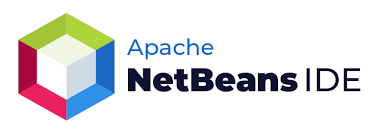
**Funchal, 23 de janeiro de 2021**

**Curso: Engenharia Informática**

**Disciplina: Programação Orientada a Objetos**

**FCEE – Faculdade de Ciências Exatas e da Engenharia**

**Projeto nº2**

**Trabalho realizado por:**

Pedro Jardim nº 2015118

Francisco Belo nº 2076420

Diogo Rodrigues nº 2046719

# Índice

[Índice 2](#_Toc93872778)

[Introdução 4](#_Toc93872779)

[Objetivos 5](#_Toc93872780)

[Desenvolvimento 5](#_Toc93872781)

[Classe Config 6](#_Toc93872782)

[Classe Família 6](#_Toc93872783)

[Classe Pessoa 6](#_Toc93872784)

[Classe Associate 6](#_Toc93872785)

[Classe Mafioso 6](#_Toc93872786)

[Classe Boss 7](#_Toc93872787)

[Classe Underboss 7](#_Toc93872788)

[Classe Caporegime 7](#_Toc93872789)

[Classe Consiglieri 7](#_Toc93872790)

[Classe Soldier 7](#_Toc93872791)

[Classe Negócio 8](#_Toc93872792)

[Classe Roubo, LavagemDinheiro, Droga, Armas 8](#_Toc93872793)

[Classe Casino 8](#_Toc93872794)

[Interface Policiável 8](#_Toc93872795)

[Classe RandomAtributesGenerator 8](#_Toc93872796)

[Conclusão 9](#_Toc93872797)

[Bibliografia 9](#_Toc93872798)

[Código do Projeto 9](#_Toc93872799)

[Main 9](#_Toc93872800)

[Armas 21](#_Toc93872801)

[Associate 23](#_Toc93872802)

[Boss 23](#_Toc93872803)

[CapoRegime 27](#_Toc93872804)

[Casino 28](#_Toc93872805)

[Config 30](#_Toc93872806)

[Consiglieri 31](#_Toc93872807)

[Droga 32](#_Toc93872808)

[Familia 34](#_Toc93872809)

[LavagemDinheiro 39](#_Toc93872810)

[Mafioso 41](#_Toc93872811)

[Negocio 41](#_Toc93872812)

[Pessoa 44](#_Toc93872813)

[Policiavel 46](#_Toc93872814)

[RandomAtributesGenerator 47](#_Toc93872815)

[Roubo 48](#_Toc93872816)

[Soldier 51](#_Toc93872817)

[Underboss 51](#_Toc93872818)

# Introdução

Este projeto foi realizado no âmbito da cadeira “Programação Orientada por Objetos”. Este projeto teve como objetivo aplicar os conceitos de **Programação Orientada a Objetos** através da linguagem **JAVA**.

Foi-nos proposto na unidade curricular de **Programação Orientada a Objetos** que neste segundo trabalho fossemos capazes de desenvolver primeiramente da elaboração de um diagrama UML e só depois, recorrendo ao uso da aplicação **Apache NetBeans IDE, elaborar** um programa na qual se baseava no **gerenciamento de uma máfia**, onde se tinha de ter presente pelo menos 5 famílias, que através deste programa em Java seriamos capazes de, conforme a família selecionada inicialmente para a qual iriamos gerenciar, **escolher inúmeras opções** desde o **recrutamento de indivíduos para os diferentes estatutos da família**, tornando-se assim de **linhagem**, até aos **negócios** de onde se obtêm o **rendimento** para a sustentabilidade de cada família mafiosa podendo estes serem abordados pela **policia** podendo então haver a possibilidade de elementos da equipa que se encontrava relacionada com esse negócio **serem presos** ou **se tornaram cúmplices da polícia**, podendo no futuro ter consequências graves como o tal indesejado do “**desaparecimento misterioso**”.

Contudo, recorremos também, embora que não fosse pedido nem requisito, à aplicação do **GitHub,** pois através deste recurso o trabalho em grupo foi bastante facilitado, pois a medida que íamos programando era possível analisar o código feito por todos os membros através de **merge requests**. Também a partilha de código à medida que ia sendo elaborado para com os restantes elementos torna-se muito mais facilitada. Também utilizamos os **slides** fornecidos pelos docentes nas aulas **teóricas** e **práticas**, **fóruns**, **vídeos** no **Youtube** e **autoinvestigação**.

# Objetivos

O objetivo do nosso projeto foi conseguir concluir o objeto ao demonstrar o domínio da linguagem **JAVA** e do conceito de **POO**. Tivemos dificuldades ao longo do projeto assim não conseguindo desenvolver ao máximo cada objetivo do projeto.

# Desenvolvimento

Neste segundo projeto antes de começar logo com a programação em si já das várias componentes a implementar, tivemos de fazer, assim de uma forma global um **diagrama UML** recorrendo ao **diagrams.net**, à elaboração das tabelas já das **classes** e suas **subclasses** (quando assim fosse o caso) que desde logo se previa com a interpretação do enunciado. Assim sendo fizemos a sua implementação já introduzindo também algumas **variáveis de instância** da respetiva classe. Posto isto feito, o processo para começar a programar já se tornou mais fácil de estruturar.

Quanto a **nível de código**, começamos tendo por base o tal diagrama elaborado, por implementar as **classes principais**, e elaborando já em cada uma as **respetivas variáveis** seguido do **construtor** e como não se pode quebrar o **encapsulamento** (visto as variáveis de instancia serem **private**) fazer os respetivos **modificadores de acesso**, conhecidos como os **Getters** e **Setters** e por fim o respetivo **toString()** que implementava as **informações características** de cada classe.

Posteriormente, já feita a elaboração do **menu inicia**l na qual constam as diversas opções aquando gerir a família de mafiosos após serem logo inicializadas 5 delas tendo o **utilizador** antes de passar ao menu de gerenciamento, ter de **optar** por qual **família iria abordar**.

Contudo, foi através das várias opções que continha este menu (já apresentadas no enunciado do projeto) que passamos então ao desenvolvimento dos vários **métodos** em cada uma das classes correspondentes então a cada uma dessas opções. Assim sendo de um modo geral temos as seguintes classes:

## Classe Config

Nela é onde se consegue guardar/obter as diferentes famílias, sendo então depois pedido ao utilizador qual deseja gerir.

## Classe Família

Onde toda a **informação** (riqueza, custos, obituário, …) e **membros** (Boss, Underboss, Consiglieri, …) fica tudo “guardado” podendo se obter/atribuir certos conteúdos com os demais métodos nela presentes. Nela também podemos realçar que consta de um **ArrayList** de presos onde ficam guardados os **membros encarcerados** da família.

## Classe Pessoa

Uma superclasse de onde se faz a elaboração do construtor com as demais variáveis de instância.

## Classe Associate

Subclasse de **Pessoa**, que neste caso recebe todos os parâmetros de Pessoa (usando o **super**), sendo que neste caso difere de **não** serem de **linhagem**.

## Classe Mafioso

Subclasse também de **Pessoa**, sendo que estes já são de **linhagem** pois estes já **pertencem** a uma **família** especifica.

## Classe Boss

**Subclasse** de **Mafioso**. Nesta classe é onde se implementou os diversos métodos para o **recrutamento** dos vários elementos da família, e também o que possibilita **criar** e **atribuir** um **negócio** a um Caporegime e à sua família.

## Classe Underboss

**Subclasse** de **Mafioso**. Nela está implementada o método do **LoyaltyTest** que basicamente vai **analisar** todos os **Soldiers** e **caporegimes** da família e consoante for o valor obtido da sua **lealdade** e/ou ser **informador** (estar envolvido com a polícia) estes sofrem o tal “desaparecimento misterioso”.

## Classe Caporegime

**Subclasse** de **Mafioso**. Nesta apenas contem os métodos necessário para não quebrar o encapsulamento das suas respetivas variáveis de instância.

## Classe Consiglieri

**Subclasse** de **Mafioso**. Consta 2 métodos necessários para o seu menu de opções, **expandirNegocio** e **mafiaSitdown**. No primeiro caso, para se poder expandir é feito a adição de um valor ao **ValorBaseTributável** (variável de instância de Negócio) consoante for o valor da **característica Estratega** do próprio **Consiglieri**. No segundo caso, conforme o seu **nível de Estratega**, é feito um **aumento** ou **redução** dos **custos fixos** que constam na família.

## Classe Soldier

**Subclasse** de **Mafioso**. Apenas tem presente os métodos que recebe por herança da sua superclasse.

## Classe Negócio

**Superclasse** de onde vão ser especializados alguns diferentes tipos de negócios sendo um deles não policiável. Aqui é feito o método que vai **analisar as equipas** relacionados com cada negócio obtendo tanto a **média do músculo** como **da inteligência dos Soldiers** dessa tal equipa, e também um método que adiciona um X valor de associados ao negócio. Foi também implementado **um método abstrato AtualizaValoresNegocio**.

## Classe Roubo, LavagemDinheiro, Droga, Armas

Tudo **subclasses** de **Negócio**. Em cada uma delas é feito um **override** sobre o **método de criar associados e associá-los ao negócio**, **o método de analisar o negócio**, onde para além de estabelecer novos valores para a riqueza e/ou valor tributável, é também implementado um **método** de uma **interface**, o **PoliciarNegocio**, que faz a análise dos Soldiers associados ao negócio e caso tenham um X valor de lealdade tornam-se **informadores**, senão podem (ou não) ser **presos**.

## Classe Casino

**Subclasse** de **Negócio**. O que esta classe difere das restantes subclasses é que esta **não implementa a interface Policiável**, não podendo então a polícia atuar no mesmo.

## Interface Policiável

Interface policiável onde apenas vai ser implementada nos tipos de negócios em que estes sejam policiáveis.

## Classe RandomAtributesGenerator

Nesta classe é apenas elaborado diversos **métodos** que retornam um valor **random** conforme o que se pretende para que possam ser utilizados nos diversos casos ao longo do desenvolvimento do projeto para estabelecer certos valores como por exemplo para os valores das características dos vários membros da família.

