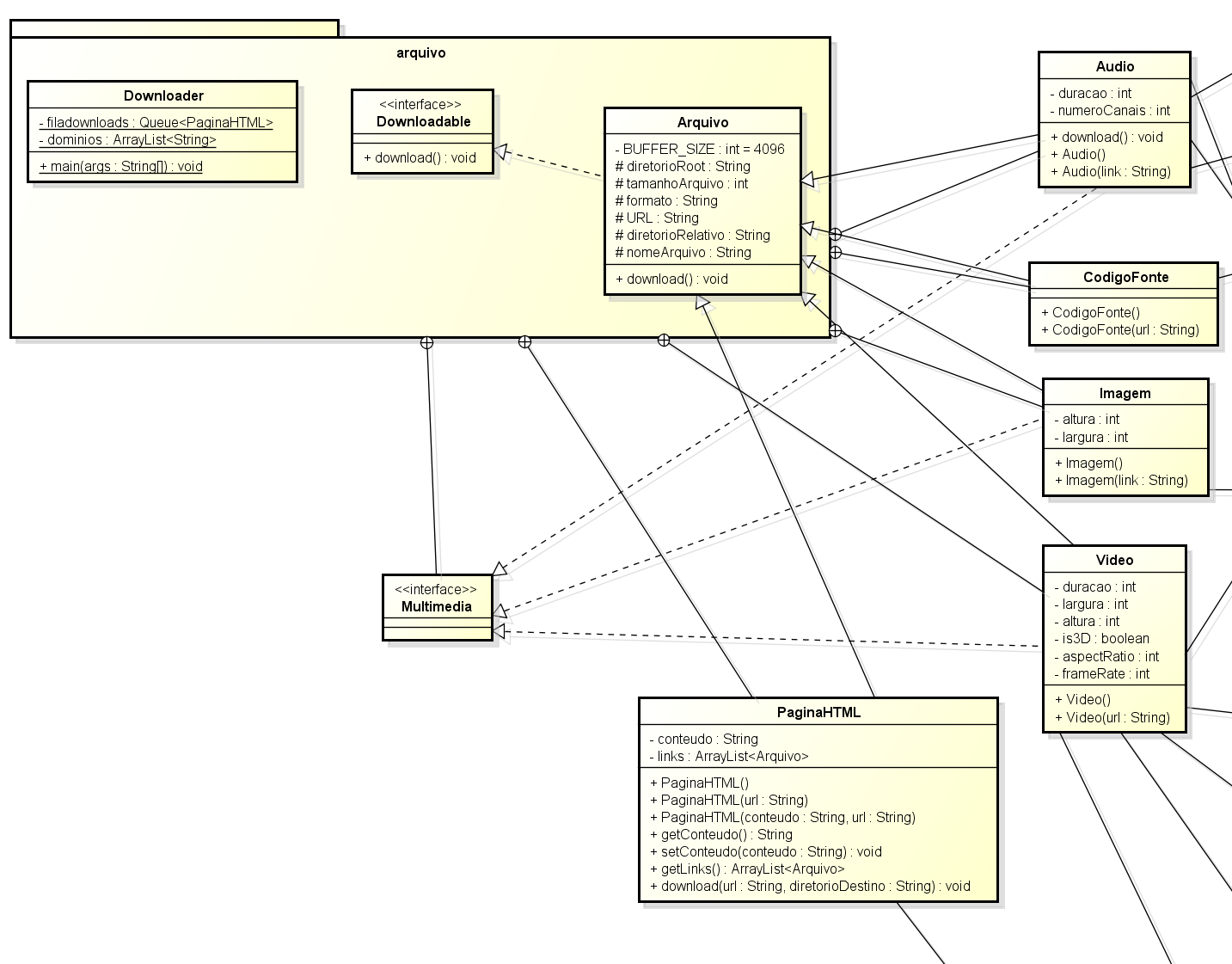


1. Sobrescrever **equals** para de Object

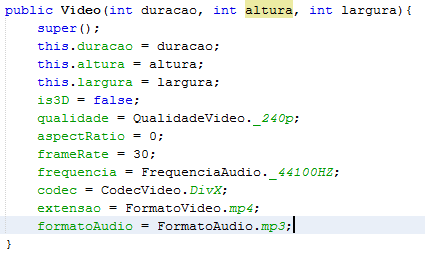
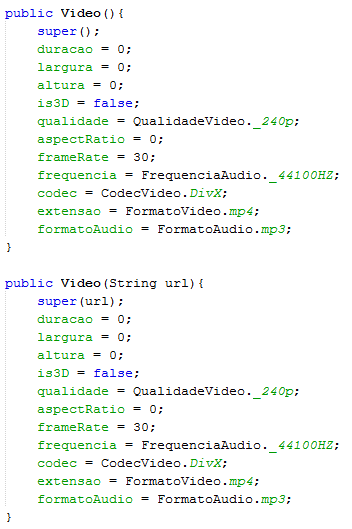
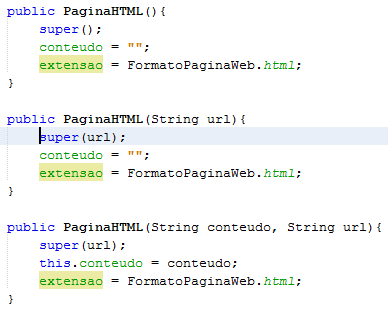
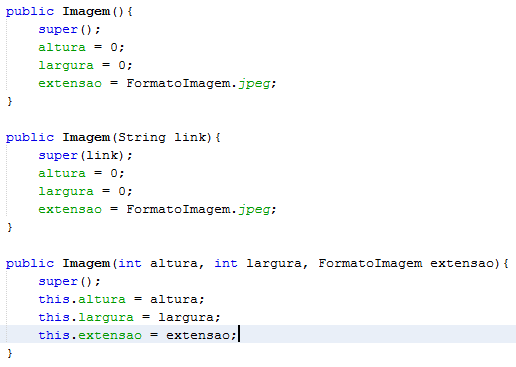
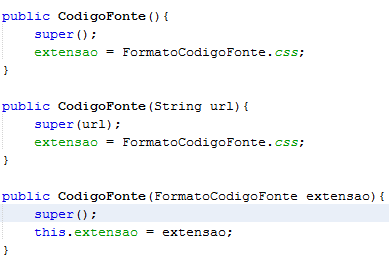
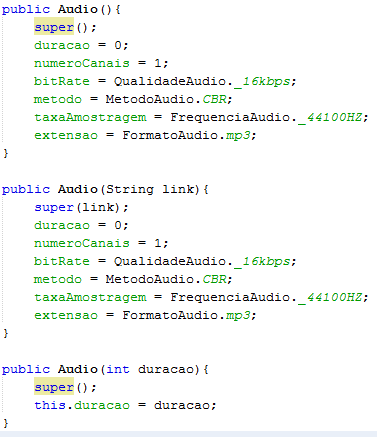
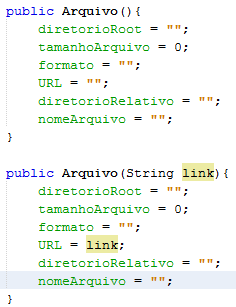
(Arquivo)



1. 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>

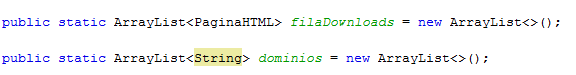


1. Sempre usar o super para o máximo de reaproveitamento de código em todas as classes



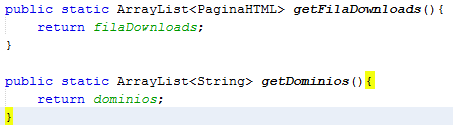
1. Atributos static e const static em todas as hierarquias de classe

(Downloader, Arquivo)

