Documentação de desenvolvimento do jogo Genius

Integrantes: Alana Santos de Oliveira, Caian de Jesus Santana,
Matheus de Lima Oliveira

Sumário

Pacote Audio (sfx)

Esse pacote contém todos os sons dos botões que foram utilizados no jogo.

Pacote Enums

Esse pacote contém um conjunto fixo de cores para serem utilizado de forma mais legível

Cor: Criada para definir uma enumeração das cores do jogo e deixar o código mais legível.

Pacote imagens

Esse pacote contém todas as imagens que foram criadas para a construção da tela de fundo do jogo e dos botões.

Pacote Negócio

Esse pacote contém um conjunto fixo de cores para serem utilizado de forma mais legível

CLASSE DATA:

Classe para pegar a data atual, sobrescrever o método toString retorna uma string no formato de data DD/MM/YYYY. Implementa serializable

CONSTRUTOR:

public Data(): Construtor que recebe a data atual.

TOSTRING():

public String toString(): Retorna a data formatada em DD/MM/YYYY.

CLASSES GENIUS:

Genius:
Classe abstrata, que define a lógica do jogo e seus atributos, implementa serializable.

CONSTRUTORES:

protected Genius(Data data, String titulodoCampeonato, int ritmo, int dificuldade, List<Jogador> jogadores)
construtor que inicializa o jogo mantendo as informações principais do jogo já instanciado, utilizado para manter atributos entre as dificuldades do jogo

protected Genius()
Construtor utilizado para inicializar o jogo a primeira vez

MÉTODOS GET:

public Data getData(): retorna a data de criação do campeonato.

public Jogador getJogadorAtual(): retorna o jogador atual

public List<Jogador> getListaJogadores(): retorna uma cópia da lista de jogadores

public List<Jogador> getVencedores(): retorna uma lista de jogadores ordenados por pontos

public List<Integer> getSequencia(): retorna a sequência de cores gerada

public String getTitulodoCampeonato() retorna o título do campeonato

MÉTODOS SET:

public void setTitulo(String tituloNovo) throws Exception
Método que define um título para o jogo e lança uma exceção se tiver menos que 2 letras

public void setRitmo(): método que define o ritmo, não recebe nenhum parâmetro, pois o jogador conta apenas com 1 botão para alterar a dificuldade, assim ele aumenta até o ritmo mais alto, caso esteja no máximo retorna ao inicial, define também o tempo de reação do jogador que varia de acordo com o ritmo.

private void setTempodeReacao(int ritmo): define o tempo que o jogador tem para reagir de acordo com o ritmo recebido

protected void setDificuldade(int dificuldade): altera o valor da dificuldade.

OUTROS MÉTODOS:

private void alteraJogadorAtual(): altera o jogador atual e chama métodos para verificar se foi a maior pontuação, caso tenha mais jogadores finalizada a rodada e gera uma nova sequência para o próximo jogador, se não finaliza o jogo.

public boolean ehUltimaJogada(): informa se é a última jogada do jogador atual.

public void adicionaJogador: adiciona um jogador a lista de jogadores.

private void pontua(): dá pontos ao jogador atual.

public boolean analisaJogada(Long instantedaExibicao, Cor jogada) throws Exception public boolean analisaJogada(Long instantedaExibicao, Cor jogada) throws Exception: analisa a jogada feita, recebe a cor selecionada e o instante que a interface terminou sua exibição. lança exceções caso a lista de jogadores esteja vazia, ou o campeonato esteja sem título, caso o jogo tenha sido finalizado retorna falso. chama métodos de análise de jogada como o reagiu em tempo e acertou a sequência.

private boolean reagiuEmTempo(Long instantedaExibicao): recebe o instante que a interface finalizou a exibição. caso seja a primeira jogada após a exibição compara a soma do instantedaExibicao mais o tempo para reagir com o instante atual, se nao for é utilizado o tempo da última jogada.

private boolean acertouaSequencia(Cor cor) recebe uma cor escolhida, compara com a cor da lista na posição da jogada atual. O jogo mantém o índice da jogada que o jogador deve fazer. Dá pontos ao jogador caso não seja modo de treino, aumenta o índice da jogada atual caso o jogador tenha acertado e se for a última jogada reinicia o índice e invalida o instante armazenado e adiciona mais cores a sequência. Caso o jogador erre retorna falso.

