

UNIVERSIDADE PAULISTA
CAMPUS TATUAPÉ

ATIVIDADES PRÁTICAS SUPERVISIONADAS

Kaio Ruan Felix Martins N608480

Marcelo Nicolas Duzzi Olivares N529FJ9

Matheus Santos Brito de Almeida F215JG3

Nicolas Paiuca Buscarini N613HB9

“CONSUMO CONSCIENTE, LIXO E RECICLAGEM”

SÃO PAULO
2021/ 3º semestre

SUMÁRIO

1	Objetivo Do trabalho	5
2	Introdução.....	6
3	Elementos de programação Orientado a Objeto	9
3.1	Classe	9
3.2	Package	9
3.3	Atributos.....	10
3.4	Classe Abstrata.....	10
3.5	Métodos	11
3.6	Construtor	11
3.7	Sobrecarga.....	12
3.8	Método Abstrato	12
3.9	Interface.....	12
3.10	Objeto	13
3.11	Instância	14
3.12	Encapsulamento	14
3.13	Modificadores de Acesso	14
3.13.1	Public.....	14
3.13.2	Private	15
3.13.3	Protected.....	15
3.13.4	Default	15
3.14	Herança	15
3.15	Herança Múltipla	16
3.16	Polimorfismo.....	16
3.17	Sobrescrita.....	16
3.18	Exceções	17

4	Dissertação.....	18
5	Projeto (estrutura) do programa	25
6	Relatorio com as linhas de código do programa	38
6.1	Package Control	38
6.1.1	Classe: 'Main'	38
6.2	Package Model.....	41
6.2.1	Interface: 'InterLixos'	41
6.2.2	Classe Abstrata: 'Lixo'	41
6.2.3	Classe: Metal.....	42
6.2.4	Classe: 'Organico'	43
6.2.5	Classe: 'Papel'	44
6.2.6	Classe: 'Plastico'	45
6.2.7	Classe: 'Reciclagem'	47
6.2.8	Classe: 'Usuario'	52
6.2.9	Classe: 'Vidro'.....	54
7	Bibliografia	56
7.1	Material usado para a pesquisa:	56
7.2	Material usado para a programação:	56
8	Ficha de atividades	57

ÍNDICE DE FIGURAS

Figura 1 - Packages	25
Figura 2 – InterLixos.java.....	25
Figura 3 – Lixo.java.....	26
Figura 4 – Metal.java.....	27
Figura 5 - Organico.java	28
Figura 6 - Papel.java	29
Figura 7 – Plástico.java	30
Figura 8 – Reciclagem.java	31
Figura 9 - Ex.: Menu 2.....	31
Figura 10 - Ex.: Menu 1	31
Figura 11 – Ex.: Adicionando material a reciclagem	32
Figura 12 - Ex.: Adicionando a quantidade do material	32
Figura 13 - Ex.: Mostrar Reciclagem	32
Figura 14 - Usuario.java	33
Figura 15 - Métodos Usuario.....	33
Figura 16 – Ex.: PrintCalculos	34
Figura 17 - Vidro.java.....	35
Figura 18 - Instanciar Usuario.....	36
Figura 19 - Main.java.....	36
Figura 20 - While e Switch Case	36
Figura 21 - Switch Case: default.....	37
Figura 22 - Mensagem de erro	37
Figura 23 - Finalização do programa.....	37

1 OBJETIVO DO TRABALHO

O trabalho tem como objetivo em grupo concluir e apresentar o mesmo de forma de forma correta e eficiente com que consiga atingir todos os requisitos impostos previamente para o trabalho listados a seguir.

Desenvolver, pesquisar e aprender técnicas e conceitos sobre a importância da reciclagem e a partir desses conhecimentos , devemos utilizar e escolher uma técnica para criar um programa em que possa diminuir o lixo produzido pela sociedade com o objetivo da reutilização e com o objetivo de calcular o lixo reciclável e fazer a separação de cada material .

Com a Consciência ecológica, a estimulação da cultura da reciclagem e a possibilidade de promover o desenvolvimento sustentável do meio ambiente, por meio de ações direcionadas.

Discutir e entender qual a abordagem utilizada e os benefícios que ela trouxe em relação a outras técnicas. Elaborar uma dissertação para explicar todo o processo de criação e de funcionabilidade e expor eventuais vulnerabilidades e falhas detectadas pelo grupo

2 INTRODUÇÃO

Reciclagem é um termo genericamente utilizado para separar e reutilizar materiais beneficiados como matéria-prima para um novo produto. Há uma variedade enorme de matérias que podem ser reciclados e os exemplos mais comuns são o papel, o vidro, o metal e o plástico. Os melhores pontos positivos da reciclagem são a minimização da utilização de fontes naturais, muitas vezes não renováveis; e a minimização da quantidade de resíduos que precisa de tratamento final, como aterramento, ou incineração.

A palavra *reciclagem* difundiu-se na mídia a partir do final da década de 1980, quando foi constatado que as fontes de petróleo e de outras matérias-primas não renováveis estavam se esgotando rapidamente, e que havia falta de espaço para a disposição de lixo e de outros dejetos na natureza. A expressão vem do inglês *recycle* (*re* = repetir, e *cycle* = ciclo).

O conceito de reciclagem serve apenas para os materiais que podem voltar ao estado original e ser transformado novamente em um produto igual em todas as suas características. O conceito de reciclagem é diferente do de reutilização.

Apesar de não ser a única medida a ser realizada para a diminuição do lixo produzido pela sociedade, a reciclagem possui um importante papel, uma vez que, além de reduzir a quantidade de rejeitos, também diminui a procura por novas matérias-primas. Dessa forma, quanto mais se recicla, mais se reaproveita e, conseqüentemente, menor é a necessidade de extrair novos materiais da natureza, além de quanto mais reciclar, mais diminuirá os custos com limpeza urbana, além de evitar a poluição reduzindo as emissões de gases de efeito estufa que provocam a mudança climática global, mantendo o Meio Ambiente sustentável para as gerações futuras.

Além de favorecer uma atividade rentável gerando novos empregos, a reciclagem reduz a quantidade de resíduos (lixo não reciclável) enviados para aterros sanitários ou depósitos de lixo, prolongando a vida útil desses locais.

Há alguns casos em que a reciclagem reduz o consumo de energia. O exemplo mais clássico nesse sentido é o alumínio, um material quase que totalmente reciclável, pois a sua produção a partir da bauxita (recurso mineral

não renovável extraído do solo) demanda o consumo de uma grande quantidade de energia elétrica em uma indústria de base. Dessa forma, em alguns casos, é mais vantajoso economicamente o reaproveitamento das latas.

Ela faz parte dos três "R's" ou "erres": A sustentabilidade é a integração da utilização dos recursos de forma consciente para permitir que as futuras gerações possam também se desenvolver. E os 3R's da sustentabilidade são ações simples que você pode começar a adotar hoje mesmo.

Reduza: é a primeira ação para fomentar um cenário de reciclagem eficaz, já que será muito difícil conseguir bons índices de reciclagem se a quantidade de material descartado continuar a crescer. Então repense todos os resíduos que possam sair de cena ou ser substituído por reutilizável.

Reutilize: Busque reutilizar ao máximo. Baldes de insumos não contaminantes, potes de vidros para armazenar alimentos, se o seu negócio for de alimentos aproveite para utilizar todas as partes dos ingredientes utilizados.

Recicle: o que não der para reutilizar ou reduzir, direcione para reciclagem. Se eles não forem recicláveis volte para o primeiro item e reduza ou substitua por materiais, pelo menos, recicláveis.

O primeiro passo para a realização do processo de reciclagem é a coleta seletiva, ou seja, a separação do lixo por material, com o seu posterior destino para o reaproveitamento. Geralmente, divide-se primeiramente o material reciclável do não reciclável e, em seguida, separa-se o que é reciclável em metais, plástico, papel e vidro.

Embora a reciclagem, como vimos, seja muito importante, ela apresenta algumas limitações. A primeira delas é a de que, mesmo que exista.

O papel chamado de reciclado não é nada parecido com aquele que foi beneficiado pela primeira vez. Este novo papel tem cor diferente, textura diferente e gramatura diferente. Isto acontece devido a não possibilidade de retornar o material utilizado ao seu estado original e sim transformá-lo em uma massa que ao final do processo resulta em um novo material de características diferentes. A reciclagem de papel no Brasil ultrapassa 60% de tudo que é produzido. É um número ótimo comparado ao vidro e plástico, por exemplo. O processo econômico para a reciclagem, o baixo risco da coleta e triagem, aliados

a um material final de boa qualidade, justificam o bom índice de reciclagem do papel.

Outro exemplo é o metal, as maiores taxas de reciclagem atualmente estão presentes em resíduos de alumínio, onde, por exemplo, temos a reciclagem de mais de 95% de todas as latas de alumínio para bebidas. Isso ocorre principalmente por ser possível sua reciclagem indefinida vezes sem a perda de qualidade, em um processo barato de fundição do metal.

Já no vidro o cenário não é tão animador, apesar de ser um dos materiais mais considerados sustentáveis. Acontece que ele é economicamente viável para reciclagem enquanto está inteiro, se ele é descartado quebrado, o procedimento para coleta, manipulação e triagem acaba ficando caro por colocar em risco a saúde dos cooperados que realizam a separação. Esse “caro” é em relação principalmente ao custo baixíssimo de se obter o vidro, diferente do alumínio. Então se forem descartar vidro garantam que ele estará envolto por jornal ou uma embalagem resistente de plástico para que os cooperados não se machuquem na manipulação.

Também temos a caixa de leite, a maioria das embalagens longa vida é feita a partir de uma mistura de materiais com propriedades diversas. Mesmo assim, é possível reciclá-las. É importante descartar os materiais recicláveis limpos, para não ocorrer a proliferação de doenças, odores, bem como para evitar a contaminação de itens recicláveis que estejam no mesmo local, pois caso ocorra a contaminação, a reciclagem dos materiais contaminados fica mais difícil.

Como disposto acima sobre a diferença entre os conceitos de reciclagem e reaproveitamento, em alguns casos, não é possível reciclar indefinidamente o material. Isso acontece, por exemplo, com o papel, que tem algumas de suas propriedades físicas minimizadas a cada processo de reciclagem, devido ao inevitável encurtamento das fibras de celulose.

Em outros casos, felizmente, isso não acontece. A reciclagem do alumínio, por exemplo, não acarreta nenhuma perda de suas propriedades físicas, e esse pode, assim, ser reciclado continuamente.

3 ELEMENTOS DE PROGRAMAÇÃO ORIENTADO A OBJETO

Um paradigma de programação fornece (e determina) a forma como o programador vai enxergar e estruturar o programa, assim como podemos usar de várias perspectivas e soluções diferente para resolver um problema. Ao criar um programa, podemos usar um determinado paradigma de programação para desenvolvê-lo.

O paradigma da Orientação a Objetos tem várias características marcantes: Alta reutilização, Desenvolvimento sempre evolutivo, Construções complexas baseadas em construções simples, de fácil manutenção.

3.1 CLASSE

O programa desenvolvido é dividido em várias partes chamadas classes, essas classes possuem semânticas que podem ser reaproveitadas em novos programas.

As classes possuem, mas não devem incluir todos os atributos e nem todas as ações dos objetos do mundo real, somente o que for pertinente ao papel a ser desempenhado dentro do programa.

A classe passa a ser um molde para se criar objetos do mesmo tipo, como por exemplo a Classe carro que pode gerar “N” tipos de carros, cada qual com suas próprias especificações.

Uma classe por ser genérica, pode pertencer a vários sistemas, desde que tenha as devidas adaptações, exemplo de classe e de situação a ser usada: Classe de nome Carro que poderia ser usada numa locadora de veículos e em uma montadora.

3.2 PACKAGE

Os pacotes são usados na organização de nosso projeto, assim como pastas ou diretórios. Inclusive influenciam no acesso de uma classe a outra, porém há também a possibilidade de importar classes dos outros pacotes de acordo com a necessidade.

Quando uma classe está localizada em um pacote, nela precisa estar definida (em código) esta situação, identificando o pacote em que ela se localiza através da palavra reservada `package`.

Quando vamos utilizar (ou referenciar) ao longo da Classe, uma outra Classe que está em outro pacote, ou em outra Biblioteca, somos obrigados a importar a Classe, utilizando a palavra reservada `import`. E indicando o caminho até a Classe, ou importando todas as classes do respectivo pacote.

3.3 ATRIBUTOS

Atributos são características presentes nos objetos, os valores de todos os atributos são chamados de estado do objeto e são sinônimos de variáveis. Variáveis são espaços ou alocações nas quais os dados são armazenados.

A declaração de uma variável instrui o programa a reservar um espaço na memória, para que seja possível armazenar um dado de determinado tipo. A quantidade de memória reservada para uma variável é definida pelo seu tipo. Declaração de Variáveis pode ser feita em qualquer parte do corpo de uma classe.

Somente atributos que são de interesse do sistema devem ser considerados, por isso deve-se abstrai-los para que o software não tenha redundâncias que interfiram no desempenho.

3.4 CLASSE ABSTRATA

As classes abstratas são criadas quando não pretendemos criar objetos a partir delas, são normalmente usadas como superclasses e se caracterizam por serem muito genéricas

Uma classe abstrata é uma classe que não pode ser instanciada, ou seja, não pode ser usada como molde para um objeto. Se definirmos um método abstrato obrigatoriamente sua classe deve ser declarada como abstrata.

