Java
17
Objectos

Vitor Vaz da Silva

- Olhemos para o mundo que conhecemos
  - Um objecto é algo concreto
  - Um objecto antes de ser criado é uma ideia
  - *Dicionário - Objecto: Tudo o que é exterior ao espírito. Coisa. Assunto, matéria, causa, motivo. Fim, escopo.*

- Na programação orientada por objectos
  - À ideia chamamos Classe
  - Ao objecto chamamos Objecto

Java

```
Classe Porta{
    //descrever Porta de um modo a conter e realizar
    //tudo aquilo que se espera de Porta quando
    // a mandarmos fazer
}
//declaro que vou ter (ainda não tenho) uma
//Porta nova e que lhe vou chamar portinhola
Porta portinhola;

portinhola = new Porta();
//acabei de criar, fazer a Porta
```

Exercício

Eratóstenes considerava os números como um rio fluindo por um monte e ele colocava crivos para filtrar os números.

O primeiro crivo tinha o valor 2, e filtrava todos os múltiplos de 2, o primeiro a entrar foi o 3, e como não foi filtrado criou um crivo mais abaixo com o 3, depois foi a vez dos crivos 5, 7, 11 , … e desse modo conseguia saber os números primos por coincidirem com os números dos crivos.

(o 1 fugiu antes de se colocarem os crivos! ☺).

Solução – Iterativa sem objectos

```
void calculaPrimos3(){
   final int NUM_CRIVOS = 100;
   int crivo[] = new int[NUM_CRIVOS];
   int rio = 1;
   int idx=0;
   int i;

   while(idx<NUM_CRIVOS){
      rio++;
      for(i=0; i<idx; i++){
         if((rio % crivo[i])==0) break; //não é primo
      }
      if(i==idx) crivo[idx++]=rio;  //é primo
   }
   for(i=0; i < NUM_CRIVOS-1; i++){
      System.out.print(crivo[i] + (i%25==0 ? "\n": " "));
   }
}
```

```
public class Crivo {
    private Crivo seguinte=null;
    private int valor=0;

    public void seguinte(Crivo crivo) {
        seguinte=crivo;
    }

    public boolean temValor() {
        return (valor!=0);
    }

    public void novoNumero(int novoNumero) {
        if(valor==0) valor=novoNumero;  // o primeiro valor é primo
        if(novoNumero % valor != 0)
            if(seguinte!=null) seguinte.novoNumero(novoNumero);
    }

    public int valor() {
        return valor;
    }
}
```

```
public class Rio {
    private int fonte=1;

    public int novoNumero(){
        return (++fonte);
    }
}
```

Solução – com objectos

```
void calculaPrimos(){
	final int NUM_CRIVOS = 1000;
	Crivo crivo[] = new Crivo[NUM_CRIVOS];
	Rio rio = new Rio();

	crivo[NUM_CRIVOS-1]=new Crivo();
	for(int i=NUM_CRIVOS-2; i >=0 ; i--){
		crivo[i]=new Crivo();
		crivo[i].seguinte(crivo[i+1]);
	}

	while(!crivo[NUM_CRIVOS-1].temValor()){
		crivo[0].novoNumero(rio.novoNumero());
	}

	for(int i=0; i < NUM_CRIVOS-1; i++){
		System.out.print(crivo[i].valor() + (i%25==0 ? "\n": " "));
	}

}
```

Solução – com objectos

- Os objectos não necessitam de ficarem num array ….
- Desde que eles estejam referenciados (ligados a outros objectos) não ficam perdidos!

