6. Plano de desenvolvimento do jogo (elementos e ferramentas que serão utilizadas).

**Plano de Desenvolvimento**

O jogo contem 1 pacote com 5 classes e 3 folders com um banco de imagens utilizadas no próprio jogo.

A princípio o jogo tem como objetivo e tema a reciclagem, então, foi desenvolvido um jogo no qual você precisa identificar o lixo e logo após clicar ou pressionar o botão correspondente a lixeira correspondente ao lixo. O jogo possui um menu inicial que você pode tanto quanto prosseguir quanto sair do jogo, logo após caso prossiga, você se depara com a tela de dicas do jogo e seleção de nível, você pode tanto voltar para o menu principal quanto ir para a tela do jogo. Na tela do jogo, você verá as lixeiras desabilitadas esperando que o jogador comece para que sejam habilitadas, quando começado, o jogo contabiliza os pontos, erros, acertos, combos e quantas palavras e/ou lixo foram mostrados na tela, tudo isso durante a jogatina, você tem o objetivo de marcar “x” pontos de acordo com os níveis.

A primeira classe que foi criada foi Jogo.java, porque precisávamos de uma base, e a base do jogo era o próprio jogo em si.

As lixeiras, os botões, a barra de progresso, os contadores de pontuação e os combos foram importados tanto Swing quanto AWT.

Para que as 90 palavras fossem criadas e colocadas em suas respectivas lixeiras, fizemos vetores de 0 à 89 para cada lixo, em um jogo é impossível um lixo se repetir.

Um passo importante para o desenvolvimento foi a interação da classe Jogo com a classe Opcoes. Na linha de código a seguir, estamos recebendo o valor retornado do método getA(), que no caso é o valor escolhido na comboBox cbNivel da classe Opcoes:

**int** nivel = Opcoes.*getA*();

Feito isso, temos o nível escolhido pelo usuário em mãos, sendo 0 para o nível fácil, 1 para o médio e 2 para o difícil. A diferença entre eles é o número de palavras e a pontuação mínima, portanto temos esse trecho do código fonte:

**if** (nivel == 0)

{

npalavras = 60;

titulo = "Fácil";

pontMinima = 230;

}

**else** **if** (nivel == 1)

{

npalavras = 71;

titulo = "Médio";

pontMinima = 280;

}

**else** **if** (nivel == 2)

{

npalavras = 90;

titulo = "Difícil";

pontMinima = 330;

}

A variável titulo mostrada acima é uma variável que irá ser usada para mostrar a dificuldade no titulo do frame, portanto nada que interfira diretamente na forma de jogar.

O método sorteio é essencial para que o jogo funcione de forma correta. Nele é contido a parte do código que faz o sorteio de uma posição no vetor e atribui esse valor sorteado para a variável contPalavras. Usando esse número sorteado, o programa procura no vetor palavras qual é a palavra correspondente a aquela posição no vetor. E ainda depois o programa procura a imagem correspondente a esse número, que está associado a palavra escrita no vetor. Vamos mostrar a linha de código e dar um exemplo:

**public** **void** sorteio(){

cont += 1;

lblContagem.setText(String.*valueOf*(cont));

String palavras [] = {/\*Papel\*/"Jornal", "Sulfite", "Folha de caderno", "Envelope", "Caixa de pizza", "Cartolina", "Papel Cartão", "Revista", "Lista telefônica", "Formulário", "Papel de Fax", "Fotocópia", "Caixa de Chá", "Caixa de Lápis Vazia", "Caixa de remédio",

/\*Plastico\*/"Garrafa Pet", "Sacola de mercado", "Cano PVC", "Tampa de caneta", "Garrafa de óleo", "Copo descartável", "Talher de festa", "Frasco de produto", "Embalagem de shampoo", "Embalagem de detergente", "Galão de água", "CD", "Pratos de festa", "Saco de Lixo", "Bolsa de soro medicinal",

/\*Vidro\*/"Embalagem de Azeitona", "Frasco de remédio","Copo", "Caco de vidro", "Tampa de forno", "Garrafa de cerveja", "Garrafa de Refrigerante", "Frasco de perfume", "Embalagem de molho", "Prato", "Tampa de microondas", "Garrafa de suco", "Vidro de janela", "Para-brisas", "Tigela de vidro",

