JogoTabuleiro – src – model – Casa

```
package model;

public abstract class Casa {
    public int posicao;
    public abstract int getIncremento();
}
```

JogoTabuleiro – src – model – CasaAzar

```
package model;

import java.util.Random;

public class CasaAzar extends Casa {
    private int incremento;

    public CasaAzar() {

        Random rand = new Random();
        this.incremento = rand.nextInt(6) + 1;

    }
    public String getDescricao() {
        String retorno  = "pouco AZAR";
        if (incremento > 4) {
            retorno = "Muito AZAR";
        } else {
            if (incremento > 2) {
                retorno = "AZAR";
            }
        }
        return retorno;
    }
    public int getIncremento() {
        return incremento * -1;
    }
    @Override
    public String toString(){
        return "Casa de Azar: retorne "+incremento+" posições";
    }

}
```

JogoTabuleiro – src – model – CasaNeutra

```
package model;

public class CasaNeutra extends Casa {
```

```
    @Override
    public int getIncremento() {
        return 0;
    }
}
```

JogoTabuleiro – src – model – CasaSorte

```
package model;

import java.util.Random;

public class CasaSorte extends Casa{
    private int incremento;

    public CasaSorte() {

        Random rand = new Random();
        this.incremento = rand.nextInt(6) + 1;
    }

    public int getIncremento() {
        return incremento;
    }
    @Override
    public String toString(){
        return "Casa de Sorte: avance "+incremento+" posições";
    }
}
```

JogoTabuleiro – src – model – Dado

```
package model;

import java.util.Random;

public class Dado {
    private int lados = 6;

    public Dado() {
    }

    public Dado(int lados) {
        this.lados = lados;
    }

    public int jogar() {
        Random rand = new Random();
```

```
        return rand.nextInt(this.lados) + 1;
    }

}
```

JogoTabuleiro – src – model – Jogador

```
package model;

import java.util.Observable;

public class Jogador extends Observable {
    private String nome;
    private int posicaoAtual;

    public Jogador(String nome) {
        this.nome = nome;
        this.posicaoAtual = -1;
    }

    public String getNome() {
        return nome;
    }

    public int getPosicaoAtual() {
        return posicaoAtual;
    }
    public void avanca(int qtd) {
        this.posicaoAtual += qtd;
        this.setChanged();
        this.notifyObservers();
    }

    @Override
    public String toString() {

        return "Jogador: "+nome;
    }



}
```

JogoTabuleiro – src – model – MuitosJogadoresException

```
package model;

public class MuitosJogadoresException extends Exception{
    @Override
    public String getMessage() {
```

```
        return "O tabuleiro só aceita no máximo 6 jogadores";
    }
```

JogoTabuleiro – src – model – NovaRegraDoJogo

```
package model;

public class NovaRegraDoJogo extends RegraDoJogo{

    @Override
    public boolean alguemGanhou(Tabuleiro tab) {
        boolean retorno = super.alguemGanhou(tab);
        if (!retorno) {
            for (Jogador jog : tab.getJogadores()) {

                // troquei pelo super()
                //if (jog.getPosicaoAtual() >= tab.getQtdCasas()) {
                //    retorno = true;
                //    ganhador  =jog;
                //}
                //if (jog.getPosicaoAtual() % 3 == 0) {
                if (jog.getPosicaoAtual()  == 8) {
                    retorno = true;
                    ganhador = jog;
                }
            }
        }
        return retorno;
    }
}
```

JogoTabuleiro – src – model – RegraDoJogo

```
package model;

public class RegraDoJogo {
    protected Jogador ganhador;

    public boolean alguemGanhou(Tabuleiro tab) {
        boolean retorno = false;
        for(Jogador jog : tab.getJogadores()) {
            if (jog.getPosicaoAtual() >= tab.getQtdCasas()) {
                retorno = true;
                ganhador  =jog;
            }
        }
        return retorno;
    }
    public Jogador quemGanhou() {
```

```
        return ganhador;
    }
}
```