Solução com objectos

```
void calculaPrimos(){
        final int NUM_CRIVOS = 1000;
        Rio rio = new Rio();

        Crivo ultimoCrivo= new Crivo();
        Crivo primeiroCrivo;
        Crivo actual=ultimoCrivo;

        for(int i=0; i<NUM_CRIVOS; i++){
                Crivo novo=new Crivo();
                novo.seguinte(actual);
                actual=novo;
        }
        primeiroCrivo=actual;

        while(!ultimoCrivo.temValor()){
                primeiroCrivo.novoNumero(rio.novoNumero());
        }

        // actual = primeiroCrivo;
        for(int i=0; i < NUM_CRIVOS-1; i++){
                System.out.print(actual.valor() + (i%25==0 ? "\n": " "));
                actual=actual.seguinte();
        }
}
```

```
public class Crivo {

// ACRESCENTAR

        public Crivo seguinte() {
                return seguinte;
        }
}
```

Diagrama de Classes

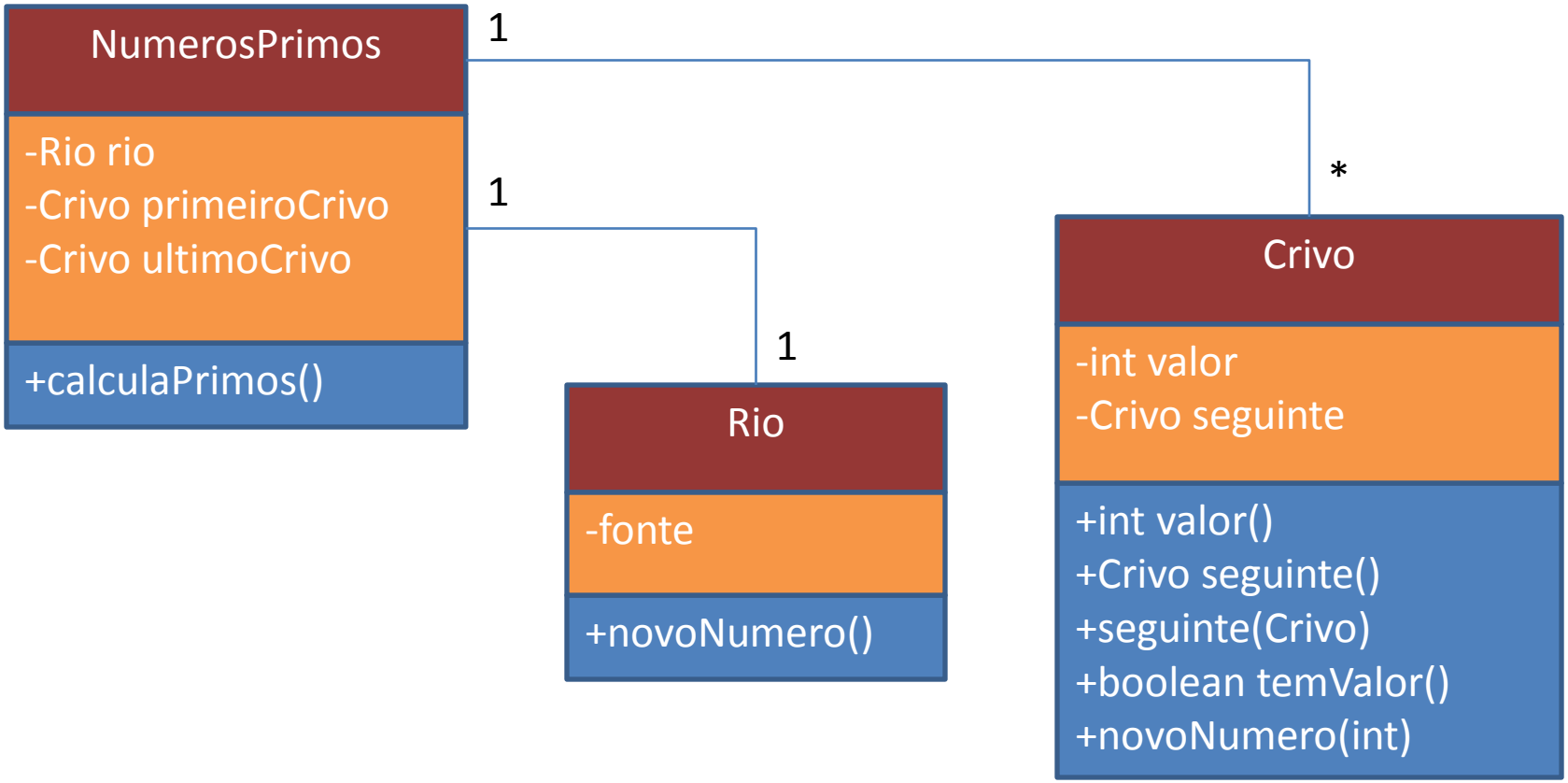
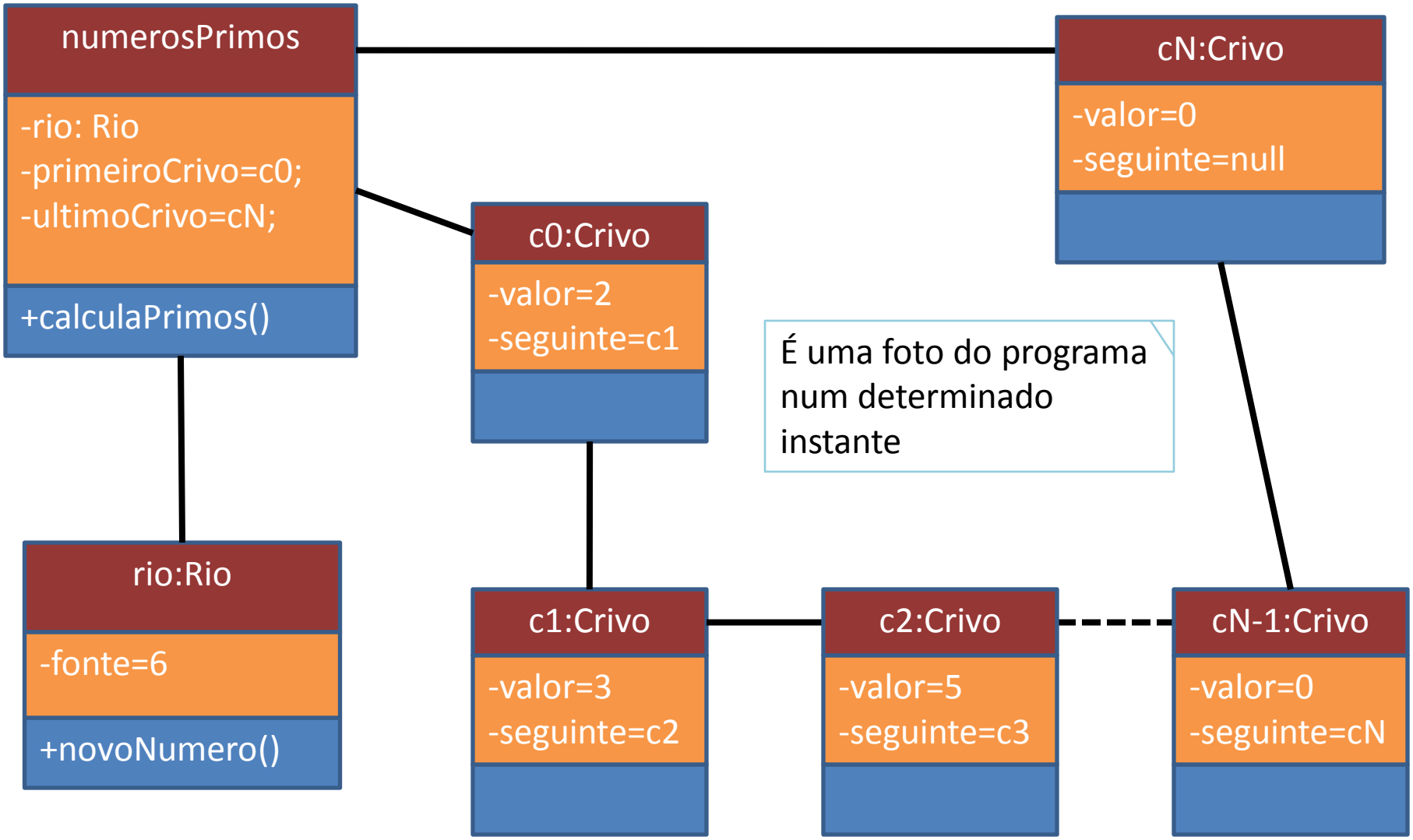
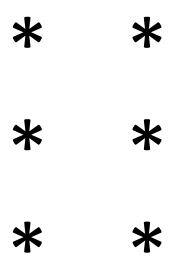