# Conclusão

Dado por concluído este segundo projeto, podemos referir que este serviu de certa forma para **consolidarmos** a nossa capacidade de elaborar programas recorrendo à programação em Java. Contudo o código ficou **operacional** embora possa mesmo haver pelo meio, partes de código em que possa haver a possibilidade de **simplificação** de código ou até mesmo outras formas de o elaborar de forma mais simples não complicando assim tanto.

# Bibliografia

1. <https://moodle.cee.uma.pt/> (Slides dos professores)
2. <https://www.w3schools.com/java/> (JAVA tutorial)
3. <https://stackoverflow.com/> (Duvidas sobre Java)

# Código do Projeto

## Main

package com.mycompany.projeto2;

import java.util.Scanner;

public class Main {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

int opcao, capoRegimeId = 0, presoId, negocioId = 0, familiaId;

String nome, nomeNegocio = "";

boolean sair = false, underbossMenu = false, consiglieriMenu = false, capoRegimeActive = false, isValid = true;

RandomAtributesGenerator randomAtributesGenerator = new RandomAtributesGenerator();

Scanner scan = new Scanner(System.in);

Boss boss;

Config config = new Config();

Familia familia1 = new Familia(1, "Peaky Blinders", 1000.0, 2000.0, 100.0);

config.addFamilia(familia1);

Boss chefeDaMafiaFamilia1 = new Boss(familia1, "Thomas Shelby", 1, randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), 0, false, true, false);

familia1.setBoss(chefeDaMafiaFamilia1);

Familia familia2 = new Familia(2, "Yakuza", 3000.0, 5000.0, 100.0);

config.addFamilia(familia2);

Boss chefeDaMafiaFamilia2 = new Boss(familia2, "Jackie Chan", 1, randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), 0, false, true, false);

familia2.setBoss(chefeDaMafiaFamilia2);

Familia familia3 = new Familia(3, "Sicilian Mafia", 10000.0, 20000.0, 100.0);

config.addFamilia(familia3);

Boss chefeDaMafiaFamilia3 = new Boss(familia3, "Al Capone", 1, randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), 0, false, true, false);

familia3.setBoss(chefeDaMafiaFamilia3);

Familia familia4 = new Familia(4, "Sinaloa Cartel", 100000.0, 200000.0, 100.0);

config.addFamilia(familia4);

Boss chefeDaMafiaFamilia4 = new Boss(familia4, "Pablo Escobar", 1, randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), 0, false, true, false);

familia4.setBoss(chefeDaMafiaFamilia4);

Familia familia5 = new Familia(5, "Triad", 1000000.0, 2000000.0, 100.0);

config.addFamilia(familia5);

Boss chefeDaMafiaFamilia5 = new Boss(familia5, "Heung Wah-yim", 1, randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), 0, false, true, false);

familia5.setBoss(chefeDaMafiaFamilia5);

Underboss underboss;

Consiglieri consiglieri;

Familia familia;

Familia familiaRival;

CapoRegime capoRegime = null;

do {

if (!config.isFamiliaFoiEscolhida()) {

System.out.println("Welcome to “The five crime families of New York");

System.out.println("Selecione uma das seguintes familias para poder fazer a gestao da mesma (Introduza o id da familia)");

System.out.println(config.getFamilias());

while (!scan.hasNextInt()) {

System.out.println("O valor introduzido não é valido!");

System.out.println("Introduza novamente um valor valido");

scan.next(); // this is important!

}

familiaId = scan.nextInt();

familia = config.getFamilia(familiaId);

System.out.println("A familia escolhida foi: " + familia.getNome());

} else {

System.out.println("familia escolhida: " + config.getFamiliaEscolhida() + "\n\n");

System.out.println(" MENU MÁFIA ");

System.out.println("1 - Boss");

System.out.println("2 - UnderBoss");

System.out.println("3 - Consigliere");

System.out.println("4 - Familia");

System.out.println("5 - Mudar de Familia");

System.out.println("6 - Sair da aplicação");

System.out.println("Introduza uma opção: ");

while (!scan.hasNextInt()) {

System.out.println("O valor introduzido não é valido!");

System.out.println("Introduza novamente um valor valido");

scan.next(); // this is important!

}

opcao = scan.nextInt();

boss = config.getFamiliaEscolhida().getBoss();

switch (opcao) {

case 1:

System.out.println("\n Opções válidas ");

System.out.println("1. Recruta Soldier");

System.out.println("2. Recruta CapoRegime");

System.out.println("3. Recruta Underboss");

System.out.println("4. Gera negócios para caporegime");

System.out.println("5. Nomear consiglieri");

System.out.println("O que pretende?");

while (!scan.hasNextInt()) {

System.out.println("O valor introduzido não é valido!");

System.out.println("Introduza novamente um valor valido");

scan.next(); // this is important!

}

opcao = scan.nextInt();

switch (opcao) {

case 1:

if(capoRegimeActive){

isValid = true;

System.out.println("Atualmente existe: " + config.getFamiliaEscolhida().getSoldiers());

System.out.println("CapoRegimes disponiveis" + config.getFamiliaEscolhida().getCapoRegimes());

System.out.println("Introduza o ID do capoRegime que o soldier ira fazer parte");

while (isValid) {

if (scan.hasNextInt()) {

capoRegimeId = scan.nextInt();

} else {

scan.next();

continue;

}

isValid = false;

}

if (config.getFamiliaEscolhida().getCapoRegime(capoRegimeId) != null){

capoRegime = config.getFamiliaEscolhida().getCapoRegime(capoRegimeId);

System.out.println("Introduza o nome do soldier");

nome = scan.next();

boss.RecrutaSoldier(nome, capoRegime);

}else{

System.out.println("Não existe nenhum caporegime com esse ID \n");

}

}else{

System.out.println("Tem que criar primeiro um capoRegime antes de recrutar um soldier\n");

}

break;

case 2:

System.out.println("Introduza o nome do capoRegime");

nome = scan.next();

boss.RecrutaCapoRegime(nome);

capoRegimeActive = true;

break;

case 3:

System.out.println("Introduza o nome do Underboss");

nome = scan.next();

boss.RecrutaUnderboss(nome);

underbossMenu = true;

break;

case 4:

isValid = true;

//Gera negocios para caporegime

System.out.println("CapoRegimes disponiveis" + config.getFamiliaEscolhida().getCapoRegimes());

capoRegime = null;

while (capoRegime == null) {

System.out.println("Introduza o ID do capoRegime que ira atribuir o novo negocio");

capoRegimeId = scan.nextInt();

capoRegime = config.getFamiliaEscolhida().getCapoRegime(capoRegimeId);

}

if (config.getFamiliaEscolhida().getCapoRegime(capoRegimeId) != null){

System.out.println("capoRegime: " + capoRegime);

boss.geraNegocio(capoRegime, config);

System.out.println("negocios da familia: " + config.getFamiliaEscolhida().getNegocios());

}

else{

System.out.println("Não existe nenhum caporegime com esse ID \n");

}

break;

case 5:

System.out.println("Introduza o nome do Consiglieri");

nome = scan.next();

boss.NomearConsiglieri(nome);

consiglieriMenu = true;

break;

default:

System.out.println("\nOpção inválida!\n");

}

break;

case 2:

if (underbossMenu) {

System.out.println("\n Opções válidas ");

System.out.println("1. Libertar mafioso");

System.out.println("2. Período contabilístico");

System.out.println("3. Loyalty test");

System.out.println("O que pretende?");

while (!scan.hasNextInt()) {

System.out.println("O valor introduzido não é valido!");

System.out.println("Introduza novamente um valor valido");

scan.next(); // this is important!

}

opcao = scan.nextInt();

switch (opcao) {

case 1:

if (config.getFamiliaEscolhida().getPresosDaFamilia().size() > 0) {

for (int i = 0; i < config.getFamiliaEscolhida().getPresosDaFamilia().size(); i++) {

System.out.println("ID: " + config.getFamiliaEscolhida().getPresosDaFamilia().get(i).getCcId() + ", Nome: " + config.getFamiliaEscolhida().getPresosDaFamilia().get(i).getNome());

}

System.out.println("Introduza o id do mafioso que deseja libertar da prisao");

while (!scan.hasNextInt()) {

System.out.println("O valor introduzido não é valido!");

System.out.println("Introduza novamente um valor valido");

scan.next(); // this is important!

}

presoId = scan.nextInt();

config.getFamiliaEscolhida().libertarPreso(presoId);

} else {

System.out.println("Nenhum mafioso encontra-se preso neste momento\n");

}

break;

case 2:

//Periodo Contabilistico

if (config.getFamiliaEscolhida().getCapoRegimes().size() > 0) {

for (int i = 0; i < config.getFamiliaEscolhida().getCapoRegimes().size(); i++) {

config.getFamiliaEscolhida().getCapoRegimes().get(i).getNegocios();

for (int j = 0; j < config.getFamiliaEscolhida().getCapoRegimes().get(i).getNegocios().size(); j++) {

config.getFamiliaEscolhida().getCapoRegimes().get(i).getNegocios().get(j).AtualizaValoresNegocio();

double tributos = config.getFamiliaEscolhida().getCapoRegimes().get(i).getNegocios().get(j).getValorAtualTributavel();

double ganhos = config.getFamiliaEscolhida().getCapoRegimes().get(i).getNegocios().get(j).getRentabilidade();

double lucro = ganhos - tributos;

config.getFamiliaEscolhida().setRiqueza(config.getFamiliaEscolhida().getRiqueza() + lucro);

}

}

underboss = config.getFamiliaEscolhida().getUnderboss();

underboss.loyaltyTest(config.getFamiliaEscolhida());

System.out.println("A riqueza da familia foi atualizada!\n");

System.out.println("A riqueza atual é " + config.getFamiliaEscolhida().getRiqueza()+"\n");

} else {

System.out.println("Não existem de momento CapoRegimes ligados à familia!\n");

}

break;

case 3:

underboss = config.getFamiliaEscolhida().getUnderboss();

underboss.loyaltyTest(config.getFamiliaEscolhida());

break;

default:

System.out.println("\nOpção inválida!\n");

break;

}

} else {

System.out.println("O Menu underBoss esta indisponivel, porque o underboss ainda nao foi criado!\n");

}

break;

case 3:

if (consiglieriMenu) {

System.out.println("\n Opções válidas ");

System.out.println("1. Expandir negócio");

System.out.println("2. Mafia sitdown");

System.out.println("O que pretende?");

while (!scan.hasNextInt()) {

System.out.println("O valor introduzido não é valido!");

System.out.println("Introduza novamente um valor valido");

scan.next(); // this is important!