/\*Metal\*/"Lata de refrigerante", "Panela sem cabo", "Cano", "Tampinha de garrafa", "Arame", "Cobre", "Prego", "Embalagem de Marmitex", "Papel Alumínio Limpo", "Ferragem", "Lata de sardinha", "Tampa de iogurte", "Embalagem de café", "Lata de Creme de leite", "Lata de leite condensado",

/\*Organico\*/"Folhas", "Sementes", "Ossos", "Restos de carne", "Palito de dente", "Borra de café", "Chiclete", "Resto de bebidas", "Papel Molhado", "Fezes", "Urina",

/\*Não Reciclavel\*/"Papel Higiênico", "Espelho", "Clipes", "Interruptor", "Cinzeiro","Guardanapo", "Grampo", "Lata de verniz", "Tubo de TV", "Lente de óculos", "Adesivo", "Etiqueta", "Papel Carbono", "Papel Celofane", "Telefone", "Caneca", "Fotografia", "Bituca de cigarro", "Bandeja de Plastico"};// O vetor que recebe as palavras mostradas para o usuário.

contPalavras =(**int**) (Math.*random*() \* npalavras);//Sorteia um número inteiro, este número será usado como índice do vetor palavras[]

**if** (cont == npalavras)

{

i = 0;

}

**if**(!lista.contains(contPalavras))

{

lista.add(contPalavras);

}

**else** **if** (lista.contains(contPalavras))

{

**while**(lista.contains(contPalavras))

{

**if**(cont < npalavras)

contPalavras =(**int**) (Math.*random*() \* npalavras);

**else**

**break**;

}

lista.add(contPalavras);

}

**if**(cont < npalavras)

{

lblPalavra.setText(palavras[contPalavras]); // A label lblPalavra recebe o valor do vetor de índice randomico.

lblPalavra.setIcon(**new** ImageIcon(Jogo.**class**.getResource("/PCT\_APS/Imagens/" + contPalavras + ".png")));

}

}

Este é o método sorteio. Vamos supor que o valor sorteado seja o valor 18. O programa vai buscar no vetor o índice de número 18, e mostra essa palavra pro usuário, que no caso seria “Tampa de caneta”. Depois disso, o programa busca na pasta Imagens localizada dentro do pacote, a imagem correspondente a essa palavra. A imagem vai estar nomeada com o número da posição dela no vetor e a extensão .png. Nesse caso, a imagem da tampa de caneta está nomeada como 18.png.

Fora isso vemos a utilização de listas dentro desse método, que serve para verificar se aquela posição no vetor já foi sorteada. Se foi, é feito um outro sorteio até que o programa sorteie um número que não está na lista, ou seja, que nunca foi utilizado.

Além disso temos um contador para verificar o número de vezes que esse método foi executado. Cada vez que ele é executado acrescenta 1. Isso serve para verificar quando que o jogo deve parar de sortear palavras, no caso, até a variável cont for igual a variável npalavras (variável que representa o número de palavras dependendo do nível escolhido). Quando isso acontecer, a variável i recebe o valor de 0. A variável i é a variável que controla a barra de progresso, e ela chegando a 0 o jogo chega ao fim.

A variável contPalavra faz com que todos os vetores fiquem aleatórios para que haja a dinâmica do jogo. Cada botão foi configurado um “IF” e “ELSE” para receber os lixos correspondentes, porém, como vetores. Exemplo:

**if**((contPalavras >= 0) && (contPalavras <= 14))//Se o vetor estiver entre as posições informadas, executa o código abaixo entre as chaves:

{

acerto();

}

**else**//caso o índice do vetor não esteja entre as posições informadas acima, executa o código abaixo entre as chaves:

{

erro();

}

Os métodos acerto e erro foram criados para evitar uma repetição de código desnecessária, segue o código dos métodos:

**public** **void** acerto(){

i += 10; //adiciona 10 à variavel i, ou seja, o usuário ganha mais tempo

sorteio();//chama o método sorteio para chamar outro índice do vetor

pbVelocidade -= 2;//aumenta a velocidade da barra de progresso

lblResposta.setText("");

acerto += 1;

combo += 1;

**if** (combo > 3)

{

*pontuacao* += 7;//acrescenta 7 pontos para a pontuação do usuário

}

**else**

{

*pontuacao* += 5;//acrescenta 5 pontos para a pontuação do usuário

}

lblCerto.setText(String.*valueOf*(acerto));

contCombo.setText(String.*valueOf*(combo));

lblResposta.setIcon(**new** ImageIcon(Jogo.**class**.getResource("/PCT\_APS/Logos/acertou.png")));