Um exemplo de classe abstrata seria a classe `Funcionário` que pode ter como subclasses as classes `gerente`, `presidente`, `atendente` etc.

3.5 MÉTODOS

Os métodos são conjuntos de instruções que cumprem uma tarefa específica de uma classe e retornam ou não um valor onde foram chamados.

Os métodos podem receber dados para serem utilizados internamente, os quais são chamados de parâmetros ou de argumentos. Quando os parâmetros são passados para os métodos, é criada uma cópia dos valores para serem manipulados de acordo com as especificações do método e retornarem o resultado estipulado.

Os métodos também podem ou não assumir tipos de dados, caso não assumam, são chamados de procedimentos, porque executam um conjunto de instruções sem retornar valor algum após ser chamado. Métodos que não retornam nada e apenas efetuam ações, recebem o tipo void. Podemos passar vários parâmetros para os métodos, inclusive de tipos diferentes.

3.6 CONSTRUTOR

Os construtores são métodos, porém não devem ser chamados como métodos durante o programa, pois serão considerados como erro pelo software. São executados automaticamente quando um objeto é criado(instanciado), eles não possuem retorno, não possuem tipo e podem pedir parâmetros na instanciação caso forem exigidos pelo programador. Colocar um tipo de retorno explicitará em alterar a assinatura dele e com isso ele não será acionado quando a classe for instanciada.

Uma classe consegue ter vários construtores, desde que recebam números diferentes de parâmetros quando forem declarados. Assim quando o Objeto for instanciado, dependendo do número de parâmetros ele vai ser gerado com base no seu respectivo construtor, caso não forem passados parâmetros na instanciação e não houver um construtor vazio, resultara em erro. Caso um construtor não seja declarado o Java gera um vazio.

O método construtor obrigatoriamente deve ter o mesmo nome da classe e não pode ter tipo de retorno.

3.7 SOBRECARGA

Sobrecarregar métodos significa ter mais de um método com a assinatura igual. A assinatura de um método é composta pelo tipo de retorno, nome e parâmetros.

Ter uma sobrecarga de método implicará em erro no programa. A sobrecarga de método não se aplica ao método construtor.

3.8 MÉTODO ABSTRATO

Os métodos abstratos definidos em uma classe abstrata devem obrigatoriamente implementados em uma classe concreta. Mas se uma classe abstrata herdar outra classe abstrata, a classe que herda não precisa implementar os métodos abstratos. Indica que todas as classes filhas concretas devem reescrever esse método e ele deve ser um método concreto nas classes filhas.

Os métodos que são abstratos têm um comportamento diferente nas classes filhas, por isso não possuem corpo.

3.9 INTERFACE

As interfaces definem a especificação de funcionalidades, sem a implementação das mesmas. Elas são usadas com o único objetivo de obrigar o programador a implementar métodos que são especificados na interface ao sistema. No escopo da interface, o modo no qual os métodos são declarados possuem diferença do modo convencional, sendo somente necessário a declaração da assinatura dos métodos terminados com “;” sem a necessidade de abrir e fechar.

Apenas métodos públicos podem ser declarados nelas, mas não podem ser definidos. Todos os métodos declarados dentro de uma interface são, implicitamente, public e abstract. Da mesma forma, não é possível definir atributos – apenas constantes públicas. Todos os atributos declarados dentro de uma interface são, implicitamente, public, static e final. Uma constante é um valor que não muda, é fixo, diferente de uma variável. Um exemplo de constante é o PI.

Diferença entre uma classe abstrata e uma interface Java é que a interface obrigatoriamente não pode ter um “corpo” associado. Enquanto uma classe abstrata é “estendida” (palavra-chave `extends`) por classes derivadas, uma interface Java é “implementada” (palavra chave `implements`) por outras classes.

Interfaces x Classes Abstratas

Use classes abstratas quando você quiser definir um “template” para subclasses e você possui alguma implementação (métodos concretos) que todas as subclasses podem utilizar.

Use interfaces quando você quiser definir uma regra que todas as classes que implementem a interface devem seguir, independentemente se pertencem a alguma hierarquia de classes ou não.

3.10 OBJETO

Um objeto é o resultado das especificações de uma classe, o objeto é gerado seguindo os atributos e métodos impostos pela classe. As classes e os Objetos possuem uma relação de dependência, pois não existe objeto sem a classe.

O ato de instanciar um objeto, é nada mais que a criação de um Objeto, durante a instanciação do objeto são passados como parâmetros os atributos que são exigidos na classe usada como molde. Após a instanciação o objeto que foi criado tem acesso a todas os métodos especificados na classe à qual pertence.

Um exemplo de objeto é uma pessoa de nome qualquer, vamos chamar o Objeto de “Fernando”, Fernando pode pertencer a várias classes, mas isso vai depender do software que está sendo desenvolvido. Neste exemplo Fernando pertence a Classe genérica “pessoa, uma pessoa possui características específicas como tamanho, peso, cor, altura etc. e possui ações específicas como andar, falar, dormir entre várias outras ações.

Então quando o Objeto Fernando for instanciado, será necessário que se passe como parâmetro as características específicas de Fernando, assim que o Objeto Fernando for criado ele terá acesso aos métodos andar, falar e dormir podendo usá-los como for necessário.

3.11 INSTÂNCIA

O ato de instanciar um objeto é nada mais que criar um objeto em memória com base em um molde (classe) podendo ou não exigir parâmetros durante a instância.

3.12 ENCAPSULAMENTO

O encapsulamento é um dos conceitos mais relevantes no paradigma da orientação a objetos e tem a função de delimitar o acesso a dados importantes do sistema para que não sofram acessos indevidos. Desse modo os dados encapsulados não podem ser acessados de outras classes. Esses dados geralmente necessitam de condições específicas para poderem ser acessados como os métodos setters e getters que podem ser acessados de qualquer outra classe, sem causar inconsistência no desenvolvimento do código.

O getter tem por objetivo retornar o valor que lhe foi pedido, Sem prejudicar a integridade do dado em si, já o setter recebe como parâmetro uma informação sem que se possa burlar.

Um exemplo de situação em que o encapsulamento deve ser aplicado é no saldo e saque de uma conta, já que o ato de sacar é totalmente dependente do saldo da conta. Se o saque for burlado sem ter o saldo da conta verificado, qualquer valor poderá ser sacado independente do saldo.

3.13 MODIFICADORES DE ACESSO

Os Modificadores de acesso servem para definir o acesso que determinada classe, atributo ou método terá diante do resto do programa.

3.13.1Public

Quando o Modificador de acesso é declarado, seja numa classe, atributo ou método, ele pode ser acessado em qualquer lugar ou por qualquer objeto que possa enxergar a classe em que este elemento faz parte.

Uma declaração (Classes, atributos ou métodos) com o modificador public pode ser acessada de qualquer lugar e por qualquer

entidade (Objeto) que possa visualizar a Classe a que este elemento pertence

Pode ser aplicado em: classe, pacote, subclasse e outros...

3.13.2 Private

Uma declaração (atributos ou métodos) com o modificador `private` não pode ser acessado ou usado por nenhuma outra Classe, mas apenas por métodos da própria Classe. Esses atributos e métodos também não podem ser diretamente visualizados pelas Classes herdadas.

Pode ser aplicado em: classe

3.13.3 Protected

O modificador `protected` torna o elemento (atributo ou métodos) acessível as classes do mesmo pacote ou através de herança. Seus elementos não são acessíveis a outras classes fora do pacote em que foram declarados.

Pode ser aplicado em: Classe, Pacote e Subclasse

3.13.4 Default

Uma declaração (Classes, atributos ou métodos) em que não há definição de modificador, possui o que chamamos de modificador, possui o que chamamos de modificador padrão, fazendo com que sejam acessíveis somente por Classes do mesmo pacote.

Pode ser aplicado em: Classe e Pacote

3.14 HERANÇA

Como o próprio nome sugere, na orientação a objetos o termo herança se refere a algo herdado, como de pai para filho.

Em Java, a herança acontece assim que uma classe passa a herdar ou seja, ter acesso a características (atributos e métodos) de uma outra classe, assim que ocorre a herança a classe à qual herdou os atributos e métodos pode usufruí-los e até modificá-los internamente.

A classe fornecedora dos recursos recebe o nome de superclasse e a receptora dos recursos de subclasse.

Uma subclasse herda, mas ainda pode:

- Adicionar novos métodos e atributos

- Redefinir a implementação de métodos já existentes

Para fazer uso da herança deve-se usar a palavra-chave `extends` logo após a declaração da classe. Se uma classe não contém a declaração `extends`, automaticamente ela herda as características da classe `Object`.

A relação entre a Herança e Modificadores de Acesso ocorre da seguinte forma:

`Protected`: podem ser acessados pelos membros da subclasse.

`Private`: só podem ser acessados pela própria classe.

3.15 HERANÇA MÚLTIPLA

Algumas linguagens Orientadas a Objeto têm suporte a herança múltipla como o C++, o que significa que uma subclasse pode herdar características de duas ou mais classes. Isso permite que uma classe agrupe atributos e métodos de várias classes. No caso o Java não oferece o suporte a herança múltipla.

3.16 POLIMORFISMO

No polimorfismo duas ou mais subclasses podem chamar métodos com a mesma assinatura de método, mas com comportamentos diferentes, comportamentos esses que são específicos para cada subclasse.

A palavra polimorfismo nos remete a várias formas, e na Orientação a Objetos o polimorfismo aplica uma situação em que um objeto recebendo uma mensagem retorna uma resposta diferente.

Um exemplo, o método `mover`, num jogo é responsável por movimentar o personagem. Entretanto, em uma GUI (Graphical User Interface) o usuário pode mover arquivos.

3.17 SOBRESCRITA

A sobrescrita é uma funcionalidade usada durante o polimorfismo, é uma palavra reservada e sua sintaxe "`@override`", deve ser usada antes do método, que deve contar a mesma assinatura do método da superclasse.

Com a sobrescrita, conseguimos especializar os métodos herdados das superclasses, alterando o seu comportamento nas subclasses por um mais específico.

3.18 EXCEÇÕES

Erros são inevitáveis, ao decorrer do desenvolvimento de um programa, mas um bom programador deve estar atento a estes erros. Estes erros devem ser devidamente tratados, do contrário podem parar um servidor inteiro.

Seguem alguns exemplos destes erros:

Erros devido a condições do ambiente de execução fogem ao controle do programador, mas podem ser contornados, como um arquivo não encontrado ou conexão com o Banco de dados não estabelecida. Erros graves, onde não há recuperação fogem ao controle do programador e não podem ser contornados então devem ser devidamente tratados como a falta de memória ou um erro interno da JVM.

Ou erros de lógica de programação que podem gerar uma exceção, como por exemplo uma divisão por zero, podem e devem ser tratados pelo programador.

No caso da exceção não ser tratada, ocorrerá um erro que implicará no interrompimento da execução de todo o programa, levando a parada de todo um servidor.

4 DISSERTAÇÃO

O Brasil gerou, em 2018, 79 milhões de toneladas de lixo por ano, um aumento de quase 1% em relação ao ano anterior, segundo o Panorama dos Resíduos Sólidos 2018, elaborado pela Associação Brasileira de Empresas de Limpeza Pública e Resíduos Especiais (Abrelpe). Deste total, a estimativa é de que somente 3% sejam de fato reciclados, sendo que o potencial é de até 30%. “Não mudou muito a visão de que basta ter lixeiras e o sistema de coleta já está resolvido. Não está”, diz Ana Maria Luz, presidente do Instituto GEA — Ética e Meio Ambiente, organização que tem como finalidade desenvolver a educação ambiental.

Os dados são alarmantes. O Brasil é hoje o quarto maior produtor de lixo plástico, segundo um estudo da World Wildlife Fund (WWF): são 11,3 toneladas por ano, das quais somente 1,28% são recicladas. O número está bem abaixo da média mundial, de 9%. E, embora quase três quartos dos municípios façam algum tipo de coleta seletiva, a maioria se concentra no Sul e Sudeste. No Centro Oeste, menos da metade das cidades tem coleta seletiva.

O problema é que a maioria dos brasileiros ainda desconhece o funcionamento da reciclagem. Uma pesquisa do Ibope de 2018 mostra que 66% da população sabe pouco ou nada sobre coleta seletiva, e 39% não separam o lixo. Outro levantamento, este de 2019 feito pelo instituto Ipsos, revelou que 54% dos brasileiros não entendem como funciona a reciclagem em sua região — no restante do mundo, esse índice é de 47%, em média. “O sistema todo funciona mal porque não investimos em educação, se [o *cidadão*] não sabe nem o dia da coleta seletiva, ele não vai separar o lixo”.

Falta informação também sobre o que é reciclável ou não. Embalagens de salgadinho e balas, por exemplo, são feitas de um plástico muito mole, que tem pouco valor comercial para ser reciclado, então raramente o são. Já embalagens sujas perdem a capacidade de reaproveitamento (sim, é preciso lavar o pote de iogurte antes de jogá-lo na lixeira). E, se o rótulo é impresso diretamente na embalagem (como nos potes de margarina), a tinta do material impede a reciclagem.

