-**Principal (transmissor)**
Inicialização dos Objetos:
Primeiro, criamos um objeto Model e inicializamos uma lista chamada listQuad para armazenar as teclas do teclado virtual. Também definimos alguns objetos auxiliares como Quad e Retangulo para representar as teclas, o botão de espaço e o botão de backspace. Além disso, criamos um campo de texto chamado inputText.

Configuração Inicial:
No método setup, definimos o tamanho da janela e configuramos os sons do modelo. Em seguida, inicializamos a lista listQuad com as teclas, posicionando-as em uma grade.

Desenho dos Elementos:
No método draw, desenhamos o fundo, o campo de texto, todas as teclas e o botão de espaço. Cada tecla é exibida e sua cor é atualizada.

Interação com o Mouse:
O método mousePressed lida com os cliques do mouse. Quando uma tecla é clicada, o texto correspondente é adicionado ao campo de texto. Também permite enviar o texto ou adicionar espaços e backspaces. Ele pega o índice do botão clicado na lista e coloca na fila, que equivale à lista de letras no alfabeto, e o clickbutton muda a cor quando clicado para proporcionar o feedback adequado. A mesma lógica serve para quando se digita no teclado, com a diferença de que se usa o keyPressed, que faz a verificação se algo foi apertado no teclado.

Lógica do Backspace:
Para criar esse botão, utiliza-se a função substring para extrair uma parte de uma string e o length-1 para apagar o último elemento. Dessa forma, sempre que o botão for clicado, ele apagará a última letra digitada.

Retângulo (transmissor)
A classe Retangulo representa um botão retangular no teclado virtual, sendo responsável por sua inicialização, exibição e interação. Cada instância da classe possui atributos como as coordenadas x e y do canto superior esquerdo do retângulo, largura e altura que determinam o tamanho do retângulo, e text que representa o texto exibido dentro do botão. Além disso, há duas cores definidas: colorClick, que é a cor do botão quando clicado, e colorButton, que mantém a cor atual do botão. O atributo lastClickTime armazena o tempo (em milissegundos) do último clique, permitindo o controle da mudança de cor.

Para inicializar um botão retangular, existem dois construtores: um padrão que define as cores padrão, e outro que permite definir a cor inicial do botão. A criação e exibição do retângulo na tela são gerenciadas pelos métodos createRect(float x, float y, float largura, float altura, String text) e createRect(float x, float y, float largura, float altura). O primeiro método desenha o retângulo na posição especificada e exibe o texto no centro, enquanto o segundo método desenha apenas o retângulo sem texto.

A classe também possui métodos getX(), getY(), getLargura() e getAltura() que retornam as coordenadas, largura e altura do retângulo, respectivamente.

A interação com o botão é gerenciada pelo método clickButton(), que muda a cor do botão para colorClick e registra o tempo do clique. Para garantir que o botão retorne à sua cor original após um breve período, o método updateColor() é utilizado, restaurando a cor padrão se mais de 300 milissegundos se passaram desde o último clique.

Assim, a classe Retangulo encapsula a funcionalidade de um botão retangular do teclado virtual, permitindo sua inicialização, exibição e interação de maneira eficiente e visualmente responsiva.

Em resumo:
A classe Retangulo define a estrutura e o comportamento de um botão retangular no teclado virtual. Ela permite inicializar o botão com suas coordenadas, tamanho e texto. O botão pode ser exibido na tela, mudar de cor ao ser clicado e retornar à sua cor original após um curto período.

Quad (transmissor)
A classe Quad representa uma tecla do teclado virtual, sendo responsável por sua inicialização, exibição e interação. Cada instância da classe possui atributos como as coordenadas x e y do canto superior esquerdo do quadrado, o size, que determina o tamanho da tecla, e a letter, que representa o símbolo ou caractere exibido. Além disso, há duas cores definidas: colorClick, que é a cor da tecla quando clicada, e colorButton, que mantém a cor atual da tecla. O atributo lastClickTime armazena o tempo (em milissegundos) do último clique, permitindo o controle da mudança de cor.

Para inicializar uma tecla, o construtor Quad(float x, float y, float size, String letter) é utilizado, configurando as propriedades essenciais e definindo a cor padrão do botão. A classe também inclui um método para exibir a tecla na tela, chamado displayQuad(Quad tecla), que desenha o quadrado na posição especificada e mostra a letra no centro.

A interação com a tecla é gerenciada pelo método isMouseOver(Quad tecla), que verifica se o cursor do mouse está sobre a tecla e retorna true ou false, conforme a posição do cursor. Quando a tecla é clicada, o método clickButton() é chamado, mudando a cor do botão para colorClick e registrando o tempo do clique. Para garantir que a tecla retorne à sua cor original após um breve período, o método updateColor() é utilizado, restaurando a cor padrão se mais de 300 milissegundos se passaram desde o último clique.