**if**(combo >= 3)

{

lblCombo.setVisible(**true**);

lblCombo.setIcon(**new** ImageIcon(Jogo.**class**.getResource("/PCT\_APS/Logos/combo.png")));

contCombo.setVisible(**true**);

}

}

**public** **void** erro(){

i -= 10;//decrementa 10 aà variavel i, ou seja, o usuário perde tempo

sorteio();//chama o método sorteio para chamar outro índice do vetor

*pontuacao* -= 2;//decrementa 2 pontos para a pontuação do usuário

lblResposta.setText("");

erro += 1;

combo = 0;

contCombo.setText("0");

contCombo.setVisible(**false**);

lblErrado.setText(String.*valueOf*(erro));

lblResposta.setIcon(**new** ImageIcon(Jogo.**class**.getResource("/PCT\_APS/Logos/errou.png")));

lblCombo.setVisible(**false**);

}

Percebe-se que no método acerto ele verifica se o usuário está dentro de um combo, se estiver acrescenta 7 para a pontuação, se não ele acrescenta 5. Fora isso ele chama novamente o método sorteio, para realizar o sorteio de uma nova palavra e imagem, e ainda aumenta um pouco a barra de rolagem através da linha i+=10;. Um segredo que não é mostrado para o usuário é que, dentro desse método acerto existe uma linha de código que é pbVelocidade -= 2;, isso significa que a cada acerto do usuário, a barra de progresso se decrementa cada vez mais rápido.

No método erro é o contrário do método acerto, ele tira 2 pontos variável pontuação, diminui mais um pouco a barra de progresso e, da mesma forma que o método acerto, também chama o método sorteio para realizar um sorteio de uma nova palavra.

Notar que a label lblResposta é quem recebe a imagem em forma de ícone da informação que aparece no canto do jogo, se o usuário acertou ou se errou.

As lixeiras são JButton’s no qual foram colocados ícones das lixeiras respectivas, a principio, os as lixeiras só obtinham o clique do mouse, mas para ficar mais dinâmico foi configurado botões no teclado para cada lixeira, os botões são: Q = Papel, W = Plástico, E = Vidro, R = Metal, T = Orgânico e Y = Não Reciclável. Para que conseguíssemos isso, foi preciso usar KeyStroke, KeyEvent, InputMap e ActionMap. Segue a lógica:

KeyStroke keyStroke = KeyStroke.*getKeyStroke*(KeyEvent.*VK\_TECLA CORRESPONDENTE NO TECLADO*, 0);

String actionName = "TECLA\_TECLA CORRESPONDENTE NO TECLADO";

InputMap inputMap = btnNOME DA LIXEIRA CORRESPONDENTE.getInputMap(JComponent.*WHEN\_IN\_FOCUSED\_WINDOW*);

inputMap.put(keyStroke, actionName);

ActionMap actionMap = btnLIXEIRA CORRESPONDENTE.getActionMap();

actionMap.put(actionName, nemonicoNOME DA LIXEIRA CORRESPONDENTE);

Foi criado o objeto keyStroke da Classe KeyStroke, nele é mapeado o botão correspondente no teclado. InputMap direciona o botão que deve ser utilizado. ActionMap faz a ligação do botão com a tecla do teclado. Deixando assim, o jogo mais dimamico.

E quando a variável i chega a 0, o programa verifica se você fez os pontos necessários e se você conseguiu responder a todas as palavras. Caso sim, a seguinte linha de comando é chamada.

**if** ((cont == npalavras)&&(*pontuacao* >= pontMinima))

{

**try** {

**new** Vencedor().setVisible(**true**);

} **catch** (IOException e) {

e.printStackTrace(); }

dispose();

}

Basicamente, ele está chamando a classe Vencedor que é a classe que apresenta uma mensagem parabenizando ao jogador, e logo em seguida fecha o frame do jogo.

Porém caso você não tenha conseguido os pontos necessários ou não tenha conseguido responder a todas as palavras, o programa irá verificar o motivo da sua derrota e irá lhe informar.