Na outra ponta, porém, existe um desinteresse político e industrial no tema pela falta de vantagens econômicas da reciclagem. Enquanto algumas embalagens têm logística de reaproveitamento consagrada, como produtos de aço, alumínio e papelão, outras são descartadas pela falta de retorno econômico, como o plástico. “Se o valor pago por elas é baixo, não há motivação para que catadores separem o produto”, diz Giansesi. E esse fato é comprometedor e plausível, Pois essa é uma das explicações para o Brasil ser um dos maiores recicladores de alumínio do mundo e o lanterninha quando o assunto é plástico: segundo a Cempre, 1 quilo de alumínio é vendido por R\$ 3,7, em média; já a mesma quantidade de garrafas PET (segundo material de maior valor) rende, no máximo, R\$ 1,8. “É preciso haver um incentivo para que as pessoas atuem nessa área de negócios, com mecanismos que favoreçam as indústrias que consumam o material reciclado e o tornem competitivo em relação ao material virgem”, explica o presidente da ABLP.

E se a reciclagem em for uma farsa? Diante de tamanha dificuldade em fechar os ciclos para tornar a cadeia completa, há quem questione se a reciclagem não passa de um mito: como não funciona na prática, só serviria para “aliviar” a consciência dos consumidores, que param de se preocupar com o destino de seus resíduos.

É o ponto levantado pelo documentário *A Farsa da Reciclagem*, da série *Desserviço ao Consumidor*, disponível na Netflix. A produção mostra como grandes corporações introduziram plásticos descartáveis no mercado e investiram em propaganda defendendo que não há problema em seu consumo desenfreado — basta reciclá-los. Mas elas, muitas vezes, se isentam da responsabilidade de fazer isso acontecer. “É injusto que recaia nas costas da sociedade os custos de uma empresa que está ganhando em cima disso”, opinam Luz.

Isso significa, então, que devemos parar de nos preocupar com separar o lixo e seguir nossas vidas como se nada fosse acontecer (como de fato dificilmente acontece)? Não exatamente. O plástico é só a ponta de uma montanha de lixo — e a mais visível, por se tratar de um material leve que se espalha facilmente e boia na água. Existem alternativas para embalagens menos

agressivas ao meio ambiente, como o próprio caso da Keurig demonstra. O que falta é interesse econômico e político em adotá-las.

Lidar com o destino do lixo também é um passo essencial. No Brasil, o que não é reciclado vai parar em aterros sanitários, lixões ou os chamados aterros controlados (que, apesar do nome, não têm um controle tão rígido assim da contaminação do meio ambiente ou das pessoas que trabalham ali). O mesmo relatório da Abrelpe revela que 59,5% (118,631 toneladas) do lixo produzido no país por dia acaba em aterros sanitários; 23% (45,830 toneladas) vão para os aterros controlados e 17,5% (34,850 toneladas) se destinam aos lixões – que devem ser extintos até 2021, de acordo com a Política Nacional de Resíduos Sólidos.

Plástico: Sem o manejo correto, os resíduos podem cair em rios e córregos e, cedo ou tarde, chegam ao mar. Um estudo publicado em 2016 no periódico *Science of the Total Environment* estima que 80% do plástico no oceano vem de lixo gerado em áreas terrestres – apenas 20% são de atividades marítimas. Não à toa, existem hoje ilhas de plástico no Oceano Pacífico com tamanho equivalente a duas vezes o território da França.

Também não adianta acreditar que uma cruzada contra canudinhos ou sacolinhas plásticas vai dar conta do problema – nenhum item sozinho é responsável pelo problema. “Isso desvia a atenção do público para o que realmente importa”, diz Luz. O problema é sistêmico, e todos têm sua parcela de responsabilidade na solução: a indústria, o governo e você.

A lei, que está em vigor desde o dia 16 de julho, prevê descontos para quem optar por não usar sacolas plásticas.... O objetivo da lei, segundo o governo, não é acabar com as sacolas plásticas e, sim, educar a população e reduzir a utilização do plástico, porém os a lei optou apenas na diminuição de sacolas plásticas, não adotando um regime mais severo para tirar de vez essa a sacola plástica, além de criar uma renovação de sacolas ou até mesmo substituindo a sacola plástica por um material menos prejudicial ao meio ambiente.

Vidros: O Brasil produz aproximadamente 800.000 toneladas de embalagens de vidro anualmente e 27,6% (220,8 mil toneladas) desse total corresponde as embalagens de vidro recicladas. Deste montante, 5% é gerado

por engarrafadores de bebidas, 10% por sucateiros, 0,6% oriundo de coletas promovidas pelas vidrarias e 12% provém de refugos de vidro gerados nas fábricas. Dos outros 72,4%, parte é descartada, parte é reutilizada domesticamente e parte é retornável.

O vidro não se decompõe na natureza por isso que devemos separar tudo o que é de vidro e dar um descarte correto reciclando. O consumidor, consciente das características ecológicas e econômicas da reciclagem, ao adquirir um produto ecológico contribuirá com o meio ambiente, o que também é um exercício de cidadania.

No Brasil, os principais problemas oriundos da mineração podem ser englobados em quatro categorias: poluição da água, poluição do ar, poluição sonora, e subsidência do terreno. Em geral, a mineração provoca um conjunto de efeitos não desejados que podem ser denominados de externalidades. Algumas dessas externalidades são: alterações ambientais, conflitos de uso do solo, depreciação de imóveis circunvizinhos, geração de áreas degradadas e transtornos ao tráfego urbano. Estas externalidades geram conflitos com a comunidade, que normalmente têm origem quando da implantação do empreendimento, pois o empreendedor não se informa sobre as expectativas, anseios e preocupações da comunidade que vive nas proximidades da empresa de mineração.

Papel: De diferentes gramaturas, tamanhos e cores, ele sempre está no nosso dia a dia de alguma forma. Utilizado para anotar recados, estudar ou levar mensagens, o papel é um material muito utilizado, ainda que o método online ajude na economia do material.

Aliás, em relação ao impacto ambiental, muitas pessoas associam à extração da madeira, responsável pela celulose, matéria-prima do papel. Porém, há um dado ainda mais alarmante em relação ao ciclo do papel. Segundo o Instituto Akatu, organização não-governamental que trabalha para um consumo consciente, a cada um quilo de papel produzido, 540 litros de água são utilizados. Ou seja, se uma empresa que consome 600 mil folhas de papel por ano, imprimir os documentos em frente e verso, é possível gerar uma economia 384 mil litros por ano, o equivalente ao abastecimento para 30 famílias no mesmo período.

Com a produção abusiva do papel, surge o desmatamento é caracterizado pela remoção da vegetação nativa de uma área. A sua causa está atrelada principalmente à ação antrópica, ou seja, à atuação do homem no desenvolvimento das atividades produtivas. As consequências do desmatamento estão ligadas à perda da biodiversidade e, conseqüentemente, à extinção de espécies. Além disso, o desmate provoca um amplo conjunto de impactos ambientais negativos e é apontado como um dos grandes responsáveis pelas mudanças climáticas.

O processo de desmatamento foi iniciado mediante a necessidade de matérias-primas para a produção de diferentes elementos de uso da sociedade. Desse modo, a exploração da madeira, por exemplo, para a fabricação de papel.

A principal consequência do desmatamento está atrelada ao desequilíbrio ambiental provocado pela perda da vegetação nativa. A remoção da vegetação provoca uma grande perda da biodiversidade assim como a perda do habitat de animais e plantas, e, ainda, impacta diretamente na elevação do número de espécies em extinção.

Além disso, o desmatamento **acelera a ocorrência de processos naturais** que são intensificados pela ação humana. A remoção da vegetação impacta diretamente no aumento da erosão e da desertificação, por exemplo. No mais, o desmatamento interfere no ciclo hidrológico e ocasiona efeitos como o esgotamento das fontes de água, já que a retirada da vegetação dificulta a absorção da água da chuva pelo subsolo e o conseqüente abastecimento das reservas subterrâneas e das nascentes.

Metal: O elevado desenvolvimento industrial ocorrido nas últimas décadas, tem sido um dos principais responsáveis pela contaminação de nossas águas e solos, seja pela negligência no seu tratamento antes de despejá-las nos rios ou por acidentes e descuidos cada vez mais frequentes, que propiciam o lançamento de muitos poluentes nos ambientes aquáticos.

De acordo com dados da Associação Brasileira de Empresas de Tratamento, Recuperação e Disposição de Resíduos Especiais (ABETRE), dos 2,9 milhões de toneladas de resíduos industriais perigosos gerados anualmente no Brasil, cerca de 600 mil toneladas recebem tratamento adequado. O restante é depositado em lixões.

Dentre os poluentes podemos citar os metais pesados, outro grande problema para a saúde humana. Metais pesados são elementos químicos metálicos, de peso atômico relativamente alto, que em concentrações elevadas são muito tóxicos á vida. As atividades industriais, têm introduzido metais pesados nas águas numa quantidade muito maior do que aquela que seria natural, causando grandes poluições. Para se ter uma ideia disso, basta lembrar que os metais pesados fazem parte dos despejos de grandes indústrias, em todos os países do mundo. A ação dos metais pesados na saúde humana é muito diversificada e profunda. Entre os mais perigosos estão o mercúrio, o cádmio (encontrado em baterias de celulares), cromo e o chumbo. Os metais pesados diferem de outros agentes tóxicos porque não são sintetizados nem destruídos pelo homem. A atividade industrial diminui significativamente a permanência desses metais nos minérios, bem como a produção de novos compostos, além de alterar a distribuição desses elementos no planeta.

Entre os mais perigosos estão:

Chumbo, encontrado na indústria de baterias automotivas, chapas de metal semiacabado, canos de metal, cable sheating, aditivos em gasolina, munição;

- Cádmio, proveniente da fundição e refinação de metais como zinco, chumbo e cobre;
- Mercúrio, encontrado na mineração e utilizado em derivados na indústria e na agricultura;
- Cromo, no processo de curtição de couros;
- Zinco, presente na fundição e refinação no setor metalúrgico e nas indústrias recicladoras de chumbo.

A maior parte dos metais pesados são venenosos aos seres humanos. Eles diferenciam-se dos compostos orgânicos tóxicos, por serem absolutamente não degradáveis, de maneira que podem acumular no meio ambiente onde manifestam sua toxicidade. Isso significa que eles têm elevados níveis de reatividade e bioacumulação, o que faz com que permaneçam em caráter cumulativo ao longo da cadeia alimentar.

Os problemas gerados pelos metais são inúmeros. Além de prejudicar o meio ambiente, esses elementos influenciam negativamente na vida humana. As

principais fontes de exposição a esses metais são os alimentos. Dessa forma, é cada vez maior o número de pessoas infectadas com doenças e problemas provenientes desses elementos.

Normalmente os mais comuns são considerados de curto e longo prazo, como a diarreia. No entanto, as incidências de médio e longo prazo estão cada vez maiores. O problema é que esses efeitos são pouco evidentes e difíceis de serem distinguidos. Frequentemente estão associados à quantidade ingerida e podem atingir vários órgãos. Além disso, podem ser transportados pelo ar através de partículas em suspensão, podendo contaminar o homem pelas via aéreas.

A contaminação pelos metais pesados não ocorre somente através de desastres ambientais, ela é causada também pelo descarte indevido de produtos industrializados como pilhas, baterias e lâmpadas no lixo doméstico. Eles acabam entrando em contato com o solo, onde materiais constituintes de metais pesados acabam contaminando assim o solo, as águas e os seres vivos.

Concluiremos, que a reciclagem é importante e pode contribuir para a redução da poluição, para o planeta e o aquecimento da economia. Lembre-se de que todos têm um papel importante para que esse processo ocorra.

Reciclar o lixo é uma ação que depende de todos para diminuir os danos ao meio ambiente. Em tempos em que os recursos naturais diminuem e a quantidade de resíduos aumenta, a conscientização em torno da reciclagem precisa crescer o mais rápido possível.

Em nossas próprias casas e locais de trabalho é possível separar o lixo reciclável e encaminhar para os locais adequados de destinação (coleta seletiva, cooperativas, pontos de recepção, etc). Juntos vamos acabar com os impactos causados ao meio-ambiente.

5 PROJETO (ESTRUTURA) DO PROGRAMA

O projeto apresenta 2 pacotes principais, sendo **control** e o **model**.

O pacote **control** contém a classe main (**Main.java**) responsável pela execução do programa.

O pacote **model** contém uma interface (**InterLixos.java**), uma classe abstrata (**Lixo.java**) e **outras 7 classes** como demonstrado na imagem.