private void validaInstante(): diz que o instante da última reação do jogador pode ser utilizado como comparação.

private void invalidaInstante(): diz que o instante da última reação do jogador não pode ser utilizado como comparação.

protected void geraSequencia(): gera a sequência de cores de forma aleatória

protected void adicionanaSequencia(): adiciona números a sequência varia com a dificuldade.

public boolean jogofoiEncerrado(): diz que o jogo foi encerrado.

private boolean ehAmaiorPontuacao(): caso seja armazena a maior pontuacao.

public boolean temEmpate(): diz se tem empate

public abstract void ativaDesativaTreino() throws Exception: ativa ou desativa o modo de treino, lança exceção para evitar que seja acionado em um jogada em andamento.

public boolean ehmododeTreino(): diz se é modo de treino

public void inciaRodada(): informa que a jogada foi iniciada

private void finalizaRodada(): informa que a rodada foi finalizada

public boolean jogoEstaAtivo(): diz se a rodada foi iniciada

public boolean jogofoiEncerrado(): diz se o jogo foi finalizado

public abstract Genius mudaDificuldade(): retorna um genius com a dificuldade alterada

public abstract Genius getRodadadeDesempate() throws Exception retorna um campeonato de desempate

GeniusBase:
Classe que define o jogo no modo normal e herda de Genius.

CONSTRUTORES:

public GeniusBase()
Construtor para inicializar o jogo em sua dificuldade base

protected GeniusBase(Data data, String titulodoCampeonato, int ritmo, int dificuldade, List<Jogador> jogadores)
Construtor que inicia o modo com os dados salvos

OUTROS MÉTODOS:

public Genius mudaDificuldade()
Método que muda a dificuldade para médio

GeniusMedio:
Classe que serve para colocar o jogo no modo médio e herda de Genius

protected void adicionanaSequencia()

Método que adiciona cores na sequencia

protected void geraSequencia()
Método que gera sequência de 3 números

GeniusDificil:
Classe que serve para colocar o jogo no modo difícil e herda de Genius

protected void adicionanaSequencia()
Método que adiciona cores na sequencia

protected void geraSequencia()
Método que gera sequência de 3 números

CLASSE JOGADOR:
Jogador: Classe que define a lógica do jogador, seus atributos e métodos. Implementa serializable e Comparable

CONSTRUTOR:

public Jogador(String nome, String apelido) throws Exception:
Construtor do jogador. Ele lança exceção, pois, chama os métodos setNome e SetApelido.

MÉTODOS GET:

public long getTempoTotal(): retorna o tempo total do jogador.

public String getNome(): retorna nome do jogador.

public String getApelido(): retorna apelido do jogador.

public int getPontos(): retorna os pontos do jogador

public int getPontosFeitosnaUltimaRodada(): retorna os pontos ganhos na rodada anterior

public long getJogadaMaisRapidaEmUnidadedeTempo(): retorna a jogada mais rápida

public int getRecordPessoal(): retorna o recorde do jogador

MÉTODOS SET:

public void setNome(String nome) throws Exception: Método que coloca o nome do jogador e lança uma exception caso o nome tenha menos que 3 caracteres

public void setApelido(String apelido) throws Exception: Método para colocar o apelido do jogador lança uma exception caso o apelido tenha menos que 3 caracteres

public void setTempoInicial(): Método que coloca o tempo de início do jogo do jogador

public void setTempoTotal(): coloca o tempo total de jogo do jogador

private void setPontosGanhosnaUltimaRodada(int pontosGanhos): guarda os pontos que o jogador ganhou na rodada anterior

OUTRO MÉTODOS:
public void pontua(int pontos): adiciona pontos

public void foiJogadaMaisRapida(Long jogada): método que compara as jogadas e coloca a mais rápida

public int compareTo(Jogador outroJogador): Método que compara os pontos do jogador com outro jogador

Pacote View

Classe GeniusView:
Classe que inicia a aplicação.