{

**if**((cont < npalavras)&&(*pontuacao* >= pontMinima)) {

fim = "Você fez os pontos necessários mas o tempo acabou!\n\nVocê perdeu!";

}

**else** **if**((cont == npalavras)&&(*pontuacao* < pontMinima))

{

fim = "Você respondeu todas as palavras dentro do tempo mas não fez os pontos necessários\n\nVocê perdeu!";

}

**else** **if**((cont < npalavras)&&(*pontuacao* < pontMinima))

{

fim = "O tempo acabou e você não fez os pontos necessários.\n\nVocê perdeu!";

}

**int** resposta = JOptionPane.*showConfirmDialog*(**null**, fim + "\n\nPontuação: " + *pontuacao* + " pontos\n\nDeseja jogar novamente?", "Fim de jogo!", JOptionPane.*YES\_NO\_OPTION*);//mostra uma caixa de mensagem no final, mostrando a pontuação final

**if**(resposta == JOptionPane.*YES\_OPTION*)

{

**try** {

Jogo frame = **new** Jogo();

frame.setVisible(**true**);

} **catch** (Exception e) {

e.printStackTrace();

}

dispose();

}

Isso encerra o jogo da seguinte maneira, em caso de derrota, o jogo vai mostrar o motivo e sua pontuação através de uma caixa de mensagem, e nessa mesma caixa lhe oferecendo a jogar novamente. O motivo da derrota é controlado através da variável fim, do tipo String.

7. Projeto (estrutura e módulos que serão desenvolvidos) do programa

**Projeto**

No começo, fizemos uma reunião para decidir o que fazer, qual jogo. Dentre várias opiniões, decidimos fazer um jogo que fosse dinâmico, no qual o jogador pudesse interagir com o jogo.

Foi feito vários desenhos em caderno para chega em uma conclusão de como seria a interface gráfica até que todos chegaram em um consenso e foi decido.

O jogo tem como base a reciclagem de lixo, que consiste em ver o lixo que aparece na tela e jogar na lixeira correspondente.

A classe Jogo.java foi a primeira classe a ser criada, nela foi criado a interface gráfica que planejávamos, com 6 lixeiras, com barra de tempo e botão para iniciar o jogo.

Dividimos as tarefas entre os 4 integrantes, sempre sincronizando os projetos.

Assim que foi criado a primeira parte, o jogo mostrava o nome do lixo e tínhamos que clicar no botão (que era a lixeira) para que pudéssemos ter um acerto ou um erro. O acerto e erro impactava no tempo.

Cada vez que o projeto era aberto e a cada reunião, vinham propostas de todos nós para que fossem feitas melhorias.

Colocamos em uma segunda etapa o contador de pontos e palavras, assim, o jogador sabia quantos pontos ele ia ganhando e quantas palavras já foram, até que decidimos criar o Menu inicial ou tela de inicio.

A criação da tela de inicio foi um tanto quanto trabalhoso, porque necessitava de uma arte para ser colocada e decidir um nome para o jogo. Foram várias sugestões para o nome do jogo, mas nenhuma parecia se encaixar no nosso projeto, até que um nome foi decidido: Reciclando com Pendoleo.

A nome Reciclando com Pendoleo foi decidido pelo grupo e faz alusão ao nome do Pen Drive do integrante Leonardo Pereira. Com o nome decidido, partimos para os desenhos de como deveria ser a tela inicial do jogo.

A idéia é que mostrasse o objetivo do jogo logo na tela inicial, que era reciclagem. Decidimos então colocar o símbolo de reciclagem junto ao nome do jogo. Como o eclipse não ofereceu recursos necessários para a criação da arte e do nome do jogo como queríamos, foi necessário o uso da ferramenta Photoshop.

No Photoshop foi criado o símbolo de reciclagem em coloração verde, para lembrar o meio ambiente, e o nome do jogo. No nome do jogo foi acrescentado no lugar do “L” um pendrive em forma de “L” para que ficasse claro a intenção do nome do Pen Drive.

Com as artes criadas, criamos a classe Menu.java. No Menu.java, colocamos o botão para iniciar o jogo, que no caso te transferia para tela do jogo e o botão “Sair”.

No meio da tela foi a hora de colocar a arte criada anteriormente, o logo de reciclagem verde por traz e o nome do jogo na frente dele. O nome do jogo foi colocado em cor azul para representar o Pen Drive que era também azul. Para ficar mais claro ainda, colocamos no canto superior esquerdo o desenho do Pen Drive que é o “Pendoleo”.