}

opcao = scan.nextInt();

switch (opcao) {

case 1:

isValid = true;

System.out.println("Negocios: " + config.getFamiliaEscolhida().getNegocios());

System.out.println("Introduza o nome do negocio que deseja expandir");

while (isValid) {

if (scan.hasNext()) {

nomeNegocio = scan.next();

} else {

scan.next();

continue;

}

isValid = false;

}

System.out.println("Consiglieri: " + config.getFamiliaEscolhida().getConsiglieri());

consiglieri = config.getFamiliaEscolhida().getConsiglieri();

consiglieri.expandirNegocio(nomeNegocio);

break;

case 2:

System.out.println("Mafia Sitdown");

if (boss.getFamiliaPartilhada() != null) {

int estrategaConsiglieri = config.getFamiliaEscolhida().getConsiglieri().getEstratega();

System.out.println("Custos fixos:");

System.out.println("Custo da familia " + config.getFamiliaEscolhida().getNome() + " :" + config.getFamiliaEscolhida().getCustoFixo());

System.out.println("Custo da familia partilhada " + boss.getFamiliaPartilhada().getNome() + ": " + boss.getFamiliaPartilhada().getCustoFixo());

if (estrategaConsiglieri >= 90) {

config.getFamiliaEscolhida().setCustoFixo(0);

boss.getFamiliaPartilhada().setCustoFixo(0);

} else if (estrategaConsiglieri >= 70 && estrategaConsiglieri < 90) {

config.getFamiliaEscolhida().setCustoFixo(config.getFamiliaEscolhida().getCustoFixo() / 5);

boss.getFamiliaPartilhada().setCustoFixo(boss.getFamiliaPartilhada().getCustoFixo() / 5);

} else if (estrategaConsiglieri >= 50 && estrategaConsiglieri < 70) {

config.getFamiliaEscolhida().setCustoFixo(config.getFamiliaEscolhida().getCustoFixo() / 2);

boss.getFamiliaPartilhada().setCustoFixo(boss.getFamiliaPartilhada().getCustoFixo() / 2);

} else {

config.getFamiliaEscolhida().setCustoFixo(config.getFamiliaEscolhida().getCustoFixo() \* 2);

boss.getFamiliaPartilhada().setCustoFixo(boss.getFamiliaPartilhada().getCustoFixo() \* 2);

}

System.out.println("Custos Fixos Atualizados!");

System.out.println("Custo da familia atual: " + config.getFamiliaEscolhida().getCustoFixo());

System.out.println("Custo da familia partilhada: " + boss.getFamiliaPartilhada().getCustoFixo()+"\n");

} else {

System.out.println("A familia nao tem relações com outra familia!\n");

}

break;

default:

System.out.println("\nOpção inválida!\n");

}

} else {

System.out.println("O Menu consiglieri esta indisponivel, porque o consiglieri ainda nao foi criado!\n");

}

break;

case 4:

System.out.println("\n Opções válidas ");

System.out.println("1. Retrato de familia");

System.out.println("2. Plano de negocios");

System.out.println("3. Mapa de custos");

System.out.println("4. Espirito de equipa");

System.out.println("5. Obituario");

System.out.println("6. Encarcerados");

System.out.println("7. All out war");

System.out.println("O que pretende?");

while (!scan.hasNextInt()) {

System.out.println("O valor introduzido não é valido!");

System.out.println("Introduza novamente um valor valido");

scan.next(); // this is important!

}

opcao = scan.nextInt();

switch (opcao) {

case 1:

System.out.println("Boss: " + config.getFamiliaEscolhida().getBoss() + "\n");

System.out.println("UnderBoss: " + config.getFamiliaEscolhida().getUnderboss() + "\n");

System.out.println("Consiglieri: " + config.getFamiliaEscolhida().getConsiglieri() + "\n");

System.out.println("CapoRegimes: " + config.getFamiliaEscolhida().getCapoRegimes() + "\n");

System.out.println("Soldiers: " + config.getFamiliaEscolhida().getSoldiers() + "\n");

System.out.println("Negócios: " + config.getFamiliaEscolhida().getNegocios() + "\n");

System.out.println("Custo Fixo: " + config.getFamiliaEscolhida().getCustoFixo() + "\n");

System.out.println("Riqueza Acumulada: " + config.getFamiliaEscolhida().getRiqueza() + "\n");

System.out.println("Obituário: " + config.getFamiliaEscolhida().getObituarioFamiliares() + "\n");

System.out.println("Encarcerados: " + config.getFamiliaEscolhida().getPresosDaFamilia() + "\n");

break;

case 2:

System.out.println("Negócios: " + config.getFamiliaEscolhida().getNegocios() + "\n");

break;

case 3:

System.out.println("Custos: " + config.getFamiliaEscolhida().getCustoFixo() + "\n");

break;

case 4:

for (int i = 0; i < config.getFamiliaEscolhida().getCapoRegimes().size(); i++) {

System.out.println("O CapoRegime " + config.getFamiliaEscolhida().getCapoRegimes().get(i).getNome() + " tem os seguintes soldiers na sua equipa:" + "\n");

System.out.println(config.getFamiliaEscolhida().getCapoRegimes().get(i).getSoldiers().toString());

}

break;

case 5:

System.out.println("Obituario: " + config.getFamiliaEscolhida().getObituarioFamiliares());

break;

case 6:

System.out.println("Encarcerados: ");

System.out.println(config.getFamiliaEscolhida().getPresosDaFamilia());

break;

case 7:

System.out.println("Selecione uma segunda família para o All Out War (Introduza o id da segunda familia.)");

System.out.println(config.getFamilias());

System.out.println("A familia que está selecionada é: " + config.getFamiliaEscolhida().getNome() + "\n");

System.out.println("Escolha a segunda família.");

familiaId = scan.nextInt();

familiaRival = config.getFamilia(familiaId);

System.out.println("A segunda familia escolhida foi: " + familiaRival.getNome());

System.out.println("A segunda familiagetConsiglieri: " + familiaRival.getConsiglieri());

if (familiaRival.getConsiglieri() == null) {

System.out.println("A familia rival nao tem um consiglieri");

System.out.println("Introduza o nome do consiglieri da familia Rival");

String consiglieriFamiliaRival = scan.next();

familiaRival.getBoss().NomearConsiglieri(consiglieriFamiliaRival);

}

if (config.getFamiliaEscolhida().getConsiglieri().getEstratega() > 60 && familiaRival.getConsiglieri().getEstratega() < 40) {

config.getFamiliaEscolhida().setCustoFixo(config.getFamiliaEscolhida().getCustoFixo() \* 1.5);

familiaRival.setCustoFixo(familiaRival.getCustoFixo() / 1.5);

}

if (config.getFamiliaEscolhida().getConsiglieri().getEstratega() < 40 && familiaRival.getConsiglieri().getEstratega() > 60) {

config.getFamiliaEscolhida().setCustoFixo(config.getFamiliaEscolhida().getCustoFixo() / 1.5);

familiaRival.setCustoFixo(familiaRival.getCustoFixo() \* 1.5);

}

if (config.getFamiliaEscolhida().getConsiglieri().getEstratega() > 80 && familiaRival.getConsiglieri().getEstratega() < 30) {

config.getFamiliaEscolhida().setCustoFixo(config.getFamiliaEscolhida().getCustoFixo() \* 3);

familiaRival.setCustoFixo(familiaRival.getCustoFixo() / 3);

}

if (config.getFamiliaEscolhida().getConsiglieri().getEstratega() < 30 && familiaRival.getConsiglieri().gê Estratega() > 80) {

config.getFamiliaEscolhida().setCustoFixo(config.getFamiliaEscolhida().getCustoFixo() / 3);

familiaRival.setCustoFixo(familiaRival.getCustoFixo() \* 3);

}

System.out.println("Custo fixo familia atual: " + config.getFamiliaEscolhida().getCustoFixo());

System.out.println("Custo fixo familia rival: " + familiaRival.getCustoFixo()+"\n");

break;

case 8:

System.out.println("\nSaindo da aplicação...");

sair = true;

break;

default:

System.out.println("\nOpção inválida!\n");

}

break;

case 5:

config.setFamiliaFoiEscolhida(false);

break;

case 6:

sair = true;

System.out.println("Saiu da app!");

break;

default:

System.out.println("\nOpção inválida, tente novamente!\n");

}

}

} while (!sair);

}

}

## Armas

package com.mycompany.projeto2;

import java.util.ArrayList;

public class Armas extends Negocio implements Policiavel {

private ArrayList<Associate> associatesArmas;

private RandomAtributesGenerator randomAtributesGenerator = new RandomAtributesGenerator();

private Familia familia;

public Armas(Familia familia, String nome, double ValorBaseTributavel, int ProbPolicia, boolean Policiavel, CapoRegime CapoRegime, double rentabilidade, double ValorAtualTributavel, boolean Musculado, boolean Inteligente) {

super(familia, nome, ValorBaseTributavel, ProbPolicia, true, CapoRegime, rentabilidade, ValorAtualTributavel, true, false);

this.familia = familia;

associatesArmas = new ArrayList<Associate>();

this.createAssociates();

}

//Cria x valor de associados para este negócio

@Override

public void createAssociates() {

int quantidadeDeAssociates = randomAtributesGenerator.generateRandomNumber();

System.out.println("quantidadeDeAssociates " + quantidadeDeAssociates);

for (int i = 0; i < quantidadeDeAssociates; i++) {

associatesArmas.add(new Associate(familia, randomAtributesGenerator.generateRandomName("Associate"), 1, randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), 0, false, true, false));

}

System.out.println("Associates Criados: " + associatesArmas);

}

//Obtem média do Musculo da equipa de Soldiers e consoante esse valor é atribuido novos valores para a rentabilidade, Valor base tributavel e probabilidade da policia atuar

