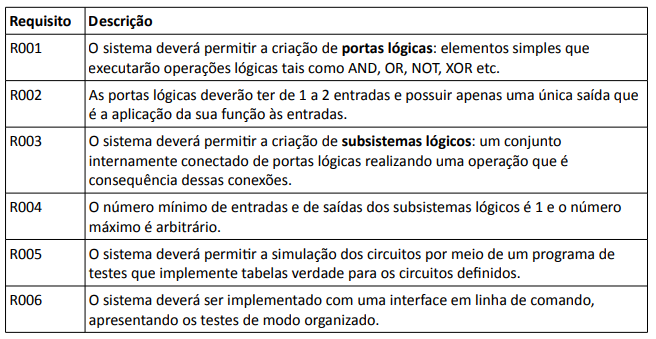


**Laboratório de Engenharia de Software**

**Lista 1**

|  |  |
| --- | --- |
| **Nome** | **RA** |
| Andressa Almeida | 1680481522041 |
| Daniel Augusto | 1680481521030 |
| Diego de Melo | 1680481521036 |
| Elias Sanai | 1680481521016 |

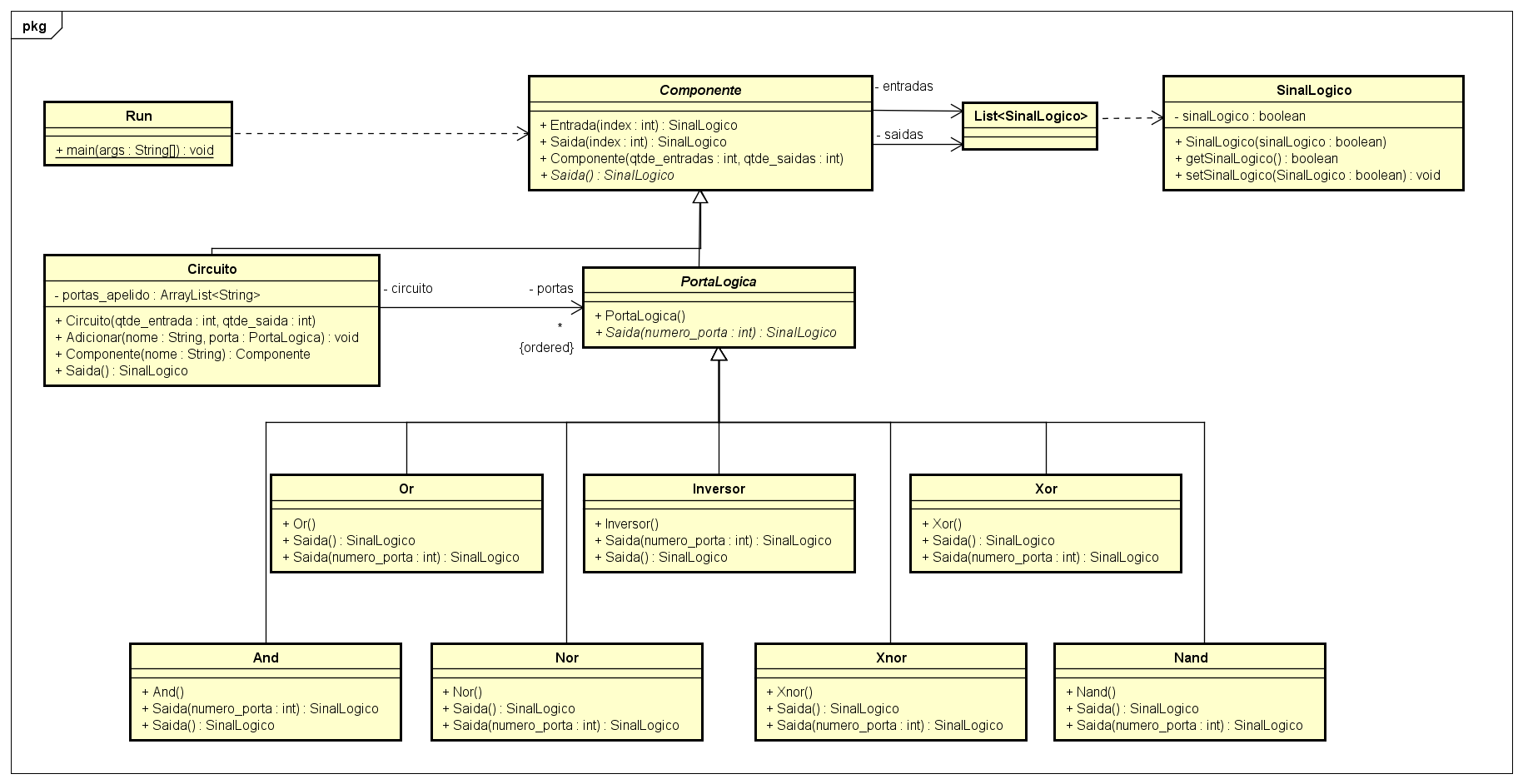


O que é para fazer?

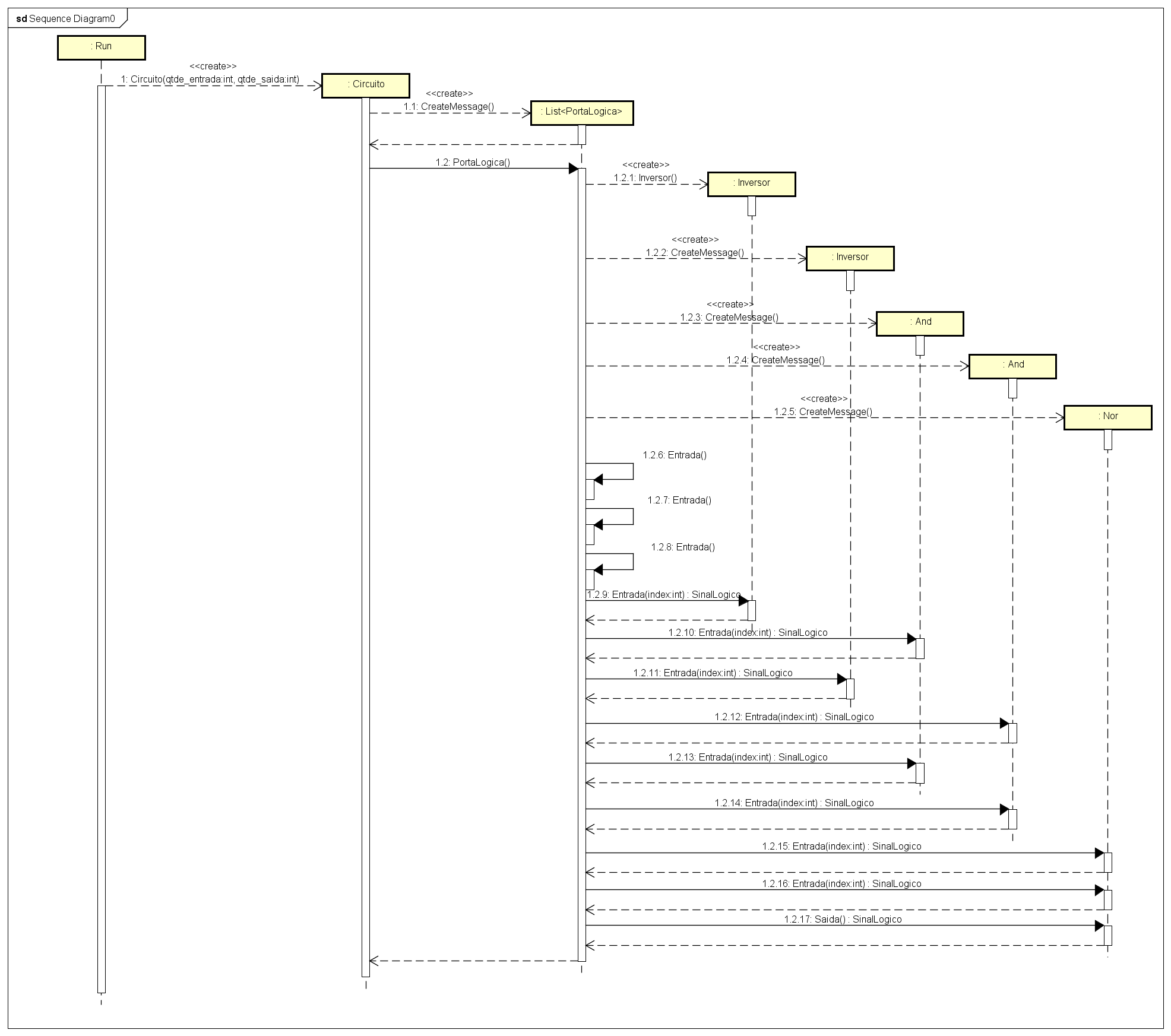
1. Procurar padrões de projeto que poderiam ser utlizados na resolução deste problema: indicar o nome, referência e justfcar sua escolha.