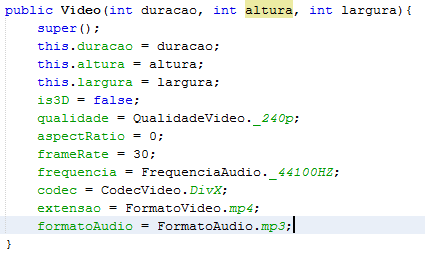
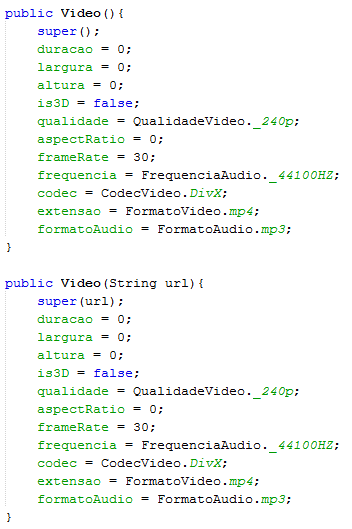
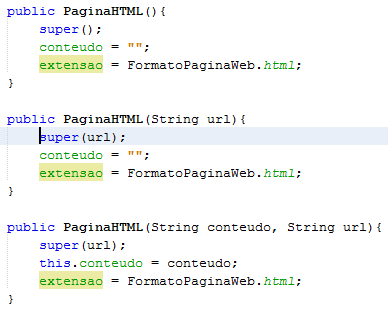
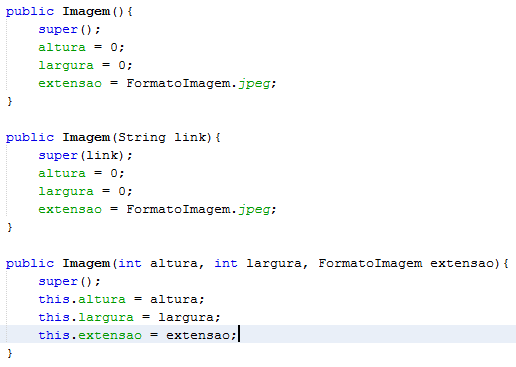
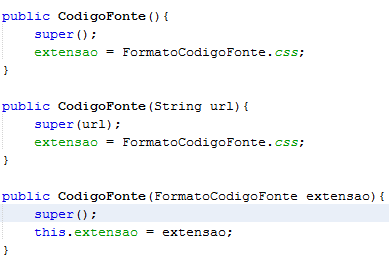
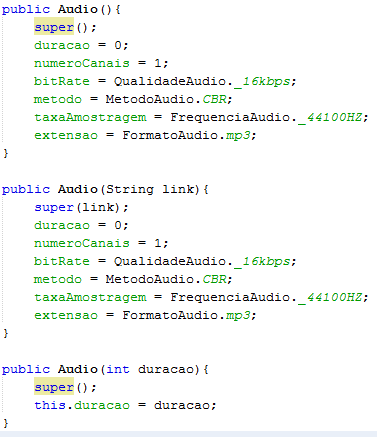
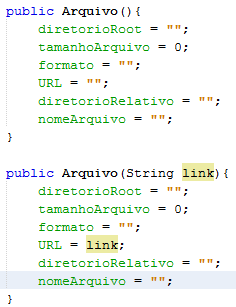


1. Método static em todas as hierarquias de classe

(Downloader.java)



1. Três construtores para todas as classes, sendo um deles o construtor de cópia. Sempre validar os dados em todas as classes

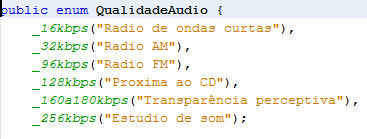
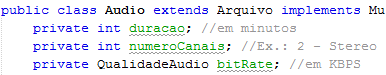


1. ArrayList na hierarquia principal



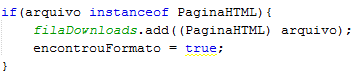
1. ENUM na hierarquia principal

(Audio, CodigoFonte, Imagem, PaginaHTML, Video)

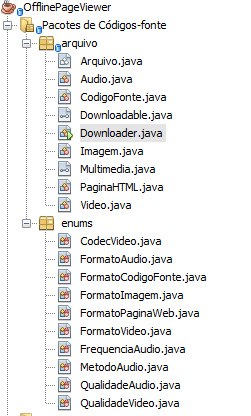


1. Usar o **instanceof** no main junto com as classes concretas. Para uma da classe concreta identificada, chamar um método dessa classe e fazer uma ação;

(Downloader.java)

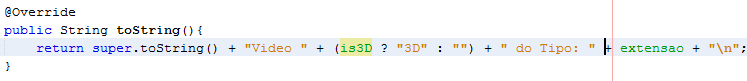
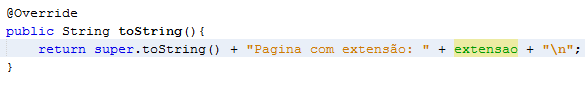
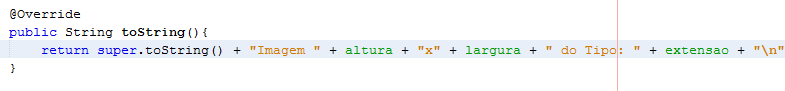
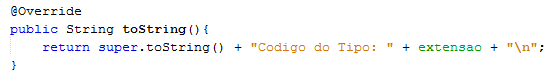
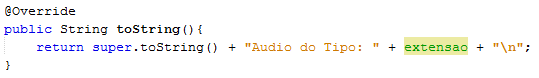
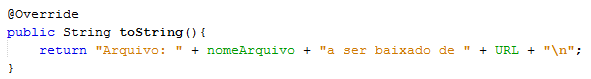


1. Dividir o projeto em pacotes



1. Sobrescrever para todas as classes o método toString

(Arquivo, Audio, CodigoFonte, Imagem, PaginaHTML, Video)



1. 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.

Não consegui pensar em um uso para este requisito.

1. 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.

(Downloader.java)





Figura 1 – Interface Corredor conectando duas hierarquias de classe