@Override

public void AtualizaValoresNegocio() {

if (VerificaMusculo() >= 50 && VerificaMusculo() < 70) {

double novaRentabilidade = getRentabilidade() \* 1.5;

setRentabilidade(novaRentabilidade);

setValorAtualTributavel(novaRentabilidade \* 0.15);

policiarNegocio();

} else if (VerificaMusculo() >= 70 && VerificaMusculo() < 90) {

double novaRentabilidade = getRentabilidade() \* 2;

setRentabilidade(novaRentabilidade);

setProbPolicia(getProbPolicia() - 5);

setValorAtualTributavel(novaRentabilidade \* 0.2);

policiarNegocio();

} else if (VerificaMusculo() >= 95) {

double novaRentabilidade = getRentabilidade() \* 3;

setRentabilidade(novaRentabilidade);

setProbPolicia(getProbPolicia() - 10);

setValorAtualTributavel(novaRentabilidade \* 0.3);

policiarNegocio();

} else {

setProbPolicia(getProbPolicia() + 10);

policiarNegocio();

}

}

//Método que analisa os valores de lealdade da equipa e consoante esse valor, o soldier pode se torna informador ou ir preso

@Override

public void policiarNegocio() {

RandomAtributesGenerator randomAtributesGenerator = new RandomAtributesGenerator();

int atuar = randomAtributesGenerator.generateRandomProbabilidadePoliciaAtuar();

int prender = randomAtributesGenerator.generateProbabilidadeSerPreso();

if (atuar <= getProbPolicia()) {

System.out.println("A policia está a atuar no negocio.");

for (int i = 0; i < getCapoRegime().getSoldiers().size(); i++) {

if (getCapoRegime().getSoldiers().get(i).getLealdade() < 15) {

getCapoRegime().getSoldiers().get(i).setInformador(true);

} else if (prender <= getCapoRegime().getSoldiers().get(i).getProbabilidadeSerPreso()) {

familia.addPreso(getCapoRegime().getSoldiers().get(i));

System.out.println("O " + getCapoRegime().getSoldiers().get(i).getNome() + "foi preso!");

getCapoRegime().getSoldiers().get(i).setEstaPreso(true);

}

}

}

}

@Override

public String toString() {

String texto;

texto = "Associates Armas: " + "\n";

for (Associate associate : associatesArmas) {

texto += "Associate: " + associate.getNome() + "\n";

};

return super.toString() + texto;

}

}

## Associate

package com.mycompany.projeto2;

public class Associate extends Pessoa {

//Construtores

// Construtor Mafioso

public Associate(Familia familia, String nome, int ccId, int lealdade, int musculo, int inteligencia, int estratega, int carisma, int probabilidadeSerPreso, boolean estaPreso, boolean linhagem, boolean informador) {

super(familia, nome, ccId, lealdade, musculo, inteligencia, estratega, carisma, probabilidadeSerPreso, estaPreso, false, informador);

}

@Override

public String toString() {

return super.toString(); //To change body of generated methods, choose Tools | Templates.

}

}

## Boss

package com.mycompany.projeto2;

public class Boss extends Mafioso {

private boolean hasConsiglieri = false;

private Familia familia;

private Familia familiaPartilhada;

private CapoRegime capoRegime;

private RandomAtributesGenerator randomAtributesGenerator = new RandomAtributesGenerator();

private int quantidadeDeNegocios, randomNumber;

private Negocio negocioPartilhado;

public Boss(Familia familia, String nome, int ccId, int lealdade, int musculo, int inteligencia, int estratega, int carisma, int probabilidadeSerPreso, boolean estaPreso, boolean linhagem, boolean informador) {

super(familia, nome, ccId, lealdade, musculo, inteligencia, estratega, carisma, probabilidadeSerPreso, estaPreso, true, false);

this.familia = familia;

}

//Recruta alguém para ser Soldier da família

public void RecrutaSoldier(String nome, CapoRegime capoRegime) {

//Codigo para recrutar um soldier

Soldier soldier = new Soldier(familia, nome, randomAtributesGenerator.generateRandomCCID(), randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), randomAtributesGenerator.generateProbabilidadeSerPreso(), false, true, false);

familia.addSoldier(soldier);

capoRegime.addSoldier(soldier);

System.out.println("Familia do soldier " + familia.getNome());

System.out.println(capoRegime.capoRegimeTeam());

System.out.println("Atualmente existem: " + familia.getSoldiers().size());

}

public Familia getFamiliaPartilhada() {

return familiaPartilhada;

}

public void setFamiliaPartilhada(Familia familiaPartilhada) {

this.familiaPartilhada = familiaPartilhada;

}

//Recruta alguém para o cargo de CapoRegime

public void RecrutaCapoRegime(String nome) {

//Codigo para recrutar um CapoRegime

familia.addCapoRegime(new CapoRegime(familia, nome, randomAtributesGenerator.generateRandomCCID(), randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), getProbabilidadeSerPreso(), false, true, false));

System.out.println("Familia do capoRegime " + familia.getNome());

System.out.println("Atualmente existem: " + familia.getCapoRegimes().size());

}

//Recruta alguém para o cargo de underboss

public void RecrutaUnderboss(String nome) {

//Codigo para recrutar um Underboss

Underboss underboss = new Underboss(familia, nome, randomAtributesGenerator.generateRandomCCID(), randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), getProbabilidadeSerPreso(), false, true, false);

System.out.println("Familia do underboss " + familia.getNome());

familia.setUnderboss(underboss);

}

//Vai nomear alguém para o cargo de consiglieri

public void NomearConsiglieri(String nome) {

//Codigo para nomear um consiglieri

hasConsiglieri = true;

Consiglieri consiglieri = new Consiglieri(familia, nome, randomAtributesGenerator.generateRandomCCID(), randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), getProbabilidadeSerPreso(), false, true, false);

System.out.println("Familia do consiglieri" + familia.getNome());

System.out.println("Consiglieri anterior: " + familia.getConsiglieri());

familia.setConsiglieri(consiglieri);

System.out.println("Consiglieri atual: " + familia.getConsiglieri());

}

//Método que vai gerar consoante o carisma do boss um certo valor de negocio para o caporegime associado

public void geraNegocio(CapoRegime capoRegime, Config config) {

if (getCarisma() >= 80) {

quantidadeDeNegocios = 5;

} else if (getCarisma() >= 50 && getCarisma() < 80) {

quantidadeDeNegocios = 3;

} else if (getCarisma() < 50) {

quantidadeDeNegocios = 1;

}

if (randomAtributesGenerator.generateProbabilidadeDeSerPartilhado() >= 1) {

int familiaIndex = randomAtributesGenerator.generateRandomIndex(config.getFamilias().size());

if (config.getFamilia(familiaIndex) == null) {

familiaPartilhada = null;

} else {

familiaPartilhada = config.getFamilia(familiaIndex);

if (familiaPartilhada.getNegocios().size() == 0) {

negocioPartilhado = null;

} else {

negocioPartilhado = familiaPartilhada.getNegocio(null, randomAtributesGenerator.generateRandomIndex(familiaPartilhada.getNegocios().size()), false);

}

}

}

if (familiaPartilhada != null && negocioPartilhado != null) {

familia.addNegocio(negocioPartilhado);

capoRegime.addNegocio(negocioPartilhado);

}

System.out.println("familiaPartilhada: " + familiaPartilhada);

System.out.println("negocioPartilhado: " + negocioPartilhado);

for (int i = 0; i < quantidadeDeNegocios; i++) {

randomNumber = randomAtributesGenerator.generateRandomNumber();

System.out.println("quantidadeDeNegocios " + quantidadeDeNegocios);

//Alterar criação de negocios

switch (randomNumber) {

case 0:

Casino casino = new Casino(familia, randomAtributesGenerator.generateRandomName("Casino"), randomAtributesGenerator.generateRandomValorBaseTributavel(), randomAtributesGenerator.generateRandomProbabilidadePoliciaAtuar(), false, capoRegime, randomAtributesGenerator.generateRandomRentabilidade(), randomAtributesGenerator.generateRandomValorBaseTributavel(), false, true);

familia.addNegocio(casino);

capoRegime.addNegocio(casino);

break;

case 1:

Armas armas = new Armas(familia, randomAtributesGenerator.generateRandomName("Armas"), randomAtributesGenerator.generateRandomValorBaseTributavel(), randomAtributesGenerator.generateRandomProbabilidadePoliciaAtuar(), true, capoRegime, randomAtributesGenerator.generateRandomRentabilidade(), randomAtributesGenerator.generateRandomValorBaseTributavel(), true, false);

familia.addNegocio(armas);

capoRegime.addNegocio(armas);

break;

case 2:

Droga droga = new Droga(familia, randomAtributesGenerator.generateRandomName("Droga"), randomAtributesGenerator.generateRandomValorBaseTributavel(), randomAtributesGenerator.generateRandomProbabilidadePoliciaAtuar(), true, capoRegime, randomAtributesGenerator.generateRandomRentabilidade(), randomAtributesGenerator.generateRandomValorBaseTributavel(), true, false);

familia.addNegocio(droga);

capoRegime.addNegocio(droga);

break;

case 3:

LavagemDinheiro lavagemDinheiro = new LavagemDinheiro(familia, randomAtributesGenerator.generateRandomName("LavagemDinheiro"), randomAtributesGenerator.generateRandomValorBaseTributavel(), randomAtributesGenerator.generateRandomProbabilidadePoliciaAtuar(), true, capoRegime, randomAtributesGenerator.generateRandomRentabilidade(), randomAtributesGenerator.generateRandomValorBaseTributavel(), false, true);

familia.addNegocio(lavagemDinheiro);

capoRegime.addNegocio(lavagemDinheiro);

break;

case 4:

Roubo roubo = new Roubo(familia, randomAtributesGenerator.generateRandomName("Roubo"), randomAtributesGenerator.generateRandomValorBaseTributavel(), randomAtributesGenerator.generateRandomProbabilidadePoliciaAtuar(), true, capoRegime, randomAtributesGenerator.generateRandomRentabilidade(), randomAtributesGenerator.generateRandomValorBaseTributavel(), true, true);

familia.addNegocio(roubo);

capoRegime.addNegocio(roubo);

break;

default:

// code block

}

}

}

@Override

public String toString() {

return super.toString(); //To change body of generated methods, choose Tools | Templates.