2. Elaborar um diagrama UML de classes do projeto do sistema e um diagrama de sequências UML que exemplifque um cenário de utlização do sistema (por exemplo, a construção e uso do segundo circuito de exemplo desta lista).

3. Implementar o sistema em Java. Não é necessário criar uma interface gráfca – o sistema pode funcionar em linha de comando, se preferir.



Foi utilizado o Composite porque um circuito é feito de várias PortaLógicas e além disso, uma Porta Lógica pode depender de outra porta lógica, a partir dessa informação o que acontece é a necessidade de hierarquia.



public class And extends PortaLogica {

public And() {

super();

}

@Override

public SinalLogico Saida() {

boolean r =( entradas.get(0).getSinalLogico() && entradas.get(1).getSinalLogico());

saidas.add(0, new SinalLogico(r));

return saidas.get(0);

}

}

public class Xor extends PortaLogica{

public Xor() {

super();

}

@Override

public SinalLogico Saida() {

boolean r=(entradas.get(0).getSinalLogico() == entradas.get(1).getSinalLogico());

saidas.add(0, new SinalLogico(r));

return saidas.get(0);

}

}

public abstract class Componente {

protected final ArrayList<SinalLogico> entradas;

protected final ArrayList<SinalLogico> saidas;

public SinalLogico Entrada(int index) {

return entradas.get(index - 1);

}

public SinalLogico Saida(int index) {

return saidas.get(index - 1);

}

public Componente(int qtde\_entradas, int qtde\_saidas) {

this.entradas = new ArrayList<>(qtde\_entradas);

this.saidas = new ArrayList<>(qtde\_saidas);

for(int i = 0; i < qtde\_entradas; i++)

this.entradas.add(new SinalLogico(false));

for(int i = 0; i < qtde\_saidas; i++)

this.saidas.add(new SinalLogico(false));

}

public abstract SinalLogico Saida();

}

public class Circuito extends Componente {

private final ArrayList<PortaLogica> portas;

private final ArrayList<String> portas\_apelido;

public Circuito(int qtde\_entrada, int qtde\_saida) {

super(qtde\_entrada, qtde\_saida);

this.portas = new ArrayList<>();

this.portas\_apelido = new ArrayList<>();

}

public void Adicionar(String nome, PortaLogica porta) {

portas.add(porta);

portas\_apelido.add(nome);

}

public Componente Componente(String nome)

{

for(int i = 0; i < portas\_apelido.size(); i++)

{

if(portas\_apelido.get(i).equals(nome))

return portas.get(i);

}

return null;

}

@Override

public SinalLogico Saida() {

throw new UnsupportedOperationException("Not supported yet."); //To change body of generated methods, choose Tools | Templates.