JogoTabuleiro – src – model – Tabuleiro

```
package model;

import java.util.ArrayList;
import java.util.List;
import java.util.Random;

public class Tabuleiro  {
    private Casa[] casas;
    private List<Jogador> jogadores;
    //                  10          20          30
    public Tabuleiro(int qtdCasas, int percSorte, int percAzar) {
        this.jogadores = new ArrayList<>();
        this.casas = new Casa[qtdCasas];
        int qtdSorte = qtdCasas * percSorte / 100;
        int qtdAzar = qtdCasas * percAzar / 100;
        int i = 0;
        for ( ; i < (qtdCasas - qtdSorte - qtdAzar);  i++) {
            casas[i] = new CasaNeutra();

        }
        for (int x = 0; x < qtdSorte; x++) {
            casas[i] = new CasaSorte();
            i++;
        }
        for (int x = 0; x < qtdAzar; x++) {
            casas[i] = new CasaAzar();
            i++;
        }
        embaralhaCasas();

    }
    private void embaralhaCasas() {
        Random rand = new Random();
        for (int x = 0; x < this.casas.length; x++) {
            int pos1 = rand.nextInt(this.casas.length);
            int pos2 = rand.nextInt(this.casas.length);
            Casa temp = this.casas[pos1];
            this.casas[pos1] = this.casas[pos2];
            this.casas[pos2] = temp;
        }
        for (int x = 0; x < casas.length; x++) {
            this.casas[x].posicao = x + 1;
        }
    }
```

```
    public Casa getCasa(int pos) {
        return this.casas[pos];
    }
    public Casa getCasaOcupada(int pos) {
        Casa retorno = null;
        for(Jogador jog : jogadores) {
            if (jog.getPosicaoAtual() == pos) {
                retorno = this.casas[pos];
            }
        }
        return retorno;
    }
    public String getJogadoresCasa(int pos) {
        String retorno = "";
        for(Jogador jog : jogadores) {
            if (jog.getPosicaoAtual() == pos) {
                retorno += jog.getNome()+", ";
            }
        }
        return retorno;
    }
    public List<Jogador> getJogadores() {
        return jogadores;
    }
    public void addJogador(Jogador jog) throws MuitosJogadoresException {
        if (this.jogadores.size() == 6) {
            throw new MuitosJogadoresException();
        } else {
            this.jogadores.add(jog);
        }
    }
    public int getQtdCasas() {
        return this.casas.length;
    }
}
```

JogoTabuleiro – src – view – CasaView

```
package view;

import model.Casa;
import model.CasaAzar;
import model.CasaSorte;

public class CasaView {
    public static void desenhaCasa(Casa casa, String txtJogadores) {
        System.out.println(preenche("",'-', 25));
        if (casa instanceof CasaSorte) {
            System.out.println(preenche(" Casa de SORTE ",' ',25));
            System.out.println(preenche(" Avance "+casa.getIncremento()+"
```

```
casas.",' ',25));
        } else {
            if (casa instanceof CasaAzar) {
                //CasaAzar temp  = (CasaAzar)casa;
                //System.out.println(preenche("Casa de "+temp.getDescricao(),' ',25));

                System.out.println(preenche("Casa de
"+((CasaAzar)casa).getDescricao(),' ',25));
                System.out.println(preenche(" Volte "+casa.getIncremento()+"
casas.",' ',25));
            } else{
                System.out.println(preenche(" Casa Neutra    ",' ',25));
            }

        }
        if (txtJogadores != "") {
            System.out.println(preenche("jogadores:",' ',25));
            System.out.println(preenche(txtJogadores,' ',25));
        }
        System.out.println(preenche("posição:"+casa.posicao,'-', 25));
        //System.out.println(preenche("",'-', 25));

    }
    private static String preenche(String texto, char preenchimento, int tamanho)
{
        //for(int t = texto.length(); t <= tamanho; t++) {
        //    texto += preenchimento;
        //}
        while (texto.length()<tamanho) {
            texto += preenchimento;
        }
        return "|"+texto+"|";
    }
}
```

JogoTabuleiro – src – view – JogoView

```
package view;

import model.Jogador;

import java.util.InputMismatchException;
import java.util.Observable;
import java.util.Observer;
import java.util.Scanner;

public class JogoView implements Observer {

    public static int intQtdJogadores(int min, int max) {
        System.out.println("quantos jogadores teremos ?");
```

```
        int qtd = 0;
        do {
            Scanner scanner = new Scanner(System.in);
            try {
                qtd = scanner.nextInt();
            } catch (InputMismatchException e) {
                System.out.println("valor inválido.");
                //scanner.close();
            }
        } while (qtd < min || qtd> max);
        return qtd;
    }
    public static String InformeJogador(int idx) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Informe o nome do Jogador "+idx+":");
        String nome  = scanner.nextLine();
        return nome;


    }

    public static void mostraJogadorAtual(Jogador jog) {

        System.out.println("Agora é a vez de "+jog.getNome());
    }

    public static void continuar() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("pressione enter para continuar");
        String nome  = scanner.nextLine();



    }

    @Override
    public void update(Observable o, Object arg) {

System.out.println("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@");
        System.out.println("    "+((Jogador)o).getNome()+"
"+((Jogador)o).getPosicaoAtual());

System.out.println("@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@");
    }
}
```