Diagrama de Objectos



Exercício – Um Dado

- Faça um programa orientado por objectos que simule o lançamento de um dado



Exercício – Dois Dados

- Altere o programa para utilizar dois dados

```
        *   *
  * * *     *   *
        *   *
```

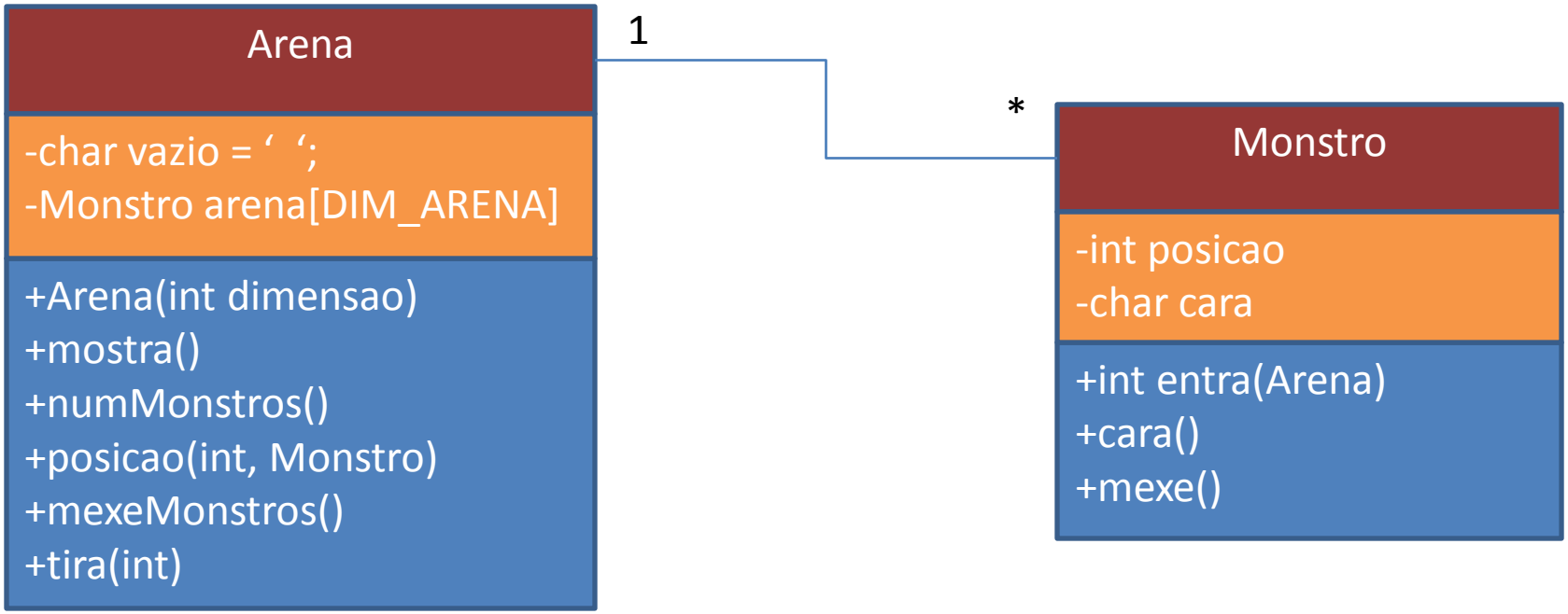
Este programa funcionará também para um número n de dados!