Assim, a classe Quad encapsula a funcionalidade de uma tecla do teclado virtual, permitindo sua inicialização, exibição e interação de maneira eficiente e visualmente responsiva.

Em resumo:
A classe Quad define a estrutura e o comportamento de uma tecla do teclado virtual. Ela permite inicializar a tecla com suas coordenadas, tamanho e letra. A tecla pode ser exibida na tela, detectar cliques do mouse, mudar de cor ao ser clicada e retornar à sua cor original após um curto período.

TextField (transmissor)
A classe TextField representa um campo de texto no teclado virtual, sendo responsável pela exibição do texto digitado e pelo fundo do campo de texto. Cada instância da classe possui atributos como text, que armazena o texto atual, x e y, que representam as coordenadas do canto superior esquerdo do campo de texto, e textColor, que define a cor do texto. Além disso, a classe utiliza um objeto Retângulo para representar o fundo do campo de texto com uma cor semi-transparente.

Para inicializar um campo de texto, o construtor TextField(float x, float y, String text) é utilizado, configurando as propriedades essenciais como o texto, a cor do texto e as coordenadas x e y.

A classe também inclui métodos para obter e definir o texto (getText() e setText(String text)), e para definir a cor do texto (setColor(color textColor)).

A exibição do texto na tela é gerenciada pelo método displayText(), que define a cor e o tamanho do texto, e o desenha na posição especificada. O fundo do campo de texto é desenhado pelo método backgroundDisplay(), que utiliza o objeto backgroundText para criar um retângulo de fundo.

Em resumo:
A classe TextField define a estrutura e o comportamento de um campo de texto no teclado virtual. Ela permite inicializar o campo com o texto, coordenadas e cor. O campo de texto pode exibir o texto na tela e mostrar um fundo semi-transparente.

Fila (transmissor)
A classe Fila implementa uma estrutura de dados em fila utilizando um ArrayList<Integer>, que armazena uma sequência ordenada de números inteiros. Essa classe oferece métodos para adicionar e remover elementos da fila, além de um método especial para processar todos os elementos da fila com um comportamento definido.

O método adicionar(int valor) insere um novo valor inteiro ao final da fila, enquanto o método remover(int valor) remove a primeira ocorrência do valor especificado da fila. O método submit() tem um funcionamento mais complexo: ele adiciona o valor 28 no início da fila e o valor 29 no final, depois percorre a fila tocando cada valor em sequência usando um método tocar(int) de um objeto model. Se o valor estiver na primeira posição, ele é tocado imediatamente; para os demais valores, há um atraso de 500 milissegundos entre cada toque. Após processar todos os elementos, a fila é esvaziada. O método getSize() retorna o número de elementos atualmente na fila.

Esta implementação é útil para gerenciar e processar uma lista de valores inteiros de forma ordenada, garantindo que cada valor seja tratado em sequência e permitindo a manipulação da fila com facilidade

Model (transmissor)
A classe Model é projetada para gerenciar sons e componentes gráficos, especificamente sons associados a letras e algumas operações especiais.

Ela possui vários atributos, incluindo referências a arquivos de som (SoundFile) para cada letra do alfabeto, além de sons especiais como o início e o fim da leitura. Esses arquivos de som são armazenados em variáveis como letterA, letterB, e assim por diante, até letterZ, além de readStart, readEnd, letterSpace e letterEnd. Todos esses sons são agrupados em um array chamado listSounds para facilitar o acesso.

O construtor Model(PApplet p) inicializa a referência ao objeto pai, que é uma instância da classe PApplet do Processing, utilizada para criar a interface gráfica e sons.

O método setupSounds() é responsável por carregar os arquivos de som específicos para cada letra e sons especiais. Esses sons são carregados a partir de arquivos WAV e associados às variáveis correspondentes. Em caso de erro durante o carregamento, a exceção é capturada e a pilha de chamadas é impressa para depuração.

O método tocar(int indice) toca o som correspondente ao índice fornecido no array listSounds. Se o índice não for 28 ou 29 (sons especiais), o método insere um atraso de 500 milissegundos e toca o som de letterEnd para indicar o fim da letra. Cada vez que um som é tocado, o índice é impresso no console.

Além do gerenciamento de sons, a classe Model também define a aparência e a posição de um retângulo na interface gráfica, utilizando os atributos rectX, rectY e rectSize. O método reta(String text) desenha um retângulo na posição especificada e insere um texto centralizado dentro dele. Há também métodos para alterar a cor do retângulo (setRectColor e updateRectColor) e métodos para definir e obter as coordenadas e o tamanho do retângulo (setRectX, setRectY, getRectX, getRectY, getRectSize).