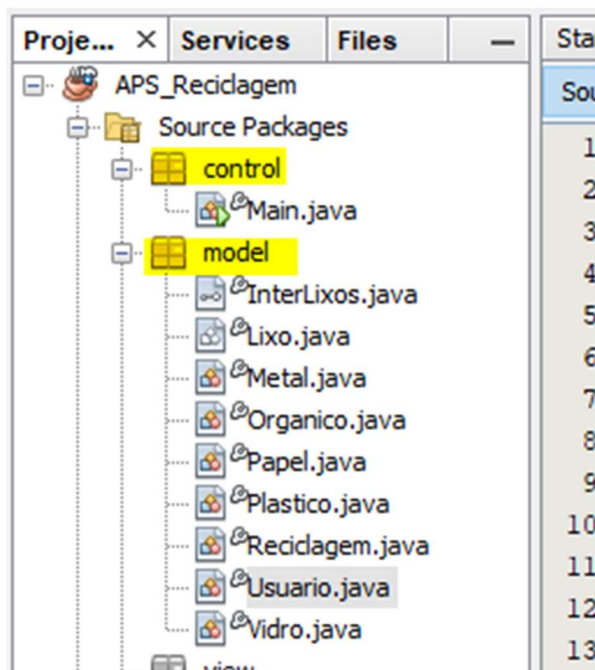


Figura 1 - Packages

A interface (**InterLixos.java**) define que qualquer classe que a implemente tenha que obrigatoriamente executar o método **void motra.Separacao();**

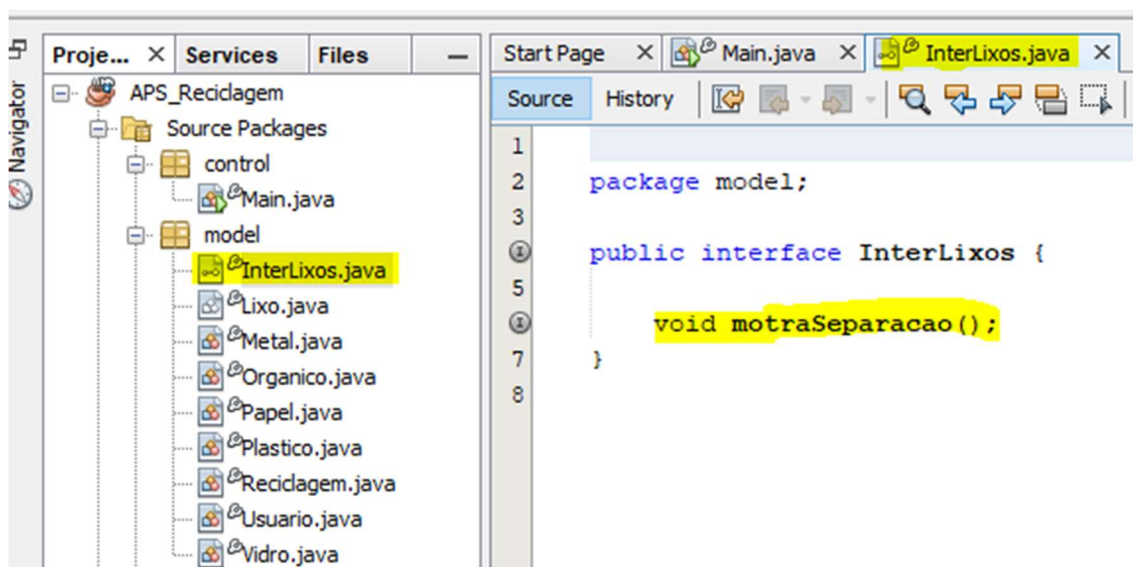


Figura 2 – InterLixos.java

A **classe abstrata (Lixo.java)** funciona como uma classe mãe que **faz com que todas as classes filhas herdem seus atributos e os métodos**, sejam eles obrigatórios ou não.

Essa classe **possuí 2 atributos (String nomeLixo; e int qtd;)**, os **Getter and Setter** dos atributos **fazendo o encapsulamento e os tornando privados**, e um **método construtor que recebe dois parâmetros** sendo uma sequência de caracteres (**String n1**) e um número inteiro (**int qt**), **além de implementar a interface (InterLixos.java)**.

```

1 package model;
2
3 public abstract class Lixo implements InterLixos {
4     private String nomeLixo;
5     private int qtd;
6
7     public Lixo(String nl, int qt) {
8         this.setNomeLixo(nl);
9         this.setQtd(qt);
10    }
11
12    public String getNomeLixo() {
13        return nomeLixo;
14    }
15
16    public void setNomeLixo(String nomeLixo) {
17        this.nomeLixo = nomeLixo;
18    }
19
20    public int getQtd() {
21        return qtd;
22    }
23
24    public void setQtd(int qtd) {
25        this.qtd = qtd;
26    }
27
28

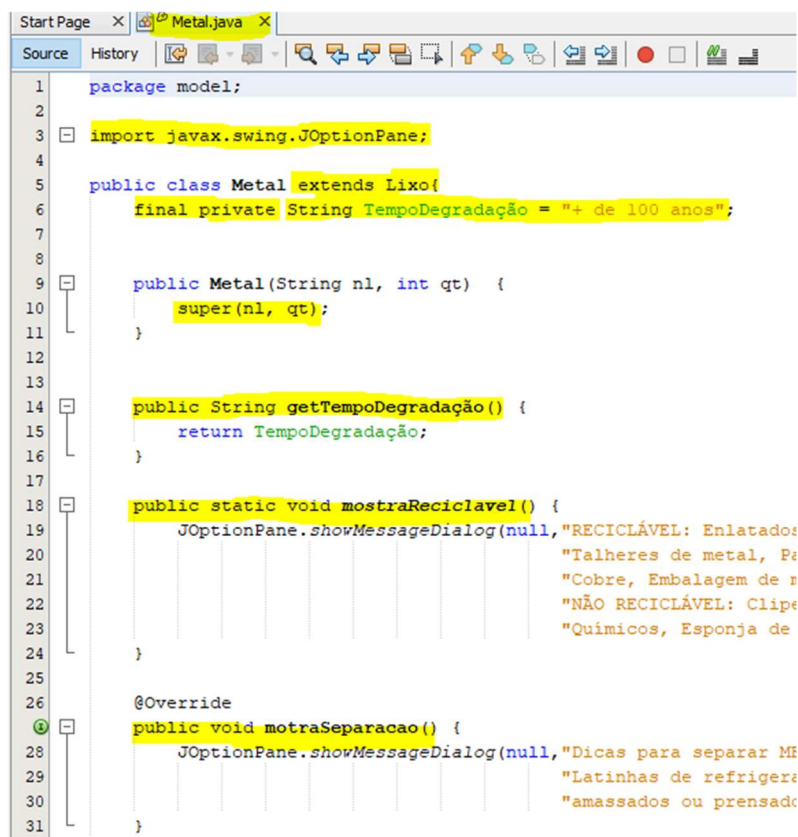
```

Figura 3 – Lixo.java

A classe filha (**Metal.java**) herda a classe mãe abstrata (**Lixo.java**) herdando seus atributos e métodos, incluindo o método obrigatório (**void motra.Separacao();**) da interface (**InterLixos.java**) implementado na classe mãe.

Ela possui 1 atributo com parâmetro definido (**String TempoDegradação = "+ de 100 anos";**) declarado como final private que não pode ser alterado, um **método construtor que recebe dois parâmetros** sendo uma sequência de caracteres (**String n1**) e um número inteiro (**int qt**) **como super**, fazendo dessa forma que **os parâmetros sejam lidos pela classe mãe que contém os atributos**.

Possui um método público (**String getTempoDegradação()**) que retorna parâmetro definido do atributo (**String TempoDegradação**), e dois métodos de impressão (**mostraRecicavel()** e **motraSeparacao()**) que se utilizam da importação da biblioteca **javax.swing.JOptionPane**;



```

1 package model;
2
3 import javax.swing.JOptionPane;
4
5 public class Metal extends Lixo{
6     final private String TempoDegradação = "+ de 100 anos";
7
8
9     public Metal(String n1, int qt) {
10         super(n1, qt);
11     }
12
13
14     public String getTempoDegradação() {
15         return TempoDegradação;
16     }
17
18     public static void mostraRecicavel() {
19         JOptionPane.showMessageDialog(null, "RECICLÁVEL: Enlatados:
20         | Talheres de metal, Pa
21         | Cobre, Embalagem de r
22         | NÃO RECICLÁVEL: Clipe
23         | Químicos, Esponja de
24     }
25
26     @Override
27     public void motraSeparacao() {
28         JOptionPane.showMessageDialog(null, "Dicas para separar M
29         | Latinhass de refrigere
30         | amassados ou prensado
31     }

```

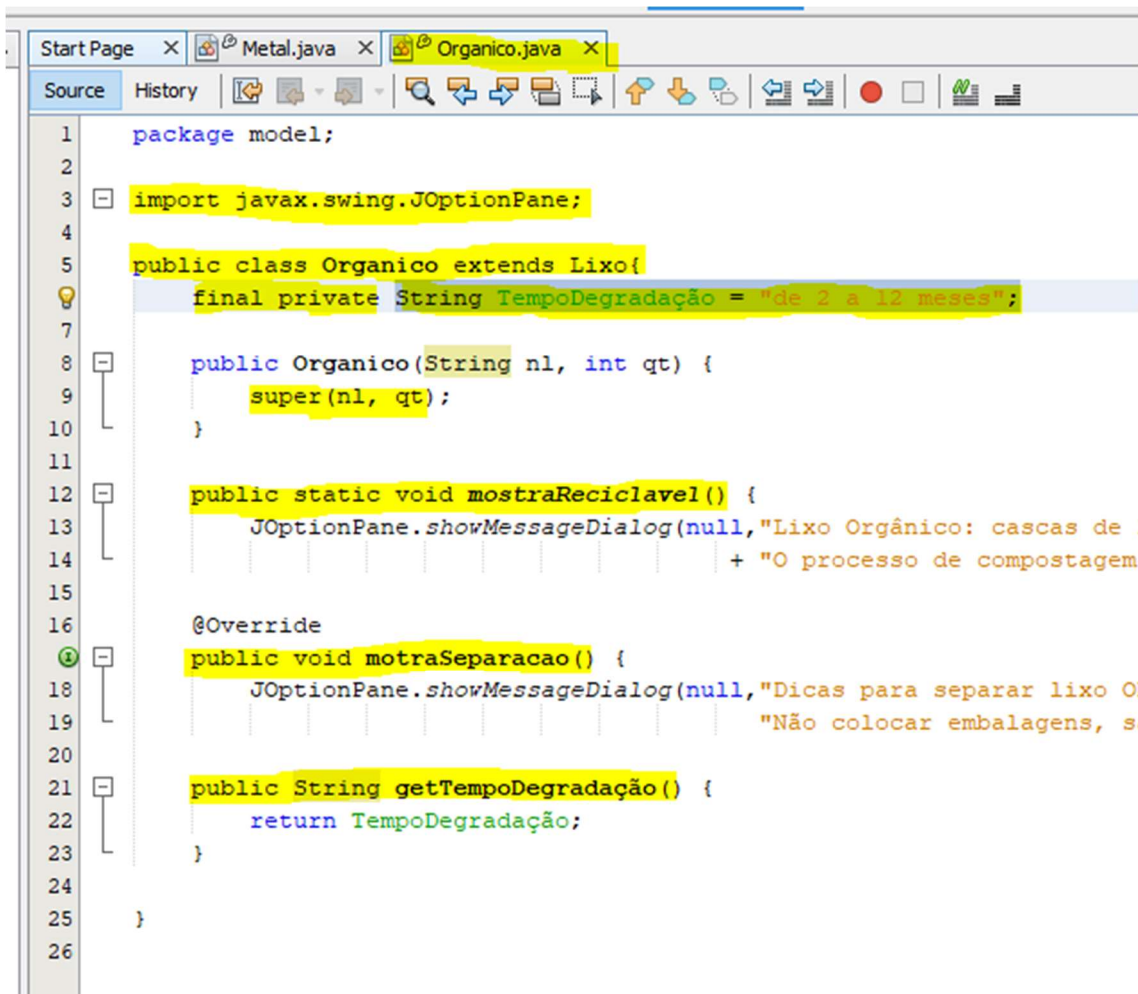
Figura 4 – Metal.java

A classe filha (**Organico.java**) herda a classe mãe abstrata (**Lixo.java**) herdando seus atributos e métodos, incluindo o método obrigatório (**void motra.Separacao();**) da interface (**InterLixos.java**) implementado na classe mãe.

Ela possui 1 atributo com parâmetro definido (**String TempoDegradação = "de 2 a 12 meses";**) declarado como final private que não pode ser alterado, um método construtor que recebe dois parâmetros sendo uma sequência de caracteres (**String n1**) e um número inteiro (**int qt**) como **super**, fazendo dessa forma que os parâmetros sejam lidos pela classe mãe que contém os atributos.