}

}

## CapoRegime

package com.mycompany.projeto2;

import java.util.ArrayList;

public class CapoRegime extends Mafioso {

private ArrayList<Soldier> soldiers;

private ArrayList<Negocio> negocios;

public CapoRegime(Familia familia, String nome, int ccId, int lealdade, int musculo, int inteligencia, int estratega, int carisma, int probabilidadeSerPreso, boolean estaPreso, boolean linhagem, boolean informador) {

super(familia, nome, ccId, lealdade, musculo, inteligencia, estratega, carisma, probabilidadeSerPreso, estaPreso, true, informador);

soldiers = new ArrayList<Soldier>();

negocios = new ArrayList<Negocio>();

}

public ArrayList<Negocio> getNegocios() {

return negocios;

}

//Método que adiciona um negocio à lista dos soldiers associados ao capoRegime

public void addNegocio(Negocio negocio) {

negocios.add(negocio);

}

//Método que retorna o Array com os Soldiers associados ao Caporegime

public ArrayList<Soldier> getSoldiers() {

return soldiers;

}

//Método que adiciona um soldier à lista dos soldiers associados ao capoRegime

public void addSoldier(Soldier soldier) {

soldiers.add(soldier);

}

//Mostra a equipa liderada pelo CapoRegime e com a sua equipa de Soldiers

public String capoRegimeTeam() {

String texto;

texto = "A equipa do capoRegime " + getNome() + " é " + getSoldiers();

return texto;

}

@Override

public String toString() {

return super.toString(); //To change body of generated methods, choose Tools | Templates.

}

}

## Casino

package com.mycompany.projeto2;

import java.util.ArrayList;

public class Casino extends Negocio {

private ArrayList<Associate> associatesCasino;

private RandomAtributesGenerator randomAtributesGenerator = new RandomAtributesGenerator();

private Familia familia;

public Casino(Familia familia, String nome, double ValorBaseTributavel, int ProbPolicia, boolean Policiavel, CapoRegime CapoRegime, double rentabilidade, double ValorAtualTributavel, boolean Musculado, boolean Inteligente) {

super(familia, nome, ValorBaseTributavel, ProbPolicia, false, CapoRegime, rentabilidade, ValorAtualTributavel, false, true);

this.familia = familia;

associatesCasino = new ArrayList<Associate>();

this.createAssociates();

}

//Cria x valor de associados para este negócio

@Override

public void createAssociates() {

int quantidadeDeAssociates = randomAtributesGenerator.generateRandomNumber();

System.out.println("quantidadeDeAssociates " + quantidadeDeAssociates);

for (int i = 0; i < quantidadeDeAssociates; i++) {

associatesCasino.add(new Associate(familia, randomAtributesGenerator.generateRandomName("Associate"), 1, randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), 0, false, true, false));

}

System.out.println("Associates Criados: " + associatesCasino);

}

//Obtem média da inteligencia da equipa de Soldiers e consoante esse valor é atribuido novos valores para a rentabilidade e valor base tributavel

@Override

public void AtualizaValoresNegocio() {

if (VerificaInteligencia() >= 50 && VerificaInteligencia() < 70) {

double novaRentabilidade = getRentabilidade() \* 1.5;

setRentabilidade(novaRentabilidade);

setValorAtualTributavel(novaRentabilidade \* 0.15);

} else if (VerificaInteligencia() >= 70 && VerificaInteligencia() < 90) {

double novaRentabilidade = getRentabilidade() \* 2;

setRentabilidade(novaRentabilidade);

setProbPolicia(getProbPolicia() - 5);

setValorAtualTributavel(novaRentabilidade \* 0.2);

} else if (VerificaInteligencia() >= 90) {

double novaRentabilidade = getRentabilidade() \* 3;

setRentabilidade(novaRentabilidade);

setProbPolicia(getProbPolicia() - 10);

setValorAtualTributavel(novaRentabilidade \* 0.3);

} else {

setProbPolicia(getProbPolicia() + 15);

}

}

@Override

public String toString() {

String texto;

texto = "Associates Casino: " + "\n";

for (Associate associate : associatesCasino) {

texto += "Associate: " + associate.getNome() + "\n";

};

return super.toString() + texto;

}

}

## Config

package com.mycompany.projeto2;

import java.util.ArrayList;

public class Config {

private ArrayList<Familia> familias;

private Familia familiaEscolhida, familiaPartilhada;

private boolean familiaFoiEscolhida = false;

public Config() {

familias = new ArrayList<Familia>();

}

public ArrayList<Familia> getFamilias() {

return familias;

}

public Familia getFamiliaEscolhida() {

return familiaEscolhida;

}

public void setFamiliaEscolhida(Familia familiaEscolhida) {

this.familiaEscolhida = familiaEscolhida;

}

public void setFamilias(ArrayList<Familia> familias) {

this.familias = familias;

}

public boolean isFamiliaFoiEscolhida() {

return familiaFoiEscolhida;

}

public void setFamiliaFoiEscolhida(boolean familiaFoiEscolhida) {

this.familiaFoiEscolhida = familiaFoiEscolhida;

}

//Método que adiciona familias da classe familia à lista

public void addFamilia(Familia familia) {

familias.add(familia);

}

public Familia getFamilia(int familiaId) {

if (!familiaFoiEscolhida) {

System.out.println("Familia nao foi escolhida");

for (Familia familia : familias) {

if (familiaId == familia.getFamiliaId()) {

familiaEscolhida = familia;

familiaFoiEscolhida = true;

}

}

return familiaEscolhida;

} else {

System.out.println("Familia foi escolhida");

if (familiaId != familiaEscolhida.getFamiliaId()) {

for (Familia familia : familias) {

if (familiaId == familia.getFamiliaId()) {

System.out.println("Familia partilhada certo");

familiaPartilhada = familia;

}

}

} else {

System.out.println("Familia partilhada erro");

return null;

}

return familiaPartilhada;

}

}

}

## Consiglieri

package com.mycompany.projeto2;

public class Consiglieri extends Mafioso {

private double aumentoValorBaseTributabel = 0;

private Familia familia;

private Negocio negocio;

public Consiglieri(Familia familia, String nome, int ccId, int lealdade, int musculo, int inteligencia, int estratega, int carisma, int probabilidadeSerPreso, boolean estaPreso, boolean linhagem, boolean informador) {

super(familia, nome, ccId, lealdade, musculo, inteligencia, estratega, carisma, probabilidadeSerPreso, estaPreso, linhagem, false);

this.familia = familia;

}

//Método que permite expandir um negócio implicando um aumento no valor tributavel

public void expandirNegocio(String nomeNegocio) {

negocio = familia.getNegocio(nomeNegocio, 0, true);

System.out.println("Negocio: " + negocio);

System.out.println("getEstratega() " + getEstratega());

if (getEstratega() > 50 && getEstratega() < 70) {

aumentoValorBaseTributabel = negocio.getValorBaseTributavel() + 500;

negocio.setValorBaseTributavel(aumentoValorBaseTributabel);

} else if (getEstratega() >= 70 && getEstratega() <90) {

aumentoValorBaseTributabel = negocio.getValorBaseTributavel() + 200;

negocio.setValorBaseTributavel(aumentoValorBaseTributabel);

} else if (getEstratega() >= 90) {

aumentoValorBaseTributabel = negocio.getValorBaseTributavel() + 100;

negocio.setValorBaseTributavel(aumentoValorBaseTributabel);

}

System.out.println(negocio.toString());

}

//Método onde conforme o nivel de Estratega do Consiglieri o custo fixo da familia aumenta ou diminui

public void mafiaSitdown() {

if (getEstratega() >= 80) {

familia.setCustoFixo(familia.getCustoFixo() / 2);

} else if (getEstratega() >= 50 && getEstratega() < 80) {

familia.setCustoFixo(familia.getCustoFixo() / 4);

} else if (getEstratega() < 50) {

familia.setCustoFixo(familia.getCustoFixo() \* 2);

}

}

@Override

public String toString() {

return super.toString(); //To change body of generated methods, choose Tools | Templates.

}

}

## Droga

package com.mycompany.projeto2;

import java.util.ArrayList;

public class Droga extends Negocio implements Policiavel {

private ArrayList<Associate> associatesDrogas;

private RandomAtributesGenerator randomAtributesGenerator = new RandomAtributesGenerator();

private Familia familia;

public Droga(Familia familia, String nome, double ValorBaseTributavel, int ProbPolicia, boolean Policiavel, CapoRegime CapoRegime, double rentabilidade, double ValorAtualTributavel, boolean Musculado, boolean Inteligente) {

super(familia, nome, ValorBaseTributavel, ProbPolicia, true, CapoRegime, rentabilidade, ValorAtualTributavel, true, false);

this.familia = familia;

associatesDrogas = new ArrayList<Associate>();

this.createAssociates();

}

//Cria x valor de associados para este negócio

@Override

public void createAssociates() {

int quantidadeDeAssociates = randomAtributesGenerator.generateRandomNumber();

System.out.println("quantidadeDeAssociates " + quantidadeDeAssociates);

for (int i = 0; i < quantidadeDeAssociates; i++) {

associatesDrogas.add(new Associate(familia, randomAtributesGenerator.generateRandomName("Associate"), 1, randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), 0, false, true, false));

}

System.out.println("Associates Criados: " + associatesDrogas);

}

//Obtem média do Musculo da equipa de Soldiers e consoante esse valor é atribuido novos valores para a rentabilidade, Valor base tributavel e probabilidade da policia atuar

@Override

public void AtualizaValoresNegocio() {

if (VerificaMusculo() >= 50 && VerificaMusculo() < 70) {

double novaRentabilidade = getRentabilidade() \* 2;

setRentabilidade(novaRentabilidade);

setValorAtualTributavel(novaRentabilidade\*0.2);

policiarNegocio();

} else if (VerificaMusculo() >= 70 && VerificaMusculo() < 90) {

double novaRentabilidade = getRentabilidade() \* 3;

setRentabilidade(novaRentabilidade);

setProbPolicia(getProbPolicia() - 5);

setValorAtualTributavel(novaRentabilidade\*0.3);

policiarNegocio();