Decidimos também que o fundo da tela seria verde água, também para lembrar o meio ambiente.

Parecia que faltava algo quando transitava-se entre a tela de Menu inicial e a tela do jogo. Foi quando decidimos criar uma tela que intermediasse, a tela Opções.

Na tela de opções, havíamos pensado em colocar somente a seleção de níveis, que são: Fácil, Médio e Difícil e foi o que aconteceu. Porém, ainda não tínhamos pensado como que seria a diferença entre os níveis.

O jogo estava particularmente enjoativo, você obtinha pontos e tempo adicional para cada acerto e não tinha fim, fazendo você fechar o jogo quando cansasse. Juntando as duas situações resolvemos conversar sobre a implementação dos níveis.

Estávamos em duvida se os níveis implicariam em mais palavras, quanto mais difícil mais palavras, ou se quanto mais difíceis, mais lixeiras e meta de pontos.

Se colocássemos mais palavras de acordo com a dificuldade, havia a probabilidade o jogo ficar muito fácil nos níveis mais fáceis e talvez até enjoativo novamente no nível difícil. O plano B de colocar mais lixeiras quanto mais difíceis foi a escolhida, porque se você tem uma meta de ponto com uma quantidade X de lixos você tem uma motivação de tentar bater a meta em menor tempo possível.

Então foi decidido que:

No nível fácil o jogo viria com 4 lixeiras, para bater a meta você precisa de no mínimo 230 pontos e tem um limite de 60 palavras para tentar atingir a meta.

No nível médio o jogo viria com 5 lixeiras, para bater a meta você precisa de no mínimo 280 pontos e tem um limite de 71 palavras para tentar atingir a meta.

No ultimo nível, o difícil, o jogo viria com todas as 6 lixeiras, para bater a meta você precisa de no mínimo 330 pontos e tem um limite de 90 lixos para tentar atingir a meta.

Agora o jogador poderia começar com o básico e ir melhorando e dificultando o jogo gradativamente com o seu aprendizado.

Depois de criada as três classes principais, o jogo parecia estar pronto, mas parecia que o grupo sempre queria melhorar o jogo em todos os sentidos para que pudesse ser um jogo impecável na medida do possível.

Até que em mais uma reunião em grupo, foi decido colocar na tela de opções, as dicas do jogo. As dicas do jogo foi implementada para que o jogador possa conhecer toda a dinâmica do jogo e ser guiado para não estranhar o jogo durante o mesmo. Nela tem o objetivo do jogo, as regras e um explicativo para os níveis que correspondia ao botão “Sobre”.

O botão “Sobre” trazia o resultado de uma nova classe criada, a classe Pontuação.Java.

Em Pontuação. Java, colocamos as informações dos níveis, como foi descrito anteriormente.

O jogo parecia completo, mas se o jogador conseguisse atingir a meta, o que ele ganharia?

Criamos uma nova classe chamada Vencedor.Java que era responsável por retornar para o jogador assim que ele batesse a meta uma mensagem: “Parabéns! Você ganhou! Você fez “X”(depende do nível o número necessário de pontos) pontos e um boneco segurando um troféu, no final, em baixo, há uma mensagem perguntando se deseja jogar novamente, com um 2 botões, de sim e de não.

Caso jogador não bata a meta, aparece uma mensagem na tela:

O tempo acabou e você não fez os pontos necessários.

Você perdeu!

Pontuação “X”

E também há uma mensagem perguntando se deseja jogar novamente, com 2 botões, de sim e de não.

A ultima coisa adicionada no jogo foi o contador de combo, decidimos que se o jogador adquire uma experiência e acerta vários lixos em suas lixeiras seguidamente, ele merece um aditivo na pontuação, então se o jogador começa a fazer combo (que é acertar varias seguidas) ele possui um adicional no ponto e na tela mostra o quanto de combo ele já está. Caso o jogado erre, o contador de combo é zerado e caso ele volte a acertar e fazer combo o contador começa a ser contado a partir do 1, porém o combo só é mostrado na tela a partir do 3. Toda vez também que o jogador acerta aparece na tela uma palavra dizendo: Acertou, e caso o jogador erre aparece também na tela a mensagem dizendo: errou.

Depois do jogo testado varias vezes e percebendo que não havia mais nada que pudéssemos adicionar, resolvemos finalizar o projeto que se tornou o nosso mais novo jogo chamado “Reciclando com Pendoleo”.