Classe JLabeldateladeFundo:
Feita para utilizar um fundo

CONSTRUTOR:
public JLabeldateladeFundo()

Classe JLabelFundoSemLogo:
Criada para o fundo sem o logo do jogo

CONSTRUTOR:
public JLabelFundoSemLogo()

Classe MyJLabelwithSound:

CONSTRUTOR:

public MyJLabelwithSound()
Construtor para definir o som e a imagem

OUTROS MÉTODOS:

protected void startSound() throws UnsupportedAudioFileException, IOException, LineUnavailableException
Método que pega o arquivo áudio abre e executa

protected void startSound(String nomedoArquivo)
Método que pega o arquivo áudio abre e executa

public String getImagesBasePath()
Retorna o caminho da imagem

Classe MyJPanel:
Essa classe abstrata foi criada para instanciar MYJapanel e começar com layout definido e um caminho para as imagens. Herda de JPanel.

CONSTRUTOR:

protected MyJPanel()

MÉTODOS GET:
protected String getImagesPath()
protected String getBasePath()

Classe MyJTabbedPaneUI:
Classe que esconde a barra de abas
documentação desta classe:
https://docs.oracle.com/javase/8/docs/api/javax/swing/plaf/basic/BasicTabbedPaneUI.html
recomendo leitura, pois ela não explica o que nenhum dos dois métodos fazem, os nomes produzem significado próprio mas não encontrei o código ou explicação de como é feita.

CONSTRUTOR:

public MyJTabbedPaneUI()

OUTRO MÉTODO:

protected int calculateMaxTabHeight(int tabPlacement)

Classe TelaInicial:
Herda de MyJPanel. Classe utilizada para criar a tela inicial do jogo, quando ele é iniciado. Nela contém somente o botão “Jogar” que leva para a tela de seleção de modo.
Esta classe instância um objeto do tipo JLabeldateladeFundo para poder colocar uma imagem de fundo com a logo do Genius.
Esta classe instância um objeto do tipo MyJLabelwithSound para criar o botão de Jogar.

CONSTRUTOR:

public TelaInicial(JTabbedPane tabbedPane)

MOUSELISTENER:

lblbutao.addMouseListener(new MouseAdapter(){}):
Adiciona interação de click do botão “Jogar”. Ao clicar, o botão emite um som, muda para a Tela de Seleção de Modo e remove a atual.

Classe TeladeSelecaoModo:
Herda de MyJPanel. Classe utilizada para criar a tela de seleção de modos do jogo. Ela é instanciada quando o botão “Jogar” na Tela Inicial é pressionado.
Esta classe instância um objeto do tipo JLabeldateladeFundo para poder colocar uma imagem de fundo com a logo do Genius.
Esta classe instância três objetos do tipo MyJLabelwithSound para criar os botões: Campeonato, Individual e Carregar.

CONSTRUTOR:

public TeladeSelecaoModo(JTabbedPane tabbedPane)

MOUSELISTENERS:

lblindividual.addMouseListener(new MouseAdapter(){}):
Adiciona interação de click do botão “Individual ”. Ao clicar, o botão emite um som, muda de tela e remove a atual. Este botão leva o jogador até a tela de cadastro, informando ao jogo que existe somente um jogador.

lblCampeonato.addMouseListener(new MouseAdapter(){}):
Adiciona interação de click do botão “Campeonato ”. Ao clicar, o botão emite um som, muda de tela e remove a atual. Este botão leva o jogador até a tela de Campeonato, onde ele pode escolher a quantidade de jogadores.

lblCarregar.addMouseListener(new MouseAdapter(){}):

Adiciona interação de click do botão “Carregar ”. Ao clicar, o botão emite um som e abre um seletor de arquivos dentro da pasta “Saves”. É possível navegar pelos diretórios do computador e abrir somente arquivos .obj. Lembrando que somente arquivos .obj gerados por esse jogo irão ser aceitos.

Classe TelaCampeonatoSelecao

Herda de MyJPanel. Classe utilizada para criar a tela para escolher a quantidade de jogadores em um campeonato. Ela é instanciada quando o botão “Campeonato” na Tela de Seleção de Modo é pressionado.
Esta classe instancia um objeto do tipo JLabeldateladeFundo para poder colocar uma imagem de fundo com a logo do Genius.
Esta classe instancia quatro objetos do tipo MyJLabelwithSound para criar os botões: 2 Jogadores, 4 Jogadores, 8 Jogadores e Voltar.