Possui um método público (**String getTempoDegradação()**) que retorna parâmetro definido do atributo (**String TempoDegradação**), e dois

métodos de impressão (**mostraReciclavel()** e **motraSeparacao()**) que se utilizam da importação da biblioteca **javax.swing.JOptionPane**;



```

1 package model;
2
3 import javax.swing.JOptionPane;
4
5 public class Organico extends Lixo{
6     final private String TempoDegradação = "de 2 a 12 meses";
7
8     public Organico(String n1, int qt) {
9         super(n1, qt);
10    }
11
12    public static void mostraReciclavel() {
13        JOptionPane.showMessageDialog(null, "Lixo Orgânico: cascas de :
14        + "O processo de compostagem
15
16    @Override
17    public void motraSeparacao() {
18        JOptionPane.showMessageDialog(null, "Dicas para separar lixo O
19        "Não colocar embalagens, s
20
21    public String getTempoDegradação() {
22        return TempoDegradação;
23    }
24
25 }
26

```

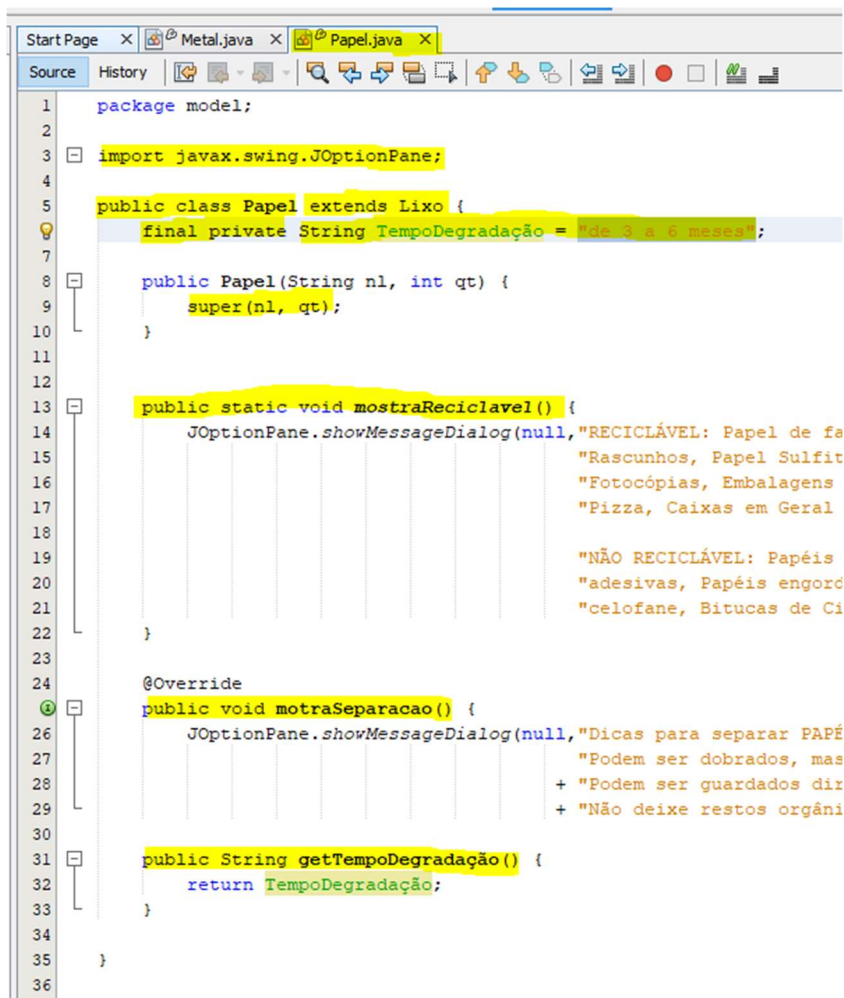
Figura 5 - Organico.java

A classe filha (**Papel.java**) herda a classe mãe abstrata (**Lixo.java**) herdando seus atributos e métodos, incluindo o método obrigatório (**void motra.Separacao();**) da interface (**InterLixos.java**) implementado na classe mãe.

Ela possui 1 atributo com parâmetro definido (**String TempoDegradação = "de 3 a 6 meses";**) declarado como final private que não pode ser alterado, um método construtor que recebe dois parâmetros sendo uma sequência de caracteres (**String n1**) e um número inteiro (**int qt**) como **super**, fazendo dessa forma que os parâmetros sejam lidos pela classe mãe que contém os atributos.

Possui um método público (**String getTempoDegradação()**) que retorna parâmetro definido do atributo (**String TempoDegradação**), e dois

métodos de impressão (**mostraReciclavel()** e **motraSeparacao()**) que se utilizam da importação da biblioteca **javax.swing.JOptionPane**;



```

1 package model;
2
3 import javax.swing.JOptionPane;
4
5 public class Papel extends Lixo {
6     final private String TempoDegradação = "de 3 a 6 meses";
7
8     public Papel(String n1, int qt) {
9         super(n1, qt);
10    }
11
12
13    public static void mostraReciclavel() {
14        JOptionPane.showMessageDialog(null, "RECICLÁVEL: Papel de fa
15        "Rascunhos, Papel Sulfit
16        "Fotocópias, Embalagens
17        "Pizza, Caixas em Geral
18
19        "NÃO RECICLÁVEL: Papéis
20        "adesivas, Papéis engord
21        "celofane, Bitucas de Ci
22    }
23
24    @Override
25    public void motraSeparacao() {
26        JOptionPane.showMessageDialog(null, "Dicas para separar PAPÉ
27        "Podem ser dobrados, mas
28        + "Podem ser guardados dir
29        + "Não deixe restos orgâni
30
31    public String getTempoDegradação() {
32        return TempoDegradação;
33    }
34
35 }
36

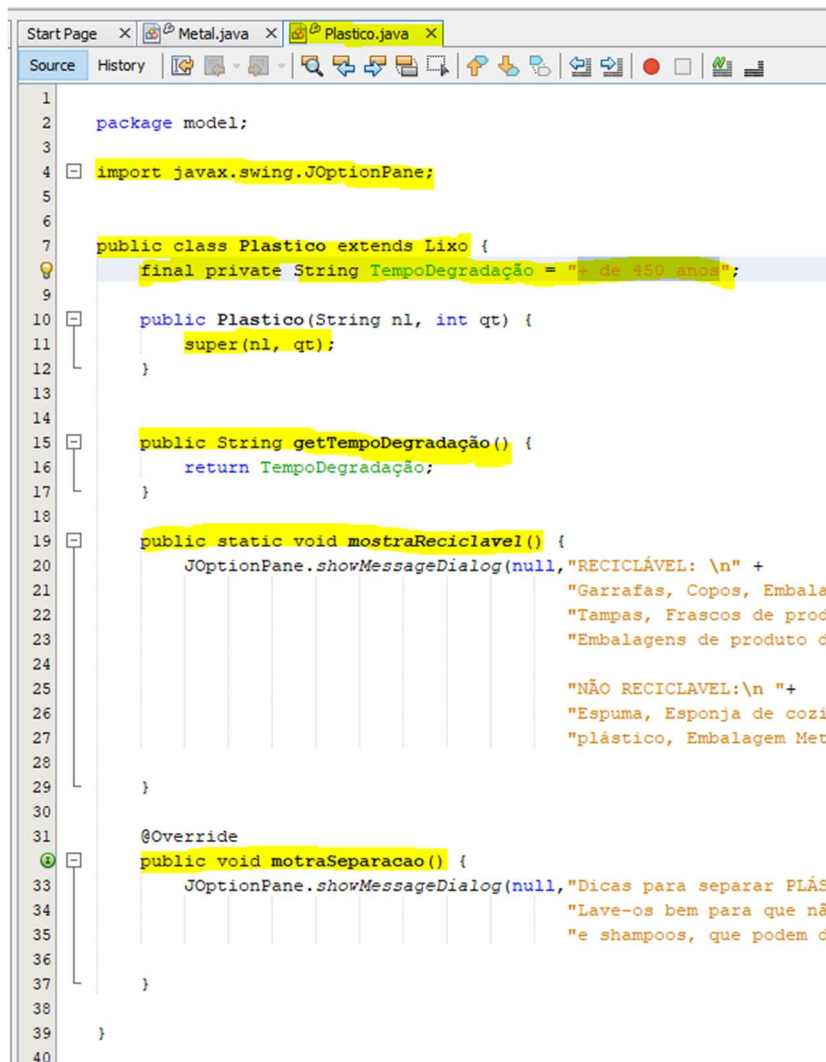
```

Figura 6 - Papel.java

A classe filha (**Plastico.java**) herda a classe mãe abstrata (**Lixo.java**) herdando seus atributos e métodos, incluindo o método obrigatório (**void motra.Separacao();**) da interface (**InterLixos.java**) implementado na classe mãe.

Ela possui 1 atributo com parâmetro definido (**String TempoDegradação = " + de 450 anos ";**) declarado como final private que não pode ser alterado, **um método construtor que recebe dois parâmetros** sendo uma sequência de caracteres (**String n1**) e um número inteiro (**int qt**) **como super**, fazendo dessa forma que **os parâmetros sejam lidos pela classe mãe que contém os atributos**.

Possui um método público (**String getTempoDegradação()**) que retorna parâmetro definido do atributo (**String TempoDegradação**), e dois métodos de impressão (**mostraReciclavel()** e **motraSeparacao()**) que se utilizam da importação da biblioteca **javax.swing.JOptionPane**;



```

1
2 package model;
3
4 import javax.swing.JOptionPane;
5
6
7 public class Plastico extends Lixo {
8     final private String TempoDegradação = "de 450 anos";
9
10    public Plastico(String nl, int qt) {
11        super(nl, qt);
12    }
13
14
15    public String getTempoDegradação() {
16        return TempoDegradação;
17    }
18
19    public static void mostraReciclavel() {
20        JOptionPane.showMessageDialog(null, "RECICLÁVEL: \n" +
21            "Garrafas, Copos, Embala
22            "Tampas, Frascos de prod
23            "Embalagens de produto d
24
25            "NÃO RECICLÁVEL: \n" +
26            "Espuma, Esponja de cozi
27            "plástico, Embalagem Met
28
29    }
30
31    @Override
32    public void motraSeparacao() {
33        JOptionPane.showMessageDialog(null, "Dicas para separar PLÁS
34            "Lave-os bem para que nã
35            "e shampoos, que podem d
36
37    }
38
39
40 }

```

Figura 7 – Plástico.java

A classe **Reciclagem.java** utiliza da importação da biblioteca **javax.swing.JOptionPane** e da **java.util.ArrayList**. Possui atributos que instancia ArrayLists das classes: Metal, Organico, Papel, Plástico e Vidro. Serve para armazenar em uma lista os materiais reciclados pelo objeto (**usu**) instanciado pela classe de controle (**Main.java**).

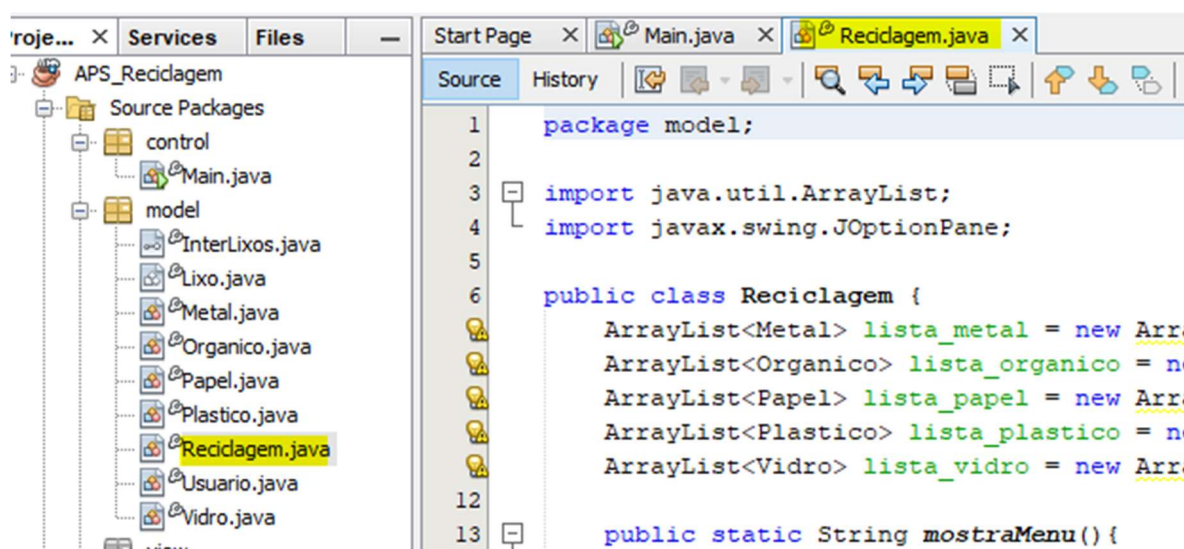


Figura 8 – Reciclagem.java

Possui um método **public static mostraMenu()**, não recebe nenhum parâmetro e retorna um dado tipo String. Caso escolha a segunda opção, irá mostrar a segunda parte do menu. O valor retornado é a concatenação dos valores inseridos.