}

}

public class Or extends PortaLogica {

public Or() {

super();

}

@Override

public SinalLogico Saida() {

boolean r=entradas.get(0).getSinalLogico() || entradas.get(1).getSinalLogico();

saidas.add(0, new SinalLogico(r));

return saidas.get(0);

}

}

public class Inversor extends PortaLogica {

public Inversor() {

super();

}

@Override

public SinalLogico Saida(int numero\_porta) {

boolean r=!entradas.get(0).getSinalLogico();

saidas.add(numero\_porta - 1, new SinalLogico(r));

return saidas.get(numero\_porta - 1);

}

public SinalLogico Saida() {

boolean r=!entradas.get(0).getSinalLogico();

saidas.add(0, new SinalLogico(r));

return saidas.get(0);

}

}

public class Nand extends PortaLogica{

public Nand() {

super();

}

@Override

public SinalLogico Saida() {

boolean r = entradas.get(0).getSinalLogico() && entradas.get(1).getSinalLogico();

r=!r;

saidas.add(0, new SinalLogico(r));

return saidas.get(0);

}

}

public class Nor extends PortaLogica {

public Nor() {

super();

}

@Override

public SinalLogico Saida() {

boolean r=(entradas.get(0).getSinalLogico() || entradas.get(1).getSinalLogico());

r=!r;

saidas.add(0, new SinalLogico(r));

return saidas.get(0);

}

}

public abstract class PortaLogica extends Componente{

public PortaLogica() {

super(2, 1);

}

@Override

public abstract SinalLogico Saida(int numero\_porta);

}

public class Run {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

Circuito circuito = new Circuito(3, 1);

circuito = new Circuito(3,1); //3 entradas e 1 saída

//elemento 1

circuito.Adicionar("INV1", new Inversor());

//elemento 2

circuito.Adicionar("INV2", new Inversor());

//elemento 3

circuito.Adicionar("AND1", new And());

//elemento 4

circuito.Adicionar("AND2", new And());

//elemento 5

circuito.Adicionar("NOR1", new Nor());

//níveis das entradas

circuito.Entrada(1).setSinalLogico(false);

circuito.Entrada(2).setSinalLogico(true);

circuito.Entrada(3).setSinalLogico(false);

//conexões

circuito.Componente("INV1").Entrada(1).setSinalLogico(circuito.Entrada(1).getSinalLogico());

circuito.Componente("AND1").Entrada(2).setSinalLogico(circuito.Entrada(2).getSinalLogico());

circuito.Componente("INV2").Entrada(1).setSinalLogico(circuito.Entrada(2).getSinalLogico());

circuito.Componente("AND2").Entrada(2).setSinalLogico(circuito.Entrada(3).getSinalLogico());

circuito.Componente("AND1").Entrada(1).setSinalLogico(circuito.Componente("INV1").Saida().getSinalLogico());

circuito.Componente("AND2").Entrada(1).setSinalLogico(circuito.Componente("INV2").Saida().getSinalLogico());

circuito.Componente("NOR1").Entrada(1).setSinalLogico(circuito.Componente("AND1").Saida().getSinalLogico());

circuito.Componente("NOR1").Entrada(2).setSinalLogico(circuito.Componente("AND2").Saida().getSinalLogico());

//saída – a função recalcula o circuito

boolean valor=circuito.Componente("NOR1").Saida().getSinalLogico(); //= circuito.Saida(1).getSinalLogico();

System.out.println("Resultado do circuito pré-programado: "+ valor);

}

}

public class Xnor extends PortaLogica{

public Xnor() {

super();

}

@Override

public SinalLogico Saida() {

boolean r=entradas.get(0).getSinalLogico() ^ entradas.get(1).getSinalLogico();

r=!r;

saidas.add(0, new SinalLogico(r));

return saidas.get(0);

}

}