CONSTRUTOR:

public TelaCampeonatoSelecao(JTabbedPane tabbedPane)

MOUSELISTENERS:

lbl2Jogadores.addMouseListener(new MouseAdapter(){}):
Adiciona interação de click do botão “2 Jogadores ”. Ao clicar, o botão emite um som, muda de tela e remove a atual. Este botão leva o jogador até a tela de cadastro, informando ao jogo que existem somente dois jogadores.

lbl4Jogadores.addMouseListener(new MouseAdapter(){}):
Adiciona interação de click do botão “4 Jogadores ”. Ao clicar, o botão emite um som, muda de tela e remove a atual. Este botão leva o jogador até a tela de cadastro, informando ao jogo que existem somente quatro jogadores.

lbl8Jogadores.addMouseListener(new MouseAdapter(){}):
Adiciona interação de click do botão “8 Jogadores ”. Ao clicar, o botão emite um som, muda de tela e remove a atual. Este botão leva o jogador até a tela de cadastro, informando ao jogo que existem oito jogadores.

lblVoltar.addMouseListener(new MouseAdapter(){}):
Adiciona interação de click do botão “Voltar”. Ao clicar, o botão emite um som e retorna para a tela de Seleção de Modo.

Classe TelaCadastro

Herda de MyJPanel. Classe utilizada para criar a tela de cadastro de jogadores. Ela é instanciada quando o botão “Individual” na Tela de Seleção de Modo é pressionado ou quando qualquer um dos botões, com exceção do botão Voltar, é pressionado na Tela de Seleção de Campeonato.
Esta classe instancia um objeto do tipo JLabelFundoSemLogo para poder colocar uma imagem de fundo com a logo do Genius.
Esta classe instancia dois objetos do tipo MyJLabelwithSound para criar os botões: Salvar e Voltar.
Esta classe instacia um objeto do tipo GeniusBase.
Para registrar os jogadores, esta classe possui três TextFields: textCampeonato, textNome e textApelido. O textCampeonato, após o primeiro jogador ser registrado, fica ineditável, para que todos os jogadores participem do mesmo campeonato.

CONSTRUTOR:

public TelaCadastro(JTabbedPane tabbedPane, int qtdJogadores)
Método construtor que recebe a quantidade de jogadores para poder cadastrar todos.

MOUSELISTENERS:

lblSalvar.addMouseListener(new MouseAdapter(){}):
Adiciona interação de click do botão “Salvar”. Ao clicar, o botão emite um som e instancia um objeto do tipo Jogador dentro da lista de jogadores do jogo instanciado anteriormente. Após isso, ele limpa os campos, menos o campo Campeonato, para que o outro jogador possa preenchê-los. Quando for clicado pelo último jogador da lista, leva o(s) jogador(es) até a tela de jogo.

lblVoltar.addMouseListener(new MouseAdapter(){}):
Adiciona interação de click do botão “Voltar”. Ao clicar, o botão emite um som e retorna para a tela de Seleção de Modo.

Classe TelaJogo

Herda de MyJPanel e implementa a interface Runnable. Classe utilizada para jogar o próprio Genius.
Esta classe instancia quatro objetos do tipo GeniusLabels, que são os botões: Azul, Vermelho, Amarelo e Verde.
Esta classe instancia seis objetos do tipo MyJLabelwithSound, que são: Iniciar, Salvar, Extras, Carregar, Ritmo e Dificuldade.

CONSTRUTOR:

public TelaJogo(JTabbedPane tabbedPane, Genius jogo):

Método construtor que recebe o Genius instanciado na classe TelaCadastro.

MÉTODOS DE CONFIGURAÇÃO:

private void keyAndMouseMapping(GeniusLabels geniusLabel):
método que é utilizado para selecionar o botão do jogo com mouse ou teclado.

private void instanciaBotoes():
Método utilizado para instanciar os botões do Genius. Esse método chama o método keyAndMouseMapping(GeniusLabels geniusLabel) quatro vezes, para cada um dos botões.

private void ativaDesativaModoMonoCor() :
Método que modifica as cores dos botões do jogo.