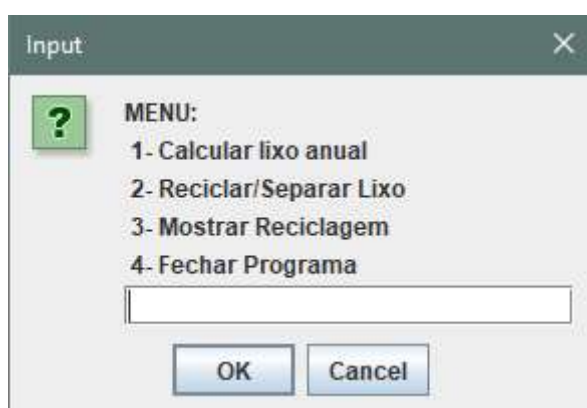


Figura 10 - Ex.: Menu 1

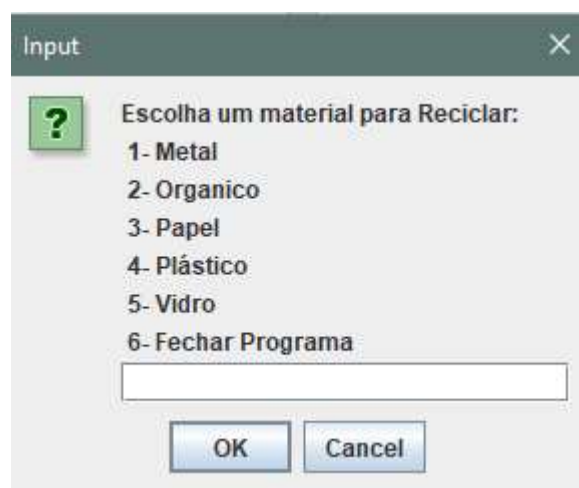
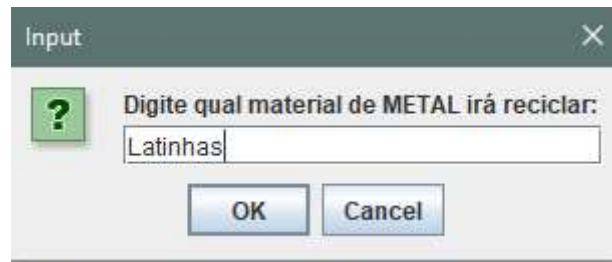


Figura 9 - Ex.: Menu 2

Possui métodos (**addLista_metal()**, **mostrarLista_metal()**, **addLista_organico()**, **mostrarLista_organico()**, **addLista_papel()**, **mostrarLista_papel()**, **addLista_plastico()**, **mostrarLista_plastico()**, **addLista_vidro()**, **mostrarLista_vidro()**), que não recebem nenhum parâmetro e retorna nenhum dado. Utilizam-nos apenas para adicionar e mostrar valores em uma determinada ArrayList com interações do usuario.



Input

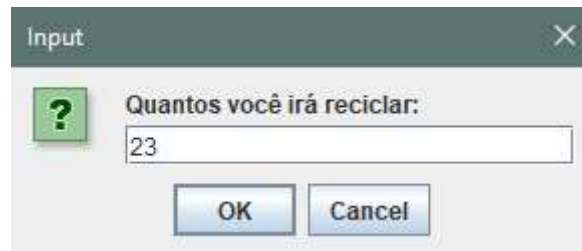
?

Digite qual material de METAL irá reciclar:

Latinhas

OK Cancel

Figura 11 – Ex.: Adicionando material a reciclagem



Input


?

Quanto você irá reciclar:

23

OK Cancel

Figura 12 - Ex.: Adicionando a quantidade do material



control.Main > main > while (mantem_menu == true) > if (null == valor_menu) >

Output - APS_Reciclagem (run) X

```
run:
-----
METAL - Tempo de degradação: + de 100 anos
Nome | quantidade
Latinhas | 23
Colher | 6

-----
Não reciclou nenhum MATERIAL ORGÂNICO.

-----
Não reciclou nenhum PAPEL.

-----
PLÁSTICO - Tempo de degradação: + de 450 anos
Nome | quantidade
Garrafa PET | 33
Tampas de Garrafa | 40
Sacos | 7

-----
VIDRO - Tempo de degradação: + de 1000 anos
Nome | quantidade
Garrafas de vidro | 5

-----
```

Message

i

Toda sua coleta de lixo para reciclagem foi mostrada no console

OK

Figura 13 - Ex.: Mostrar Reciclagem

A classe **Usuario.java** utiliza da importação da biblioteca **javax.swing.JOptionPane**. Possui atributos privados: nome, idade, kilos. Possui um atributo que instancia a classe **Reciclagem** em um objeto 'rec'. Assim que criar um objeto da classe **Usuario**, irá também instanciar o objeto 'rec' junto a ele.

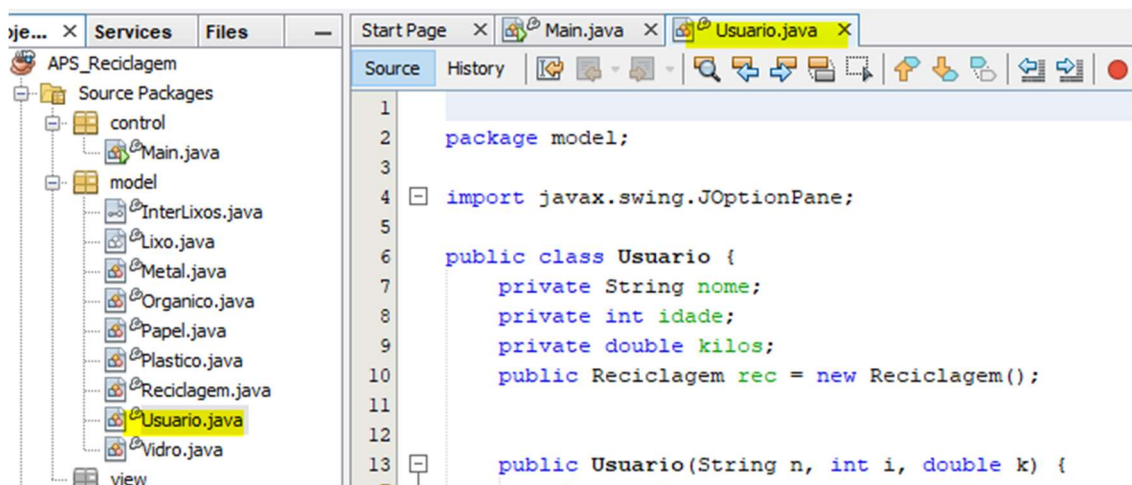


Figura 14 - Usuario.java

Possui métodos **kg_anuais()**, **kg_vida()** e **volume()**, que não recebem parâmetro, porém realizam cálculos e retornam dado tipo double. Possui também seus **getters** e **setters** com um método construtor com encapsulamento de dados

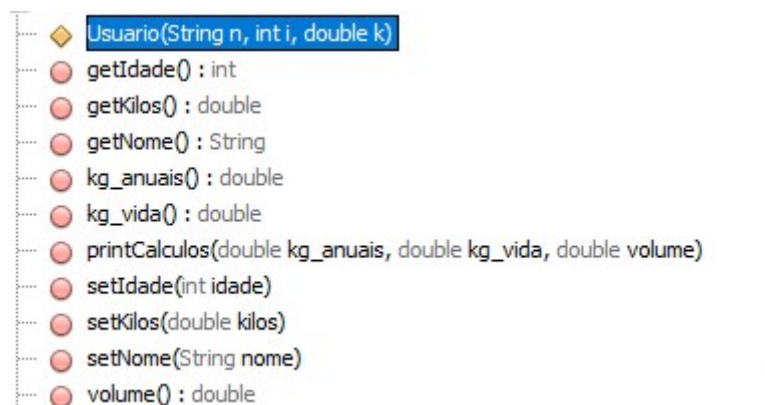


Figura 15 - Métodos Usuario

Outro método **public void printCalculos (double kg_anuais, double kg_vida, double volume)** que recebe 3 parâmetros tipo double e não retorna nada, serve para mostrar os cálculos para o usuario.

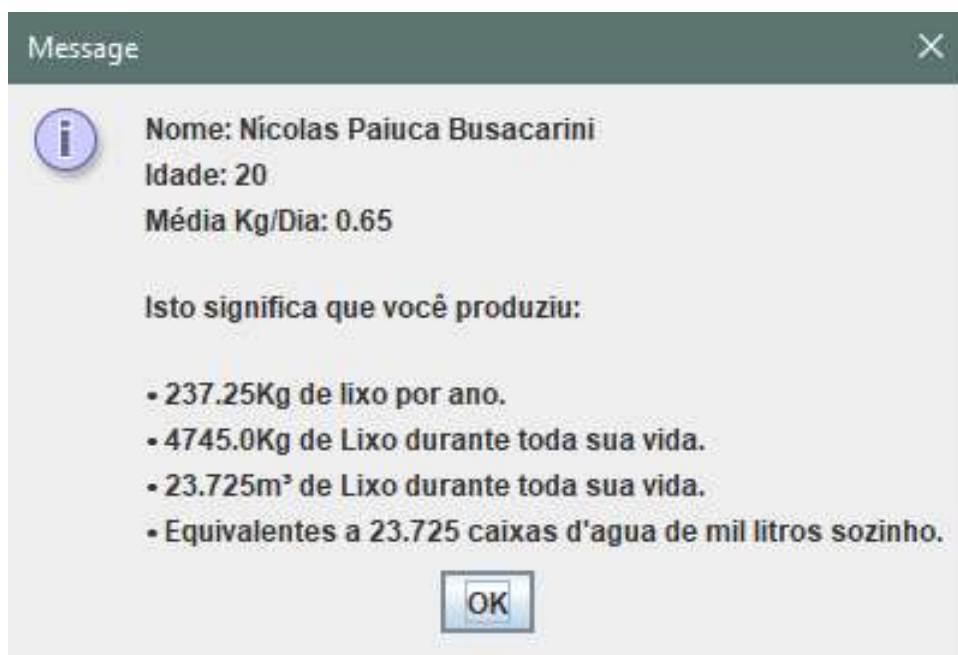


Figura 16 – Ex.: PrintCalculos

A classe filha (**Vidro.java**) herda a classe mãe abstrata (**Lixo.java**) herdando seus atributos e métodos, incluindo o método obrigatório (**void mostra.Separacao();**) da interface (**InterLixos.java**) implementado na classe mãe.

Ela possui 1 atributo com parâmetro definido (**String TempoDegradação = " + de 1000 anos ";**) declarado como final private que não pode ser alterado, **um método construtor que recebe dois parâmetros** sendo uma sequência de caracteres (**String n1**) e um número inteiro (**int qt**) **como super**, fazendo dessa forma que **os parâmetros sejam lidos pela classe mãe que contém os atributos**.

Possui um método público (**String getTempoDegradação()**) que retorna parâmetro definido do atributo (**String TempoDegradação**), e dois métodos de impressão (**mostraReciclavel()** e **mostraSeparacao()**) que se utilizam da importação da biblioteca **javax.swing.JOptionPane**;

```

1
2  package model;
3
4  import javax.swing.JOptionPane;
5
6
7  public class Vidro extends Lixo {
8      final private String TempoDegradação = "4 de 1000 anos"
9
10     public Vidro(String nl, int qt) {
11         super(nl, qt);
12     }
13
14     public String getTempoDegradação() {
15         return TempoDegradação;
16     }
17
18     public static void mostraReciclavel() {
19         JOptionPane.showMessageDialog(null, "RECICLÁVEL: Emk
20                                         "Frascos de remé
21                                         "NÃO RECICLÁVEL:
22                                         "Temperados, Pir
23                                         "monitores.");
24     }
25
26
27     @Override
28     public void motraSeparacao() {
29         JOptionPane.showMessageDialog(null, "Dicas para sepa
30                                         "Lave-os bem e r
31
32
33     }
34

```

Figura 17 - Vidro.java

A classe Main.java é a classe controladora onde possui o método public static void main. Está no package control e importa o método static **mostraMenu** na classe Reciclagem, a classe **Usuario** e a biblioteca **javax.swing.JOptionPane**

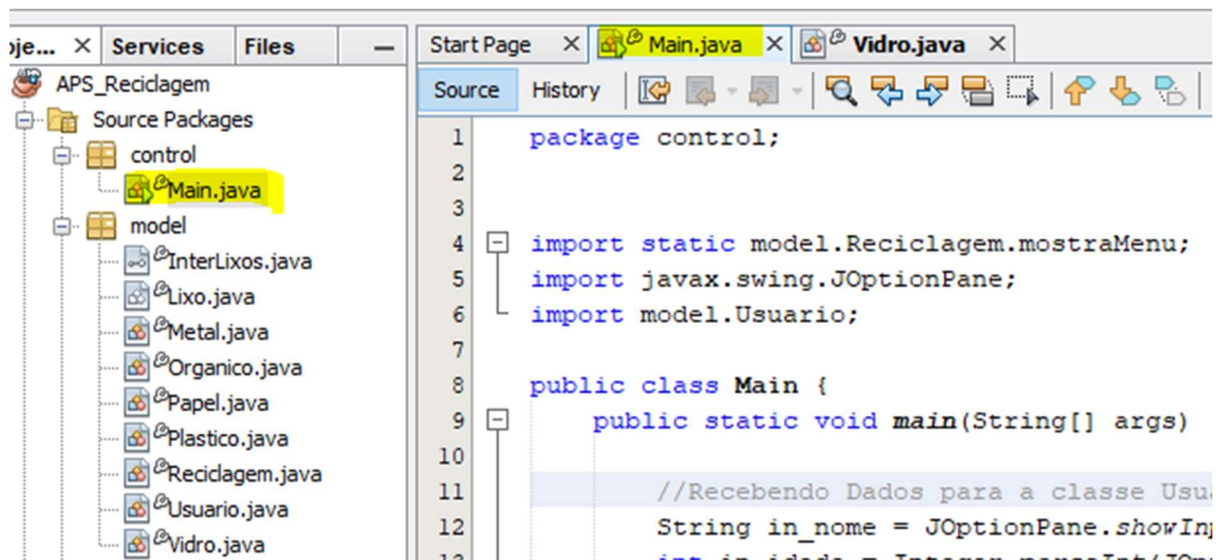


Figura 19 - Main.java

Primeiro começa recebendo dados e usa-os para instanciar um objeto da classe usuario.

```
//Recebendo Dados para a classe Usuario
String in_nome = JOptionPane.showInputDialog(null, "Insira seu Nome: ");
int in_idade = Integer.parseInt(JOptionPane.showInputDialog(null, "Insira sua idade: "));
double in_kilos = Double.parseDouble(JOptionPane.showInputDialog(null, "Insira em média o peso do seu lixo diário(Kg): "));
Usuario usu = new Usuario(in_nome, in_idade, in_kilos);
```