JogoTabuleiro – src – view – TabuleiroView

```
package view;

import model.Tabuleiro;

import java.util.Observable;
import java.util.Observer;

public class TabuleiroView implements Observer {
    private Tabuleiro tabuleiro;
    public TabuleiroView(Tabuleiro tab) {
        this.tabuleiro = tab;
    }
    public void showSituacaoAtual(Tabuleiro tab){
        for (int i = 0; i < tab.getQtdCasas(); i++) {
            //if (tab.getCasaOcupada(i) != null) {
            //    System.out.println("casa "+i+": "+tab.getCasaOcupada(i)+" -
>"+tab.getJogadoresCasa(i));
            //}
            if (tab.getCasaOcupada(i) != null) {
                CasaView.desenhaCasa(tab.getCasa(i), tab.getJogadoresCasa(i));
            }
        }
        System.out.println("============");

    }


    @Override
    public void update(Observable o, Object arg) {
        this.showSituacaoAtual(this.tabuleiro);
    }
}
```

JogoTabuleiro – src – Main

```
public class Main {
    public static void main(String[] args) {
        JogoController.getInstance().iniciarJogo();


    }
}
```

JogoTabuleiro – src – Teste

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

        JogoController.getInstance().iniciarJogo();
```

```
    }
}
```

JogoTabuleiro – src – JogoController

```
import model.*;
import view.JogoView;
import view.TabuleiroView;

import java.util.Observable;
import java.util.Observer;

public class JogoController implements Observer{

    private static JogoController instancia;

    private int qtdJogadores;
    private Tabuleiro tabuleiro;
    private int jogadorAtual = 0;
    private boolean finalizado = false;
    private TabuleiroView tbv;
    private JogoView jgv;
    private RegraDoJogo regra;


    private JogoController() {
        //regra =  new RegraDoJogo();
        regra =  new NovaRegraDoJogo();
    }
    public static JogoController getInstance() {
        if (instancia == null) {
            instancia = new JogoController();
        }
        return instancia;
    }
    public void iniciarJogo() {
        tabuleiro = new Tabuleiro(10,20,20);
        tbv = new TabuleiroView(tabuleiro);
        jgv = new JogoView();
        qtdJogadores = JogoView.intQtdJogadores(2, 6);
        registrarJogadores();

        while(! finalizado) {
            iniciarJogada();
            JogoView.continuar();
            proximoJogador();
            //tbv.showSituacaoAtual(tabuleiro);
        }

    }
```

```
    private void proximoJogador() {
        jogadorAtual++;
        if(jogadorAtual == qtdJogadores) {
            jogadorAtual = 0;
        }
    }
    public void registrarJogadores() {
        for (int i = 1; i <= qtdJogadores; i++) {
            String n = JogoView.informeJogador(i);
            try {
                Jogador j = new Jogador(n);
                tabuleiro.addJogador(j);
                j.addObserver(tbv);
                j.addObserver(jgv);
                j.addObserver(JogoController.getInstance());


            } catch (MuitosJogadoresException e) {
                e.printStackTrace();
            }
        }
    }
    private void iniciarJogada() {

JogoView.mostraJogadorAtual(tabuleiro.getJogadores().get(jogadorAtual));
        Dado d = new Dado();
        tabuleiro.getJogadores().get(jogadorAtual).avanca(d.jogar());

    }

    public void iniciarJogoOld() {
        Tabuleiro tab = new Tabuleiro(10,20,20);
        try {

            tab.addJogador(new Jogador("Jackson1"));
            tab.addJogador(new Jogador("Jackson2"));
            Jogador a = null;
            try {
                a.getNome();
            } catch (NullPointerException e) {
                System.out.println("ops, tu tentou usar um null como jogador");
            }

            tab.addJogador(new Jogador("Jackson3"));
            tab.addJogador(new Jogador("Jackson4"));
            tab.addJogador(new Jogador("Jackson5"));

            tab.addJogador(new Jogador("Jackson6"));
            tab.addJogador(new Jogador("Jackson7"));
        } catch (MuitosJogadoresException e) {
```

```
            System.out.println(e.getMessage());
        }
        catch (NullPointerException e) {
            System.out.println("ops, tu tentou usar um null como jogador");
        }
        finally {
            System.out.println("depois de tudo");
        }
        for(int i = 0; i < tab.getJogadores().size(); i++) {
            System.out.println(tab.getJogadores().get(i));
        }
    }


    @Override
    public void update(Observable o, Object arg) {
        if (regra.alguemGanhou(tabuleiro)) {
            finalizado = true;
            System.out.println(regra.quemGanhou().getNome()+" ganhou!!!");
        }
    }
}
```