Em resumo, a classe Model gerencia uma coleção de sons associados a letras, permite tocar esses sons e gerencia componentes gráficos simples para exibição e interação na interface.

Principal (receptor)
Em resumo, este programa captura o áudio do microfone, realiza a análise FFT para obter o espectro de frequência e exibe essas informações na tela. Ele inclui campos de texto para exibir mensagens e frequências, além de um botão "Limpar" que apaga o texto exibido quando clicado. As classes auxiliares AudioProcess, TextField, ButtonR e Retangulo gerenciam a funcionalidade e a exibição dos elementos da interface gráfica.

Audio (receptor)
Funcionamento Principal
Análise de Frequência (freqCalc):

O método freqCalc percorre o espectro de frequências capturadas pelo FFT (Transformada Rápida de Fourier) para determinar a frequência com a maior amplitude (maxAmplitude). Isso é crucial para identificar o pico de frequência mais significativo no áudio capturado.
Identificação da Frequência Predominante (mainFreq):

O método mainFreq realiza uma média das frequências predominantes durante um período de contagem. Ele inicia uma contagem (startWait = millis()) e continua calculando as frequências predominantes enquanto o tempo decorrido for inferior a 2000 milissegundos. Isso permite capturar a frequência principal que está sendo emitida no áudio.
Verificação de Início de Transmissão (checkFreqInit):

O método checkFreqInit monitora se houve o início de uma transmissão. Se readTransmission for falso e a frequência detectada estiver entre 4900 Hz e 5100 Hz, assume-se que uma transmissão começou. Isso é indicado alterando a cor do campo de texto textFieldMsg para verde (setColor(color(0, 255, 0))).
Quando readTransmission é verdadeiro e a frequência detectada está entre 5900 Hz e 6100 Hz, a transmissão é considerada encerrada. A cor do campo de texto é alterada para vermelho (setColor(color(255))).
Leitura da Transmissão (readTransmission):

Durante uma transmissão ativa (readTransmission é verdadeiro), o método readTransmission verifica se a frequência capturada está dentro dos intervalos específicos definidos em lists.getValue(i) e lists.getValue(i) + 99.
Se a frequência corresponder a um dos intervalos, a letra correspondente é adicionada ao texto atual do campo textFieldMsg. Isso permite decodificar a mensagem transmitida com base nas frequências capturadas.
Funcionalidades Adicionais
Exibição de Frequência Debug (displayDebugFreq):
Este método atualiza o campo de texto debugFreq com a frequência principal detectada durante a execução do programa. Isso é útil para monitorar visualmente a frequência predominante em tempo real.
Resumo
A classe AudioProcess encapsula a lógica essencial para capturar, processar e interpretar o áudio em tempo real. Ao analisar as frequências predominantes e monitorar mudanças específicas no espectro de áudio, ela facilita a detecção e a interpretação de transmissões de dados com base nas frequências transmitidas.

Button (receptor)
A classe ButtonR em Java é responsável por criar e exibir botões retangulares na tela. Ela possui atributos como coordenadas (x, y), largura (width), altura (height) e um texto (text) que é exibido dentro do botão. Além disso, há um atributo para definir a cor do botão (buttonColor).

O método buttonCreate inicializa os atributos do botão com as coordenadas, dimensões e texto especificados, e então chama o método displayButton para desenhar o botão na tela.

O método displayButton desenha o retângulo na posição determinada por x e y, com a largura e altura especificadas. Ele preenche o botão com a cor definida em buttonColor e exibe o texto no centro do botão.

Essa classe é útil para criar botões interativos em interfaces gráficas, proporcionando uma maneira simples e eficaz de interação do usuário com o programa.

Lista (receptor)
A classe ListValues em Java contém duas listas: uma alphabet que armazena as letras do alfabeto e um espaço em branco, e outra testValues que contém valores inteiros associados aos intervalos de frequência correspondentes a cada letra e ao espaço em branco.

Essas listas permitem associar diretamente cada letra ou espaço em branco a intervalos específicos de frequência, facilitando a interpretação e decodificação de mensagens transmitidas através de sinais sonoros em aplicações de processamento de áudio.

TextField Receptor
A classe TextField em Java é usada para representar um campo de texto em aplicações gráficas. Ela mantém um texto atual, definido inicialmente pelo construtor com uma posição específica (x, y) na tela. A cor do texto pode ser alterada usando o método setColor. Métodos como getText permitem acessar o texto atual, enquanto setText define um novo texto. O método displayText exibe o texto na tela na posição definida, com a cor especificada. cleanText redefine o texto para "MENSAGEM: ", atualizando imediatamente sua exibição na tela.