Figura 18 - Instanciar Usuario

Depois declara uma variável booleana que serve para controlar o laço while que está logo na próxima linha de código. Enquanto mantem_menu não receber o valor 'false', não irá sair do laço e irá continuar exibindo o menu para o usuario. Em seguida entra em um switch case para controlar o que exibir em cada opção do menu utilizando métodos e prints.

```
String valor_menu;
boolean mantem_menu = true;
while (mantem_menu == true) {
    valor_menu = mostraMenu();

    if (null == valor_menu) {
        JOptionPane.showMessageDialog(null, "Valor(es) inseridos menu incorreto");
    } else switch (valor_menu) {
        case "1":
            // 1 - Calcula Lixo anual
            usu.printCalculos(usu.kg_anuais(), usu.kg_vida(), usu.volume());
            JOptionPane.showMessageDialog(null, "É muito lixo, não é? \n"
                + "Talvez você queira diminuir lixo que você produz.\n"
                + "E você pode começar hoje!\n"
                + "Praticando a COLETA e RECICLAGEM irá diminuir seu lixo");
            break;
        case "3":
```

Figura 20 - While e Switch Case

Caso o valor inserido pelo usuário não seja esperado então exibirá uma mensagem de erro e voltará no menu para que ele possa corrigir.

```
default:  
    // Valores Incorretos  
    JOptionPane.showMessageDialog(null, "Valor(es) no menu incorreto(s) ou nulos. Tente de novo e escolha  
    break;
```

Figura 21 - Switch Case: default

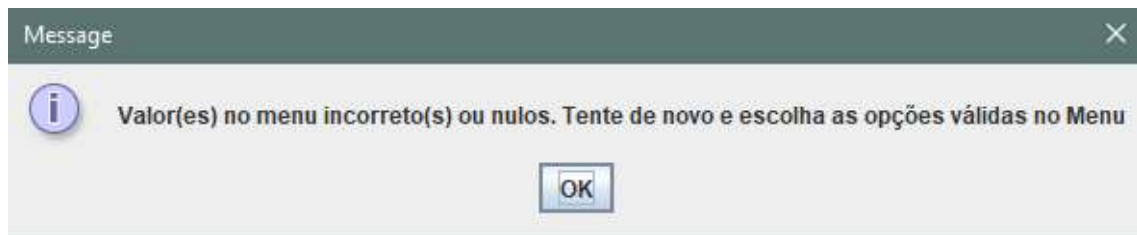


Figura 22 - Mensagem de erro

Se desejar sair do menu, então o programa irá se encerrar exibindo a mensagem de finalização.

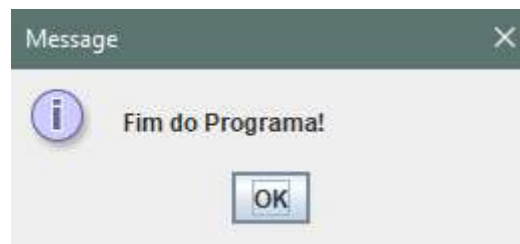


Figura 23 - Finalização do programa

6 RELATORIO COM AS LINHAS DE CÓDIGO DO PROGRAMA

6.1 PACKAGE CONTROL

6.1.1 Classe: 'Main'

```
package control;

import static model.Reciclagem.mostraMenu;
import javax.swing.JOptionPane;
import model.Usuario;

public class Main {
    public static void main(String[] args) {

        //Recebendo Dados para a classe Usuario
        String in_nome = JOptionPane.showInputDialog(null, "Insira seu
Nome: ");
        int in_idade = Integer.parseInt(JOptionPane.showInputDialog(null,
"Insira sua idade: "));
        double in_kilos =
Double.parseDouble(JOptionPane.showInputDialog(null, "Insira em média o
peso do seu lixo diário(Kg): "));
        Usuario usu = new Usuario(in_nome, in_idade, in_kilos);

        String valor_menu;
        boolean mantem_menu = true;
        while (mantem_menu == true){
            valor_menu = mostraMenu();

            if (null == valor_menu){
```

```

JOptionPane.showMessageDialog(null, "Valor(es) no menu
incorreto(s) ou nulo. Tente de novo e escolha uma das opções válidas no Menu");
}else switch (valor_menu) {
case "1":
    // 1 - Calcula Lixo anual
    usu.printCalculos(usu.kg_anuais(),          usu.kg_vida(),
usu.volume());

    JOptionPane.showMessageDialog(null, "É muito lixo, não
é? \n"

        + "Talvez você queira diminuir lixo que você produz.\n"
        + "E você pode começar hoje!\n"
        + "Praticando a COLETA e RECICLAGEM irá diminuir
seu lixo e ajudará salvar nosso planeta.\n");
    break;
case "3":
    // 3 - Mostra todas as ArrayList
    usu.rec.mostrarLista_metal();
    usu.rec.mostrarLista_organico();
    usu.rec.mostrarLista_papel();
    usu.rec.mostrarLista_plastico();
    usu.rec.mostrarLista_vidro();
    System.out.println("\n-----
");

    JOptionPane.showMessageDialog(null,"Toda sua coleta de
lixo para reciclagem foi mostrada no console");
    break;
case "21":
    // 21 - adiciona objeto METAL à reciclagem
    usu.rec.addLista_metal();
    break;
case "22":
    // 22 - adiciona objeto ORGANICO à reciclagem
    usu.rec.addLista_organico();

```

```

        break;
    case "23":
        // 23 - adiciona objeto PAPEL à reciclagem
        usu.rec.addLista_papel();
        break;
    case "24":
        // 24 - adiciona objeto PLÁSTICO à reciclagem
        usu.rec.addLista_plastico();
        break;
    case "25":
        // 25 - adiciona objeto VIDRO à reciclagem
        usu.rec.addLista_vidro();
        break;
    case "26":
        // 26 - FECHA PROGRAMA
        mantem_menu = false;
        break;
    case "4":
        // 4 - FECHA PROGRAMA
        mantem_menu = false;
        break;
    default:
        // Valores Incorretos
        JOptionPane.showMessageDialog(null, "Valor(es) no menu
        incorreto(s) ou nulos. Tente de novo e escolha as opções válidas no Menu");
        break;
    }

}

JOptionPane.showMessageDialog(null, "Fim do Programa!");
}
}

```


6.2 PACKAGE MODEL

6.2.1 Interface: 'InterLixos'

```
package model;

public interface InterLixos {

    void motraSeparacao();
}
```

6.2.2 Classe Abstrata: 'Lixo'

```
package model;

public abstract class Lixo implements InterLixos {
    private String nomeLixo;
    private int qtd;

    public Lixo(String nl, int qt) {
        this.setNomeLixo(nl);
        this.setQtd(qt);
    }

    public String getNomeLixo() {
        return nomeLixo;
    }

    public void setNomeLixo(String nomeLixo) {
        this.nomeLixo = nomeLixo;
    }

    public int getQtd() {
        return qtd;
    }
}
```

```

        public void setQtd(int qtd) {
            this.qtd = qtd;
        }

    }

```

6.2.3 Classe: Metal

```

package model;

import javax.swing.JOptionPane;

public class Metal extends Lixo{
    final private String TempoDegradação = "+ de 100 anos";

    public Metal(String nl, int qt) {
        super(nl, qt);
    }

    public String getTempoDegradação() {
        return TempoDegradação;
    }

    public static void mostraReciclavel() {
        JOptionPane.showMessageDialog(null,"REICLÁVEL: Enlatados,
Tampinhas de Garrafas, Chapas, Latas, Ferragens, Arames,\n" +
            "Talheres de metal, Panelas sem cabo, Papel
alumínio limpo, Canos, Pregos, Aerossóis,\n" +
            "Cobre, Embalagem de marmitex.\n\n"+

```

```

        "NÃO RECICLÁVEL: Clipes, Tachinhas,
        Latas de inseticidas, Grampos, Latas de solventes\n" +
        "Químicos, Esponja de Aço, Latas de
        Verniz.");
    }

    @Override
    public void motraSeparacao() {
        JOptionPane.showMessageDialog(null,"Dicas para separar
        METAIS:\n" +
        "Latinhas de refrigerantes, cervejas e
        enlatados devem ser\n" +
        "amassados ou prensados para facilitar o
        armazenamento.");
    }

}

```

6.2.4 Classe: 'Organico'

```

package model;

import javax.swing.JOptionPane;

public class Organico extends Lixo{
    final private String TempoDegradação = "de 2 a 12 meses";

```

```

public Organico(String nl, int qt) {
    super(nl, qt);
}

public static void mostraReciclavel() {
    JOptionPane.showMessageDialog(null,"Lixo Orgânico: cascas de
legumes, restos de comida, frutas, cascas de ovos, etc. \n\n"
+ "O processo de compostagem pode ser
definido como a reciclagem de matéria orgânica."); }

@Override
public void motraSeparacao() {
    JOptionPane.showMessageDialog(null,"Dicas para separar lixo
ORGÂNICO:\n" +
+Não colocar embalagens, sacolas e
receptientes que não poderá ser feita a compostagem. "); }

public String getTempoDegradação() {
    return TempoDegradação;
}

}

```

6.2.5 Classe: 'Papel'

```

package model;

import javax.swing.JOptionPane;

public class Organico extends Lixo{
    final private String TempoDegradação = "de 2 a 12 meses";

    public Organico(String nl, int qt) {
        super(nl, qt);
    }
}

```

```

    }

    public static void mostraReciclavel() {
        JOptionPane.showMessageDialog(null,"Lixo Orgânico: cascas de
legumes, restos de comida, frutas, cascas de ovos, etc. \n\n"
        + "O processo de compostagem pode ser
definido como a reciclagem de matéria orgânica."); }

    @Override
    public void motraSeparacao() {
        JOptionPane.showMessageDialog(null,"Dicas para separar lixo
ORGÂNICO:\n" +
        "Não colocar embalagens, sacolas e
receptientes que não poderá ser feita a compostagem. "); }

    public String getTempoDegradação() {
        return TempoDegradação;
    }
}

```

6.2.6 Classe: 'Plastico'

```

package model;

import javax.swing.JOptionPane;

public class Plastico extends Lixo {
    final private String TempoDegradação = "+ de 450 anos";

    public Plastico(String nl, int qt) {
        super(nl, qt);
    }
}

```

```
}
```

```
public String getTempoDegradação() {
    return TempoDegradação;
}
```

```
public static void mostraReciclavel() {
    JOptionPane.showMessageDialog(null,"REICLÁVEL: \n" +
        "Garrafas, Copos, Embalagens Pet
(Refrigerantes, Vinagre,Óleo...), Sacos/Sacolas,\n" +
        "Tampas, Frascos de produtos, Caneta (Sem
a tinta), Canos e Tubos de PVC, \n" +
        "Embalagens de produto de limpeza,
Embalagens tipo Tupperware,Brinquedos de plástico, Baldes\n\n" +

        "NÃO REICLAVEL:\n "+
        "Espuma, Esponja de cozinha, Tomadas,
Acrílico, Bandejas de\n" +
        "plástico, Embalagem Metalizada (Café e
Salgadinho), Cabos de Panela, Isopor.");
}
```

```
@Override
```

```
public void motraSeparacao() {
    JOptionPane.showMessageDialog(null,"Dicas para separar
PLÁSTICOS:\n" +
        "Lave-os bem para que não fiquem restos do
produto, principalmente as embalagens de detergentes\n" +
        "e shampoos, que podem dificultar a triagem
e o aproveitamento do material.");
}
```

```
}
```

```
}
```

6.2.7 Classe: 'Reciclagem'

```
package model;
```

```
import java.util.ArrayList;
```

```
import javax.swing.JOptionPane;
```

```
public class Reciclagem {
```

```
    ArrayList<Metal> lista_metal = new ArrayList<Metal>();
```

```
    ArrayList<Organico> lista_organico = new ArrayList<Organico>();
```

```
    ArrayList<Papel> lista_papel = new ArrayList<Papel>();
```

```
    ArrayList<Plastico> lista_plastico = new ArrayList<Plastico>();
```

```
    ArrayList<Vidro> lista_vidro = new ArrayList<Vidro>();
```

```
    public static String mostraMenu(){
```

```
        String escolha = JOptionPane.showInputDialog(null, "MENU: \n 1-  
Calcular lixo anual \n 2- Reciclar/Separar Lixo \n 3- Mostrar Reciclagem \n 4-  
Fechar Programa");
```

```
        if("2".equals(escolha)){
```

```
            escolha += JOptionPane.showInputDialog(null, "Escolha um  
material para Reciclar: \n 1- Metal \n 2- Organico \n 3- Papel \n 4- Plástico \n 5-  
Vidro \n 6- Fechar Programa");
```

```
        }
```

```
        return escolha;
```

```
    }
```

```
    public void addLista_metal(){
```

```
        Metal.mostraReciclavel();
```

```
        String nome_metal = JOptionPane.showInputDialog(null, "Digite  
qual material de METAL irá reciclar: ");
```

```

        int qtd_metal = Integer.parseInt(JOptionPane.showInputDialog(null,
"Quantos você irá reciclar: "));
        Metal m = new Metal(nome_metal, qtd_metal);
        lista_metal.add(m);
        m.mostraSeparacao();
        JOptionPane.showMessageDialog(null, "Reciclagem
concluída!\nVocê pode reciclar quantas vezes quiser.");

    }