MOUSELISTENERS:

btnIniciar.addMouseListener(new MouseAdapter(){}):
Método utilizado para iniciar o jogo, chamando todos os métodos necessários para que isso ocorra.

btnSalvar.addMouseListener(new MouseAdapter(){}):
Método utilizado para salvar um jogo quando aperta o botão "Salvar".
O método salva somente arquivos .obj. Ele possui um filtro para que não seja possível salvar ".obj.obj". Logo, se já possuir .obj, ele salva sem adicionar o .obj novamente.

btnCarregar.addMouseListener(new MouseAdapter(){}):
Método utilizado para carregar um jogo quando aperta o botão "Carregar".
O método utiliza um filter para poder carregar somente arquivos .obj

btnExtras.addMouseListener(new MouseAdapter(){}):
Método utilizado para mudar para os modos extras do jogo quando aperta o botão “Extras”.

btnDificuldade.addMouseListener(new MouseAdapter(){}):
Método utilizado para mudar a dificuldade do jogo atual quando aperta o botão azul central direito.

btnRitmSound.addMouseListener(new MouseAdapter(){}):
Método utilizado para mudar o ritmo do jogo atual quando aperta o botão azul central esquerdo.

geniusLabel.addMouseListener(new MouseAdapter(){}):

Método utilizado para o jogador escolher uma cor. Ele é chamado quando o jogador aperta em um dos quatro botões coloridos do Genius.

MÉTODOS DE JOGO:

private void getInformacoes(final GeniusLabels botao) throws Exception:
Método utilizado para validar a jogada que o jogador realizou ao apertar em um dos quatro botões coloridos do Genius. Método responsável por chamar e validar os retornos dos métodos responsáveis por: pegar o jogador atual, verificar se ele perdeu, fazer o botão pressionado piscar, fazer o placar do jogo caso finalizado e atualizar informações na tela do jogo
Se o último jogador perder, esse método levará os jogadores para a tela de Placar.

private void atualizaInformacoes():
Mini placar da tela de jogo. Método responsável por atualizar as informações na tela do jogo.

public synchronized void run():
Método responsável por guardar a sequência de cores e exibi-la na tela.

Classe TelaPlacar
Herda de MyJPanel. Classe utilizada para exibir o placar final do jogo, quando todos os jogadores perdem.
Esta classe instancia dois objetos do tipo MyJLabelwithSound, que são: Desempate e Voltar. O botão de Desempate só estará disponível caso haja empate, senão ele ficará cinza.

CONSTRUTOR:
public TelaPlacar(JTabbedPane tabbedPane, Genius jogo)

MÉTODO DE CONFIGURAÇÃO:

MÉTODOS MOUSELISTENERS:
public void desenhaPlacar(Genius jogo):
Método utilizado para dispor os jogadores por ordem de pontos no placar.

lblVoltar.addMouseListener(new MouseAdapter(){}):
Adiciona interação de click do botão “Voltar”. Ao clicar, o botão emite um som e retorna para a tela de início.

lblEmpate.addMouseListener(new MouseAdapter(){}):

Adiciona interação de click do botão “Desempate”. Ao clicar, o botão emite um som e retorna para a tela de jogo.

Pacote geniusLabels

Classe GeniusLabels

public GeniusLabels(String nomedaImagemBase, String nomedaImagemBranca, String arquivoSom, Cor cor, char keyChar)
Instância os botões do jogo

public synchronized void pisca() throws UnsupportedAudioFileException, IOException, LineUnavailableException
Faz o botão piscar através de uma thread e imagens

protected void setImagem(String nomeDaImagem)
Método feito para inserir imagem

public abstract void setImagemParaRosa();
muda imagem para rosa

public abstract void setImagemPadrao();
muda imagem para cor padrão

public Cor getCor()
Diz a cor do botão

public char getKeyChar()
Diz a tecla do botão

Classe AmareloLabel
Herda de GeniusLabels. Cria o botão amarelo e coloca as imagens

Classe AzulLabel
Herda de GeniusLabels. Cria o botão azul e coloca as imagens

Classe VerdeLabel
Herda de GeniusLabels. Cria o botão verde e coloca as imagens

Classe VermelhoLabel
Herda de GeniusLabels. Cria o botão vermelho e coloca as imagens

Pacote Saves

Esse pacote contém os jogos salvos que o jogador deseja guardar. É o diretório padrão que irá aparecer quando o jogador apertar em “Salvar” ou “Carregar”.