} else if (VerificaMusculo() >= 90) {

double novaRentabilidade = getRentabilidade() \* 4;

setRentabilidade(novaRentabilidade);

setProbPolicia(getProbPolicia() - 10);

setValorAtualTributavel(novaRentabilidade\*0.4);

policiarNegocio();

} else {

setProbPolicia(getProbPolicia() + 15);

policiarNegocio();

}

}

//Método que analisa os valores de lealdade da equipa e consoante esse valor, o soldier pode se torna informador ou ir preso

@Override

public void policiarNegocio(){

RandomAtributesGenerator randomAtributesGenerator = new RandomAtributesGenerator();

int atuar = randomAtributesGenerator.generateRandomProbabilidadePoliciaAtuar();

int prender = randomAtributesGenerator.generateProbabilidadeSerPreso();

if(atuar <= getProbPolicia()){

System.out.println("A policia está a atuar no negocio.");

for (int i =0; i < getCapoRegime().getSoldiers().size(); i++){

if ( getCapoRegime().getSoldiers().get(i).getLealdade() < 15 ){

getCapoRegime().getSoldiers().get(i).setInformador(true);

}

else if (prender <= getCapoRegime().getSoldiers().get(i).getProbabilidadeSerPreso()){

familia.addPreso(getCapoRegime().getSoldiers().get(i));

System.out.println("O "+ getCapoRegime().getSoldiers().get(i).getNome() + "foi preso!");

getCapoRegime().getSoldiers().get(i).setEstaPreso(true);

}

}

}

}

@Override

public String toString() {

String texto;

texto = "Associates Drogas: " + "\n";

for (Associate associate : associatesDrogas) {

texto += "Associate: " + associate.getNome() + "\n";

};

return super.toString() + texto;

}

}

## Familia

package com.mycompany.projeto2;

import java.util.ArrayList;

public class Familia {

//Variáveis de Instância

private int familiaId;

private String nome;

private double riqueza, patrimonio, custoFixo;

private ArrayList<Mafioso> mafiosos;

private Boss boss;

private Consiglieri consiglieri;

private Underboss underboss;

private ArrayList<CapoRegime> caporegimes;

private ArrayList<Soldier> soldiers;

private ArrayList<Object> obituarioFamiliares;

private ArrayList<Negocio> negocios;

private Familia[] familiasComNegociosPartilhados;

private CapoRegime capoRegimeSelecionado;

private Negocio negocioSelecionado;

private ArrayList<Mafioso> presosDaFamilia;

//Construtor

public Familia(int familiaId, String nome, double riqueza, double patrimonio, double custoFixo) {

this.familiaId = familiaId;

this.nome = nome;

this.riqueza = riqueza;

this.patrimonio = patrimonio;

this.custoFixo = custoFixo;

mafiosos = new ArrayList<Mafioso>();

caporegimes = new ArrayList<CapoRegime>();

soldiers = new ArrayList<Soldier>();

obituarioFamiliares = new ArrayList<Object>();

negocios = new ArrayList<Negocio>();

familiasComNegociosPartilhados = new Familia[5];

presosDaFamilia = new ArrayList<Mafioso>();

}

//Método que retorn o array com os presos

public ArrayList<Mafioso> getPresosDaFamilia() {

return presosDaFamilia;

}

//Método que adiciona um membro da mafia à prisao

public void addPreso(Mafioso mafioso) {

presosDaFamilia.add(mafioso);

}

//Método que liberta um membro da mafia da prisao

public void libertarPreso(int mafiosoId) {

if (presosDaFamilia.size() > 0) {

for (Mafioso mafioso : presosDaFamilia) {

if (mafiosoId == mafioso.getCcId()) {

presosDaFamilia.remove(mafioso);

System.out.println("O mafioso " + mafioso.getNome() + " foi libertado");

mafioso.setEstaPreso(false);

} else {

System.out.println("O valor do ID não corresponde a nenhum preso!");

}

}

} else {

System.out.println("Não existe prisioneiros");

}

}

//Método que retorn os negócios ativos na família

public ArrayList<Negocio> getNegocios() {

return negocios;

}

//Método que adiciona objetos da classe Mafiosos à lista

public void addNegocio(Negocio negocio) {

negocios.add(negocio);

}

public Negocio getNegocio(String nomeNegocio, int negocioId, boolean isByName) {

//Redefine a variavel negocio para null

negocioSelecionado = null;

if (isByName) {

for (Negocio negocio : negocios) {

System.out.println("nomeNegocio " + nomeNegocio);

System.out.println("negocio.getNome() " + negocio.getNome());

if (nomeNegocio.equals(negocio.getNome())) {

negocioSelecionado = negocio;

}

}

} else {

return negocios.get(negocioId);

}

System.out.println("negocioSelecionado " + negocioSelecionado.getNome());

return negocioSelecionado;

}

//Método que devolve a lista de mafiosos na família

public ArrayList<Mafioso> getMafiosos() {

return mafiosos;

}

//Método que adiciona objetos da classe Mafiosos à lista

public void addMafioso(Mafioso mafioso) {

mafiosos.add(mafioso);

}

//Método que retorna o boss da família

public Boss getBoss() {

return boss;

}

//Método que atribui um boss à família

public void setBoss(Boss boss) {

this.boss = boss;

}

//Método que retorna o consiglieri da família

public Consiglieri getConsiglieri() {

return consiglieri;

}

//Método que atribui um Consiglieri à família

public void setConsiglieri(Consiglieri consiglieri) {

this.consiglieri = consiglieri;

}

//Método que retorna o underboss da família

public Underboss getUnderboss() {

return underboss;

}

//Método que atribui um underboss à familia

public void setUnderboss(Underboss underboss) {

this.underboss = underboss;

}

//Método que devolve a lista com os caporegime da família

public ArrayList<CapoRegime> getCapoRegimes() {

return caporegimes;

}

//Método que adiciona um caporegime à lista dos caporegimes da família

public void addCapoRegime(CapoRegime caporegime) {

caporegimes.add(caporegime);

}

//Método que remove um capoRegime da lista dos caporegimes da família

public void removeCapoRegime(String capoRegimeNome) {

caporegimes.removeIf(capoRegime -> capoRegime.getNome().equals(capoRegimeNome));

}

//Método que retorna o caporegime na qual o seu ccID seja igual ao valor do ID introduzido como parâmetro

public CapoRegime getCapoRegime(int capoRegimeId) {

//Redefine a variavel capoRegime para null

capoRegimeSelecionado = null;

for (CapoRegime capoRegime : caporegimes) {

if (capoRegimeId == capoRegime.getCcId()) {

capoRegimeSelecionado = capoRegime;

} else {

capoRegimeSelecionado = null;

}

}

return capoRegimeSelecionado;

}

//Método que devolve a lista com os soldiers da família

public ArrayList<Soldier> getSoldiers() {

return soldiers;

}

//Método que adiciona um soldier à lista dos soldiers da família

public void addSoldier(Soldier soldier) {

soldiers.add(soldier);

}

//Método que remove um soldier da lista dos soldiers da família

public void removeSoldier(String soldierName) {

soldiers.removeIf(soldier -> soldier.getNome().equals(soldierName));

}

//Método que retorna o nome da família

public String getNome() {

return nome;

}

//Método que retorna a riqueza da familia

public double getRiqueza() {

return riqueza;

}

//Metodo que retorna o patrimonio da familia

public double getPatrimonio() {

return patrimonio;

}

//Retorna o ID da familia

public int getFamiliaId() {

return familiaId;

}

//Método que atribui um novo valor à riqueza

public void setRiqueza(double riqueza) {

this.riqueza = riqueza;

}

//Atribui um novo valor ao patrimonio

public void setPatrimonio(double patrimonio) {

this.patrimonio = patrimonio;

}

//Retorna os membros que passaram para o obituario da familia

public ArrayList<Object> getObituarioFamiliares() {

return obituarioFamiliares;

}

//Método que adiciona um soldier à lista dos soldiers da família

public void addFamiliarObituario(Object familiar) {

obituarioFamiliares.add(familiar);

}

//Retorn o custo fixo da familia

public double getCustoFixo() {

return custoFixo;

}

//Atribui um novo valor ao tal custo fixo

public void setCustoFixo(double custoFixo) {

this.custoFixo = custoFixo;

}

//Override do método toString

@Override

public String toString() {

String texto;

texto = "\n" + "Id da Familia: " + familiaId + "\n";

texto += "Nome da Familia: " + nome + "\n";

texto += "Riqueza da Familia: " + riqueza + "\n";

texto += "Patrimonio da Familia: " + patrimonio;

return texto;

}

}

## LavagemDinheiro

package com.mycompany.projeto2;

import java.util.ArrayList;

public class LavagemDinheiro extends Negocio implements Policiavel {

private ArrayList<Associate> associatesLavagemDeDinheiro;

private RandomAtributesGenerator randomAtributesGenerator = new RandomAtributesGenerator();

private Familia familia;

public LavagemDinheiro(Familia familia, String nome, double ValorBaseTributavel, int ProbPolicia, boolean Policiavel, CapoRegime CapoRegime, double rentabilidade, double ValorAtualTributavel, boolean Musculado, boolean Inteligente) {

super(familia, nome, ValorBaseTributavel, ProbPolicia, true, CapoRegime, rentabilidade, ValorAtualTributavel, false, true);

this.familia = familia;

associatesLavagemDeDinheiro = new ArrayList<Associate>();

this.createAssociates();

}

//Cria x valor de associados para este negócio

@Override

public void createAssociates() {

int quantidadeDeAssociates = randomAtributesGenerator.generateRandomNumber();

System.out.println("quantidadeDeAssociates " + quantidadeDeAssociates);

for (int i = 0; i < quantidadeDeAssociates; i++) {

associatesLavagemDeDinheiro.add(new Associate(familia, randomAtributesGenerator.generateRandomName("Associate"), 1, randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), 0, false, true, false));

}

System.out.println("Associates Criados: " + associatesLavagemDeDinheiro);

}

//Obtem média da Inteligência da equipa de Soldiers e consoante esse valor é atribuido novos valores para a rentabilidade, Valor base tributavel e probabilidade da policia atuar