```

```

    public void mostrarLista_metal(){
        System.out.println("\n-----");
        if(lista_metal.size() > 0){
            System.out.println("METAL - Tempo de degradação: " +
lista_metal.get(0).getTempoDegradação());
            System.out.println("Nome | quantidade");
            for(int i = 0; i < lista_metal.size(); i++) {
                System.out.print(lista_metal.get(i).getNomeLixo() + " | " +
lista_metal.get(i).getQtd()+"\n");
            }
        }else{
            System.out.println("Não reciclou nenhum METAL.");
        }
    }
}

```

```

    public void addLista_organico(){
        Organico.mostraReciclavel();
        String nome_organico = JOptionPane.showInputDialog(null, "Digite
qual material ORGÂNICO você irá reciclar: ");
        int qtd_organico =
Integer.parseInt(JOptionPane.showInputDialog(null, "Quantos você irá reciclar:
"));
        Organico o = new Organico(nome_organico, qtd_organico);
    }
}

```



```

        lista_organico.add(o);
        o.mostraSeparacao();
        JOptionPane.showMessageDialog(null, "Reciclagem
concluida!\nVocê pode reciclar quantas vezes quiser.");

    }

    public void mostrarLista_organico(){
        System.out.println("\n-----");
        if(lista_organico.size() > 0){
            System.out.println("ORGÂNICO - Tempo de degradação: " +
lista_organico.get(0).getTempoDegradação());
            System.out.println("Nome | quantidade");
            for(int i = 0; i < lista_organico.size(); i++) {
                System.out.print(lista_organico.get(i).getNomeLixo() + " | " +
lista_organico.get(i).getQtd()+"\n");
            }
        }else{
            System.out.println("Não   reciclou   nenhum   MATERIAL
ORGÂNICO.");
        }
    }

    public void addLista_papel(){
        Papel.mostraReciclavel();
        String nome_papel = JOptionPane.showInputDialog(null, "Digite
qual material de PAPEL você irá reciclar: ");
        int qtd_papel = Integer.parseInt(JOptionPane.showInputDialog(null,
"Quantos você irá reciclar: "));
        Papel p = new Papel(nome_papel, qtd_papel);
        lista_papel.add(p);
        p.mostraSeparacao();
        JOptionPane.showMessageDialog(null,"Reciclagem concluida!\n

```

```

Você pode reciclar quantas vezes quiser.");
    }

    public void mostrarLista_papel(){
        System.out.println("\n-----");
        if(lista_papel.size() > 0){
            System.out.println("PAPEL - Tempo de degradação: " +
lista_papel.get(0).getTempoDegradação());
            System.out.println("Nome | quantidade");
            for(int i = 0; i < lista_papel.size(); i++) {
                System.out.print(lista_papel.get(i).getNomeLixo() + " | " +
lista_papel.get(i).getQtd()+"\n");
            }
        }else{
            System.out.println("Não reciclou nenhum PAPEL.");
        }
    }
}

    public void addLista_plastico(){
        Plastico.mostraReciclavel();
        String nome_plastico = JOptionPane.showInputDialog(null, "Digite
qual material de PLÁSTICO você irá reciclar: ");
        int qtd_plastico =
Integer.parseInt(JOptionPane.showInputDialog(null, "Quantos você irá reciclar:
"));
        Plastico pl = new Plastico(nome_plastico, qtd_plastico);
        lista_plastico.add(pl);
        pl.motraSeparacao();
        JOptionPane.showMessageDialog(null, "Reciclagem
concluída!\nVocê pode reciclar quantas vezes quiser.");
    }
}

```

```

public void mostrarLista_plastico(){
    System.out.println("\n-----");
    if(lista_plastico.size() > 0){
        System.out.println("PLÁSTICO - Tempo de degradação:
" + lista_plastico.get(0).getTempoDegradação());
        System.out.println("Nome | quantidade");
        for(int i = 0; i < lista_plastico.size(); i++) {
            System.out.print(lista_plastico.get(i).getNomeLixo() + " | " +
lista_plastico.get(i).getQtd()+"\n");
        }
    }else{
        System.out.println("Não reciclou nenhum PLÁTISCO.");
    }
}

```

```

public void addLista_vidro(){
    Vidro.mostraReciclavel();
    String nome_vidro = JOptionPane.showInputDialog(null, "Digite
qual material de VIDRO você irá reciclar: ");
    int qtd_vidro = Integer.parseInt(JOptionPane.showInputDialog(null,
"Quantos você irá reciclar: "));
    Vidro v = new Vidro(nome_vidro, qtd_vidro);
    lista_vidro.add(v);
    v.motraSeparacao();
    JOptionPane.showMessageDialog(null, "Reciclagem
concluída!\nVocê pode reciclar quantas vezes quiser.");
}

```

```

public void mostrarLista_vidro(){
    System.out.println("\n-----");
    if(lista_vidro.size() > 0){

```

```

        System.out.println("VIDRO - Tempo de degradação: " +
lista_vidro.get(0).getTempoDegradação());
        System.out.println("Nome | quantidade");
        for(int i = 0; i < lista_vidro.size(); i++) {
            System.out.print(lista_vidro.get(i).getNomeLixo() + " | " +
lista_vidro.get(i).getQtd()+"\n");
        }
    }else{
        System.out.println("Não reciclou nenhum VIDRO.");
    }
}
}
}

```

6.2.8 Classe: 'Usuario'

```

package model;

import javax.swing.JOptionPane;

public class Usuario {
    private String nome;
    private int idade;
    private double kilos;
    public Reciclagem rec = new Reciclagem();

    public Usuario(String n, int i, double k) {
        this.setNome(n);
        this.setIdade(i);
        this.setKilos(k);
    }
}

```

```
}
```

```
public String getNome() {  
    return nome;  
}
```

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

```
public double getKilos() {  
    return kilos;  
}
```

```
public void setKilos(double kilos) {  
    this.kilos = kilos;  
}
```

```
public int getIdade() {  
    return idade;  
}
```

```
public void setIdade(int idade) {  
    this.idade = idade;  
}
```

```
public double kg_anuais(){  
    return (365 * this.getKilos());  
  
}
```

```
public double kg_vida(){  
    return kg_anuais() * this.getIdade() ;  
}
```

```

    }

    public double volume(){
        return (kg_vida() * 1000/200)/1000 ; // 1.000 litros = mais ou menos
200 kg
    }

    public void printCalculos(double kg_anuais, double kg_vida, double
volume){
        String msg = "Nome: " + this.getNome() +
            "\nIdade: " + this.getIdade() +
            "\nMédia Kg/Dia: " + this.getKilos() +
            "\n\nIsto significa que você produziu:\n\n• " +
            kg_anuais + "Kg de lixo por ano.\n• " +
            kg_vida + "Kg de Lixo durante toda sua vida.\n• " +
            volume + "m³ de Lixo durante toda sua vida.\n• "+
            "Equivalentes a " + volume + " caixas d'agua de mil litros
sozinho.\n"; // 1 caixa d'agua 1m³ = 1.000 litros
        JOptionPane.showMessageDialog(null, msg);
    }
}

```

6.2.9 Classe: 'Vidro'

```

package model;

import javax.swing.JOptionPane;

public class Vidro extends Lixo {
    final private String TempoDegradação = "+ de 1000 anos";

```

```

public Vidro(String nl, int qt) {
    super(nl, qt);
}

public String getTempoDegradação() {
    return TempoDegradação;
}

public static void mostraReciclavel() {
    JOptionPane.showMessageDialog(null,"RECICLÁVEL:
Embalagens, Copos, Vidros especiais como tampa de forno micro-ondas),\n" +
        "Frascos de remédio vazio, Potes de
conserva, Cacos, Garrafas.\n\n"+

        "NÃO RECICLÁVEL: : Óculos, Lâmpadas,
Espelhos, Louças, Ampolas de remédios, Boxes\n" +
        "Temperados, Pirex, Cerâmicas, Para-brisa
de carros, Porcelanas, Tubos de TV e\n" +
        "monitores.");
}

@Override
public void motraSeparacao() {
    JOptionPane.showMessageDialog(null,"Dicas para separar
VIDROS:\n" +

        "Lave-os bem e retire as tampas e os objetos
indesejados."); }

}

```

7 BIBLIOGRAFIA

7.1 MATERIAL USADO PARA A PESQUISA:

<https://www.em.com.br/app/noticia/especiais/educacao/enem/2016/03/16/noticia-especial-enem,744010/o-risco-de-contaminacao-dos-rios-e-nascentes-com-metais-pesados.shtml>

<https://www.ambiental.sc/saiba-mais/reciclagem/>

<https://www.google.com/amp/s/m.mundoeducacao.uol.com.br/amp/geografia/reciclagem.htm>

<https://www.ecycle.com.br/2046-reciclagem>

<https://blog.eureciclo.com.br>

<https://www.google.com/amp/s/revistagalileu.globo.com/amp/Ciencia/Meio-Ambiente/noticia/2020/02/por-que-o-brasil-ainda-recicla-ao-pouco-e-produz-tanto-lixo.html>

<https://www.em.com.br/app/noticia/especiais/educacao/enem/2016/03/16/noticia-especial-enem,744010/o-risco-de-contaminacao-dos-rios-e-nascentes-com-metais-pesados.shtml>

7.2 MATERIAL USADO PARA A PROGRAMAÇÃO:

<https://pt.stackoverflow.com/questions/35281/verificar-se-a-string-é-nula-ou-vazia>

<https://pt.wikihow.com/Verificar-um-Valor-Nulo-em-Java>

<https://www.devmedia.com.br/java-interface-aprenda-a-usar-corretamente/28798>

<https://qastack.com.br/programming/1481178/how-to-force-garbage-collection-in-java>

<http://old.secovi.com.br/files/Downloads/guia-lixo-reciclavel-webpdf.pdf>

http://www.pmf.sc.gov.br/arquivos/arquivos/pdf/01_12_2009_13.09.02.81cd7ca62da84376489328c705eca969.pdf

8 FICHA DE ATIVIDADES



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Kaio Ruan Felix Martins TURMA: CC3A33 RA: N608480CURSO: Ciências da Computação CAMPUS: Tatuapé SEMESTRE: 3º TURNO: MatutinoCÓDIGO DA ATIVIDADE: 76B9 SEMESTRE: 3º ANO GRADE: 2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS	ASSINATURA DO PROFESSOR
17/11/2020	Confecção do projeto destinado a Atividades Práticas Supervisionadas	75	Kaio Ruan Felix Martins	75	

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: _____

AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO

FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Marcelo Nicolas Duzzi Olivares TURMA: CC3A33 RA: N529FJ9CURSO: Ciências da Computação CAMPUS: Tatuapé SEMESTRE: 3º TURNO: MatutinoCÓDIGO DA ATIVIDADE: 76B9 SEMESTRE: 3º ANO GRADE: 2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS	ASSINATURA DO PROFESSOR
17/11/2020	Confecção do projeto destinado a Atividades Práticas Supervisionadas	75	Marcelo N. D. Olivares	75	

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: _____

AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Matheus Santos Brito de Almeida TURMA: CC3A33 RA: F215JG3

CURSO: Ciências da Computação CAMPUS: Tatuapé SEMESTRE: 3º TURNO: Matutino

CÓDIGO DA ATIVIDADE: 76B9 SEMESTRE: 3º ANO GRADE: 2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS	ASSINATURA DO PROFESSOR
17/11/2020	Confecção do projeto destinado a Atividades Práticas Supervisionadas	75	Matheus S. B. de Almeida	75	

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: _____

AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO



FICHA DAS ATIVIDADES PRÁTICAS SUPERVISIONADAS - APS

NOME: Nicolas Paiuca Buscarini TURMA: CC3A33 RA: N613HB9

CURSO: Ciências da Computação CAMPUS: Tatuapé SEMESTRE: 3º TURNO: Matutino

CÓDIGO DA ATIVIDADE: 76B9 SEMESTRE: 3º ANO GRADE: 2021

DATA DA ATIVIDADE	DESCRIÇÃO DA ATIVIDADE	TOTAL DE HORAS	ASSINATURA DO ALUNO	HORAS ATRIBUÍDAS	ASSINATURA DO PROFESSOR
17/11/2020	Confecção do projeto destinado a Atividades Práticas Supervisionadas	75	Nicolas Buscarini	75	

(1) Horas atribuídas de acordo com o regulamento das Atividades Práticas Supervisionadas do curso.

TOTAL DE HORAS ATRIBUÍDAS: _____

AVALIAÇÃO: _____

Aprovado ou Reprovado

NOTA: _____

DATA: ____/____/____

CARIMBO E ASSINATURA DO COORDENADOR DO CURSO