@Override

public void AtualizaValoresNegocio() {

if (VerificaInteligencia() >= 50 && VerificaInteligencia() < 70) {

double novaRentabilidade = getRentabilidade() \* 1.5;

setRentabilidade(novaRentabilidade);

setValorAtualTributavel(novaRentabilidade \* 0.15);

policiarNegocio();

}

if (VerificaInteligencia() >= 70 && VerificaInteligencia() < 90) {

double novaRentabilidade = getRentabilidade() \* 2;

setRentabilidade(novaRentabilidade);

setProbPolicia(getProbPolicia() - 5);

setValorAtualTributavel(novaRentabilidade \* 0.2);

policiarNegocio();

} else if (VerificaInteligencia() >= 90) {

double novaRentabilidade = getRentabilidade() \* 3;

setRentabilidade(novaRentabilidade);

setProbPolicia(getProbPolicia() - 10);

setValorAtualTributavel(novaRentabilidade \* 0.3);

policiarNegocio();

} else {

setProbPolicia(getProbPolicia() + 15);

policiarNegocio();

}

}

//Método que analisa os valores de lealdade da equipa e consoante esse valor, o soldier pode se torna informador ou ir preso

@Override

public void policiarNegocio() {

RandomAtributesGenerator randomAtributesGenerator = new RandomAtributesGenerator();

int atuar = randomAtributesGenerator.generateRandomProbabilidadePoliciaAtuar();

int prender = randomAtributesGenerator.generateProbabilidadeSerPreso();

if (atuar <= getProbPolicia()) {

System.out.println("A policia está a atuar no negocio.");

for (int i = 0; i < getCapoRegime().getSoldiers().size(); i++) {

if (getCapoRegime().getSoldiers().get(i).getLealdade() < 15) {

getCapoRegime().getSoldiers().get(i).setInformador(true);

} else if (prender <= getCapoRegime().getSoldiers().get(i).getProbabilidadeSerPreso()) {

familia.addPreso(getCapoRegime().getSoldiers().get(i));

System.out.println("O " + getCapoRegime().getSoldiers().get(i).getNome() + "foi preso!");

getCapoRegime().getSoldiers().get(i).setEstaPreso(true);

}

}

}

}

@Override

public String toString() {

String texto;

texto = "Associates Lavagem Dinheiro: " + "\n";

for (Associate associate : associatesLavagemDeDinheiro) {

texto += "Associate: " + associate.getNome() + "\n";

};

return super.toString() + texto;

}

}

## Mafioso

package com.mycompany.projeto2;

public abstract class Mafioso extends Pessoa {

//Construtores

// Construtor Mafioso

public Mafioso(Familia familia, String nome, int ccId, int lealdade, int musculo, int inteligencia, int estratega, int carisma, int probabilidadeSerPreso, boolean estaPreso, boolean linhagem, boolean informador) {

super(familia, nome, ccId, lealdade, musculo, inteligencia, estratega, carisma, probabilidadeSerPreso, estaPreso, true, informador);

}

@Override

public String toString() {

return super.toString(); //To change body of generated methods, choose Tools | Templates.

}

}

## Negocio

package com.mycompany.projeto2;

import java.util.ArrayList;

public abstract class Negocio {

private Familia familia;

private String nome;

private double ValorBaseTributavel, rentabilidade, ValorAtualTributavel = ValorBaseTributavel;

private int ProbPolicia;

private boolean Policiavel, Musculado, Inteligente;

private CapoRegime CapoRegime;

private RandomAtributesGenerator randomAtributesGenerator = new RandomAtributesGenerator();

private ArrayList<Associate> associates;

//Constutor de negocio

public Negocio(Familia familia, String nome, double ValorBaseTributavel, int ProbPolicia, boolean Policiavel, CapoRegime CapoRegime, double rentabilidade, double ValorAtualTributavel, boolean Musculado, boolean Inteligente) {

this.familia = familia;

this.nome = nome;

this.ValorBaseTributavel = ValorBaseTributavel;

this.ProbPolicia = ProbPolicia;

this.Policiavel = Policiavel;

this.CapoRegime = CapoRegime;

this.rentabilidade = rentabilidade;

this.ValorAtualTributavel = ValorAtualTributavel;

this.Musculado = Musculado;

this.Inteligente = Inteligente;

associates = new ArrayList<Associate>();

}

//Metodo que cria associados para se enquadrarem nos negocios

public void createAssociates() {

int quantidadeDeAssociates = randomAtributesGenerator.generateRandomNumber();

System.out.println("quantidadeDeAssociates " + quantidadeDeAssociates);

for (int i = 0; i < quantidadeDeAssociates; i++) {

associates.add(new Associate(familia, randomAtributesGenerator.generateRandomName("Associate"), 1, randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), 0, false, true, false));

}

System.out.println("Associates Criados: " + associates);

}

//Método que retorna os associados

public ArrayList<Associate> getAssociates() {

return associates;

}

//Retorna a rentabilidade do negocio

public double getRentabilidade() {

return rentabilidade;

}

//Atribui um novo valor à rentabilidade

public void setRentabilidade(double aumento) {

this.rentabilidade = aumento;

}

//Retorna o caporegime associado ao negocio

public CapoRegime getCapoRegime() {

return CapoRegime;

}

//Retorna o valor atual tributavel do negócio

public double getValorAtualTributavel() {

return ValorAtualTributavel;

}

//Atribui um novo valor atual tributavel

public void setValorAtualTributavel(double tributo) {

this.ValorAtualTributavel += tributo;

}

//Retorna se o negocio é ou não Musculado

public boolean isMusculado() {

return Musculado;

}

//Retorna se o negocio é ou não Inteligente

public boolean isInteligente() {

return Inteligente;

}

//Retorna o nome do negocio

public String getNome() {

return nome;

}

//Retorna o valor base tributavel

public double getValorBaseTributavel() {

return ValorBaseTributavel;

}

//Atribui um novo valor base tributavel ao negocio

public void setValorBaseTributavel(double NovoValorBase) {

this.ValorBaseTributavel = NovoValorBase;

}

//Retorna se o negocio é ou não policiavel

public boolean VerificaPoliciavel() {

return Policiavel;

}

//Retorna a probabilidade da policia atuar

public int getProbPolicia() {

return ProbPolicia;

}

//Atribui um novo valor da probabilidade da policia atuar

public void setProbPolicia(int prob) {

this.ProbPolicia = prob;

}

//Metodo abstrato defenido em cada subclasse de negocio

public abstract void AtualizaValoresNegocio();

//Retorna um valor da média obtida da Inteligencia da equipa a atuar no negocio

public double VerificaInteligencia() {

double media = 0;

double soma = 0;

for (int i = 0; i < CapoRegime.getSoldiers().size(); i++) {

soma += CapoRegime.getSoldiers().get(i).getInteligencia();

}

media = soma / CapoRegime.getSoldiers().size();

return media;

}

//Retorna o valor médio do Musculo relativo à equipa associada ao negocio

public double VerificaMusculo() {

double media = 0;

double soma = 0;

for (int i = 0; i < CapoRegime.getSoldiers().size(); i++) {

soma += CapoRegime.getSoldiers().get(i).getMusculo();

}

media = soma / CapoRegime.getSoldiers().size();

return media;

}

@Override

public String toString() {

String texto;

texto = "Nome do negocio: " + nome + "\n";

texto += "Valor Base Tributavel: " + ValorBaseTributavel + "\n";

texto += "Probabilidade da Policia: " + ProbPolicia + "\n";

texto += "Policiavel: " + Policiavel + "\n";

texto += "CapoRegime: " + CapoRegime.getNome() + "\n";

texto += "Rentabilidade: " + rentabilidade + "\n";

return texto;

}

}

## Pessoa

package com.mycompany.projeto2;

public abstract class Pessoa {

//Variaveis de Instancia

private Familia familia;

private String nome;

private int ccId, lealdade, musculo, inteligencia, estratega, carisma, probabilidadeSerPreso = 0;

private boolean estaPreso, linhagem, informador;

public Pessoa(Familia familia, String nome, int ccId, int lealdade, int musculo, int inteligencia, int estratega, int carisma, int probabilidadeSerPreso, boolean estaPreso, boolean linhagem, boolean informador) {

this.familia = familia;

this.nome = nome;

this.ccId = ccId;

this.lealdade = lealdade;

this.musculo = musculo;

this.inteligencia = inteligencia;

this.estratega = estratega;

this.carisma = carisma;

this.probabilidadeSerPreso = probabilidadeSerPreso;

this.carisma = carisma;

this.estaPreso = estaPreso;

this.linhagem = linhagem;

this.informador=informador;

}

//Getters e setters necessários para a classe

public Familia getFamilia() {

return familia;

}

public String getNome() {

return nome;

}

public int getCcId() {

return ccId;

}

public int getLealdade() {

return lealdade;

}

public int getMusculo() {

return musculo;

}

public int getInteligencia() {

return inteligencia;

}

public int getEstratega() {

return estratega;

}

public int getCarisma() {

return carisma;

}

public int getProbabilidadeSerPreso() {

return probabilidadeSerPreso;

}

public void setEstaPreso(boolean estaPreso) {

this.estaPreso = estaPreso;

}

public boolean isInformador(){

return informador;

}

public void setInformador(boolean informador){

this.informador=informador;

}

@Override

public String toString() {

String texto;

texto = "Nome: " + nome + "\n";

texto += "ccId:" + ccId + "\n";

texto += "Lealdade:" + lealdade + "\n";

texto += "Musculo:" + musculo + "\n";

texto += "Inteligencia:" + inteligencia + "\n";

texto += "Estratega:" + estratega + "\n";

texto += "Carisma:" + carisma + "\n";

texto += "ProbabilidadeSerPreso:" + probabilidadeSerPreso + "\n";

texto += "EstaPreso:" + estaPreso + "\n";

texto += "Linhagem:" + linhagem + "\n";

texto += "Informador: " + informador + "\n";

return texto;

}

}

## Policiavel

package com.mycompany.projeto2;

//Se se verificar a condição de policiar um negocio, este faz a analise da lealdade de cada soldier da equipa e consoante esse valor pode se tornar informador ou preso

public interface Policiavel {

public void policiarNegocio();

}

## RandomAtributesGenerator

package com.mycompany.projeto2;

import java.util.Random;

public class RandomAtributesGenerator {

private int randomIndex, probabilidadeDeSerPartilhado, randomCCID, randomLealdade, randomInteligencia, randomMusculo, randomEstrategia, randomCarisma, randomNumber, randomValorBaseTributavel, randomProbabPoliciaAtuar, randomRentabilidade, randomSerPreso, minValue = 100, maxValue = 1000;

private String randomName;

// create instance of Random class

Random rand = new Random();

public RandomAtributesGenerator() {

}

//Diversos randoms sendo obtidos consoante cada um valor ou uma string

public String generateRandomName(String name) {

randomName = name + rand.nextInt(1000);

System.out.println("Random Name" + randomName);

return randomName;

}

public int generateRandomIndex(int size) {

randomIndex = rand.nextInt(size);

System.out.println("RandomIndex: " + randomIndex);

return randomIndex;

}

public int generateRandomCCID() {

randomCCID = rand.nextInt(99999999-9999999);

return randomCCID;

}

public int generateRandomValorBaseTributavel() {

randomValorBaseTributavel = rand.nextInt(maxValue - minValue) + minValue;

return randomValorBaseTributavel;

}

public int generateRandomRentabilidade() {

randomRentabilidade = rand.nextInt(maxValue - minValue) + minValue;

return randomRentabilidade;

}

public int generateRandomProbabilidadePoliciaAtuar() {

randomProbabPoliciaAtuar = rand.nextInt(100);

return randomProbabPoliciaAtuar;

}

public int generateProbabilidadeDeSerPartilhado() {

probabilidadeDeSerPartilhado = rand.nextInt(100);

return probabilidadeDeSerPartilhado;

}

public int generateRandomLealdade() {

randomLealdade = rand.nextInt(100);

return randomLealdade;

}

public int generateRandomNumber() {

randomNumber = rand.nextInt(5);

return randomNumber;

}

public int generateProbabilidadeSerPreso() {

randomSerPreso = rand.nextInt(61) + 10;

return randomSerPreso;

}

public int generateRandomInteligencia() {

randomInteligencia = rand.nextInt(100);

return randomInteligencia;

}

public int generateRandomMusculo() {

randomMusculo = rand.nextInt(100);

return randomMusculo;

}

public int generateRandomEstrategia() {

randomEstrategia = rand.nextInt(100);

return randomEstrategia;

}

public int generateRandomCarisma() {

randomCarisma = rand.nextInt(100);

return randomCarisma;

}

}

## Roubo

package com.mycompany.projeto2;

import java.util.ArrayList;

public class Roubo extends Negocio implements Policiavel {

private ArrayList<Associate> associatesRoubo;

private RandomAtributesGenerator randomAtributesGenerator = new RandomAtributesGenerator();

private Familia familia;

public Roubo(Familia familia, String nome, double ValorBaseTributavel, int ProbPolicia, boolean Policiavel, CapoRegime CapoRegime, double rentabilidade, double ValorAtualTributavel, boolean Musculado, boolean Inteligente) {

super(familia, nome, ValorBaseTributavel, ProbPolicia, true, CapoRegime, rentabilidade, ValorAtualTributavel, true, true);

this.familia = familia;

associatesRoubo = new ArrayList<Associate>();

this.createAssociates();

}

//Cria x valor de associados para este negócio

@Override

public void createAssociates() {

int quantidadeDeAssociates = randomAtributesGenerator.generateRandomNumber();

System.out.println("quantidadeDeAssociates " + quantidadeDeAssociates);

for (int i = 0; i < quantidadeDeAssociates; i++) {

associatesRoubo.add(new Associate(familia, randomAtributesGenerator.generateRandomName("Associate"), 1, randomAtributesGenerator.generateRandomLealdade(), randomAtributesGenerator.generateRandomMusculo(), randomAtributesGenerator.generateRandomInteligencia(), randomAtributesGenerator.generateRandomEstrategia(), randomAtributesGenerator.generateRandomCarisma(), 0, false, true, false));

}

System.out.println("Associates Criados: " + associatesRoubo);

}

//Obtem média do Musculo e da Inteligência da equipa de Soldiers e consoante esse valor é atribuido novos valores para a rentabilidade, Valor base tributavel e probabilidade da policia atuar

@Override

public void AtualizaValoresNegocio() {

if ((VerificaMusculo() >= 50 && VerificaMusculo() < 70) && (VerificaInteligencia() >= 50 && VerificaInteligencia() < 70)) {

double novaRentabilidade = getRentabilidade() \* 2;

setRentabilidade(novaRentabilidade);

setValorAtualTributavel(novaRentabilidade \* 0.2);

policiarNegocio();

} else if ((VerificaMusculo() >= 70 && VerificaMusculo() < 90) && (VerificaInteligencia() >= 70) && VerificaInteligencia() < 90) {

double novaRentabilidade = getRentabilidade() \* 3;

setRentabilidade(novaRentabilidade);

setProbPolicia(getProbPolicia() - 5);

setValorAtualTributavel(novaRentabilidade \* 0.3);

policiarNegocio();

} else if (VerificaMusculo() >= 90 && VerificaInteligencia() >= 90) {

double novaRentabilidade = getRentabilidade() \* 4;

setRentabilidade(novaRentabilidade);

setProbPolicia(getProbPolicia() - 15);

setValorAtualTributavel(novaRentabilidade \* 0.4);

policiarNegocio();

} else {

setProbPolicia(getProbPolicia() + 15);

policiarNegocio();

}

}

//Método que analisa os valores de lealdade da equipa e consoante esse valor, o soldier pode se torna informador ou ir preso

@Override

public void policiarNegocio() {

RandomAtributesGenerator randomAtributesGenerator = new RandomAtributesGenerator();

int atuar = randomAtributesGenerator.generateRandomProbabilidadePoliciaAtuar();

int prender = randomAtributesGenerator.generateProbabilidadeSerPreso();

System.out.println("Probabilidade de ser Preso " + prender);

if (atuar <= getProbPolicia()) {

System.out.println("A policia está a atuar no negocio.");

for (int i = 0; i < getCapoRegime().getSoldiers().size(); i++) {

if (getCapoRegime().getSoldiers().get(i).getLealdade() < 15) {

getCapoRegime().getSoldiers().get(i).setInformador(true);

System.out.println("O " + getCapoRegime().getSoldiers().get(i).getNome() + " é agora um informador");

} else if (prender <= getCapoRegime().getSoldiers().get(i).getProbabilidadeSerPreso()) {

familia.addPreso(getCapoRegime().getSoldiers().get(i));//Adiciona o soldier à prisao

getCapoRegime().getSoldiers().get(i).getFamilia().removeSoldier(getCapoRegime().getSoldiers().get(i).getNome());//Remove o soldier da familia

System.out.println("O " + getCapoRegime().getSoldiers().get(i).getNome() + " foi preso!");

getCapoRegime().getSoldiers().get(i).setEstaPreso(true);

}

}

}

}

@Override

public String toString() {

String texto;

texto = "Associates Roubo: " + "\n";

for (Associate associate : associatesRoubo) {

texto += "Associate: " + associate.getNome() + "\n";

};

return super.toString() + texto;

}

}

## Soldier

package com.mycompany.projeto2;

public class Soldier extends Mafioso {

public Soldier(Familia familia, String nome, int ccId, int lealdade, int musculo, int inteligencia, int estratega, int carisma, int probabilidadeSerPreso, boolean estaPreso, boolean linhagem, boolean informador) {

super(familia, nome, ccId, lealdade, musculo, inteligencia, estratega, carisma, probabilidadeSerPreso, estaPreso, true, informador);

}

@Override

public String toString() {

return super.toString(); //To change body of generated methods, choose Tools | Templates.

}

}

## Underboss

package com.mycompany.projeto2;

import java.util.ArrayList;

public class Underboss extends Mafioso {

private ArrayList<Soldier> soldiers;

private ArrayList<CapoRegime> capoRegimes;

private Soldier soldier, traidorSoldier, MelhorSoldier;

private CapoRegime capoRegime, traidorCapoRegime;

public Underboss(Familia familia, String nome, int ccId, int lealdade, int musculo, int inteligencia, int estratega, int carisma, int probabilidadeSerPreso, boolean estaPreso, boolean linhagem, boolean informador) {

super(familia, nome, ccId, lealdade, musculo, inteligencia, estratega, carisma, probabilidadeSerPreso, estaPreso, true, false);

soldiers = new ArrayList<Soldier>();

capoRegimes = new ArrayList<CapoRegime>();

}

//Loyalty test sendo que caso seja adicionado alguem ao obituario deve ser feito um suposto pagamento aos familiares deste sendo entao retirado um certo valor à riqueza da familia dos mafiosos

public void loyaltyTest(Familia familia) {

soldiers = familia.getSoldiers();

capoRegimes = familia.getCapoRegimes();

for (int i = 0; i < soldiers.size(); i++) {

soldier = soldiers.get(i);

if (soldier.getLealdade() < 20 || soldier.isInformador() == true) {

traidorSoldier = soldiers.get(i);

System.out.println("TraidorSoldier = " + traidorSoldier.getNome());

familia.removeSoldier(traidorSoldier.getNome());

familia.addFamiliarObituario(traidorSoldier);

System.out.println("O soldier " + traidorSoldier.getNome() + " foi removido da familia");

familia.setRiqueza(familia.getRiqueza() \* 0.97); //pagamento aos tais familiares do que sofreu "desaparecimento misterioso"

}

}

for (int i = 0; i < capoRegimes.size(); i++) {

capoRegime = capoRegimes.get(i);

if (capoRegime.getLealdade() < 20) {

traidorCapoRegime = capoRegimes.get(i);

System.out.println("TraidorCapo = " + traidorCapoRegime.getNome());

familia.removeCapoRegime(traidorCapoRegime.getNome());

familia.addFamiliarObituario(traidorCapoRegime);

System.out.println("O capoRegime " + traidorCapoRegime.getNome() + " foi removido da familia");

familia.setRiqueza(familia.getRiqueza() \* 0.97); //pagamento aos tais familiares do que sofreu "desaparecimento misterioso"

}

}

if (traidorSoldier == null && traidorCapoRegime == null) {

System.out.println("Não existe triadores!");

}

}

@Override

public String toString() {

return super.toString(); //To change body of generated methods, choose Tools | Templates.

}

}