Exercício – Arena com Monstros

- Numa arena estão vários monstros. Quando os monstros se encontram no mesmo local tiram à sorte, um fica o outro desaparece.
- O jogo termina quando só houver um monstro na arena.
- A arena é um corredor circular com a largura de um monstro.
- Cada monstro decide se anda uma posição em qualquer sentido ou se fica no mesmo local.

Exercício – Arena com Monstros

Diagrama de Classes – Solução 1



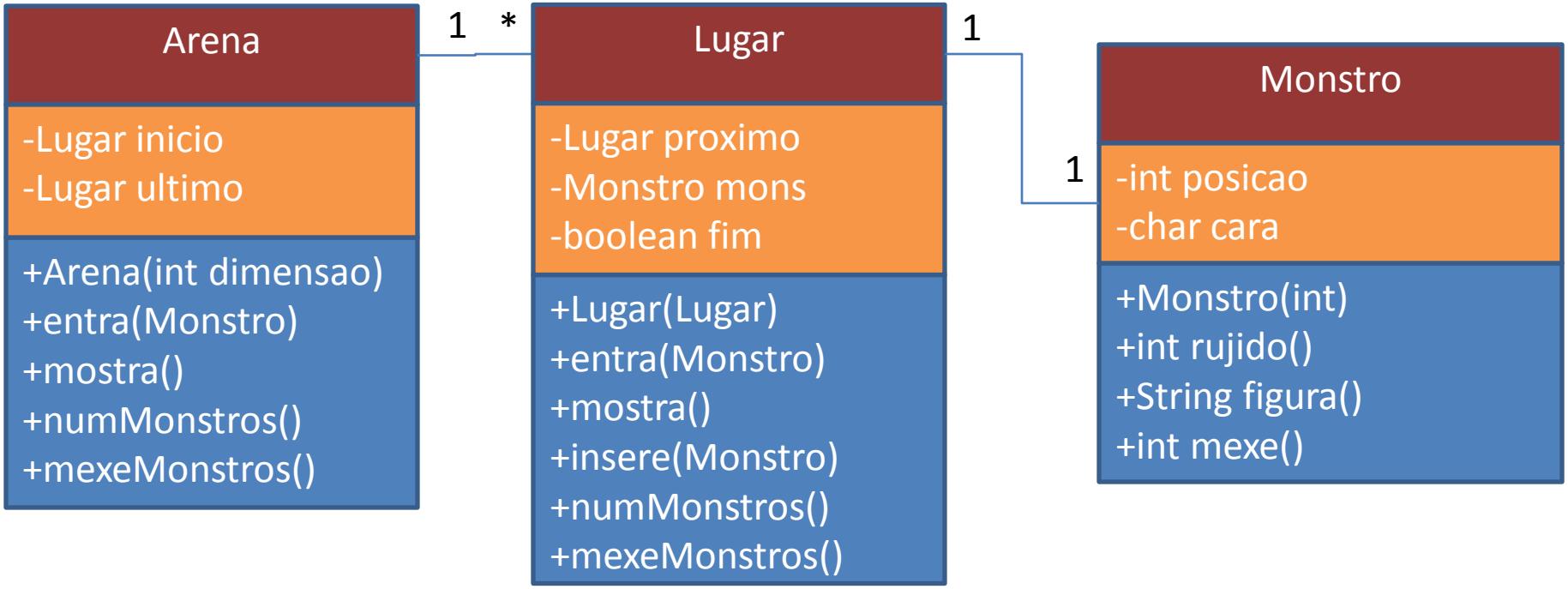
//Programa Principal

Usa arrays de objectos

static final int *NUM_MONSTROS=10;*
static final int *DIM_ARENA=20;*
static Arena *arenaMons=new Arena(DIM_ARENA);*
static Monstro *monstros[] = new Monstro[NUM_MONSTROS];*

Exercício – Arena com Monstros

Diagrama de Classes – Solução 2



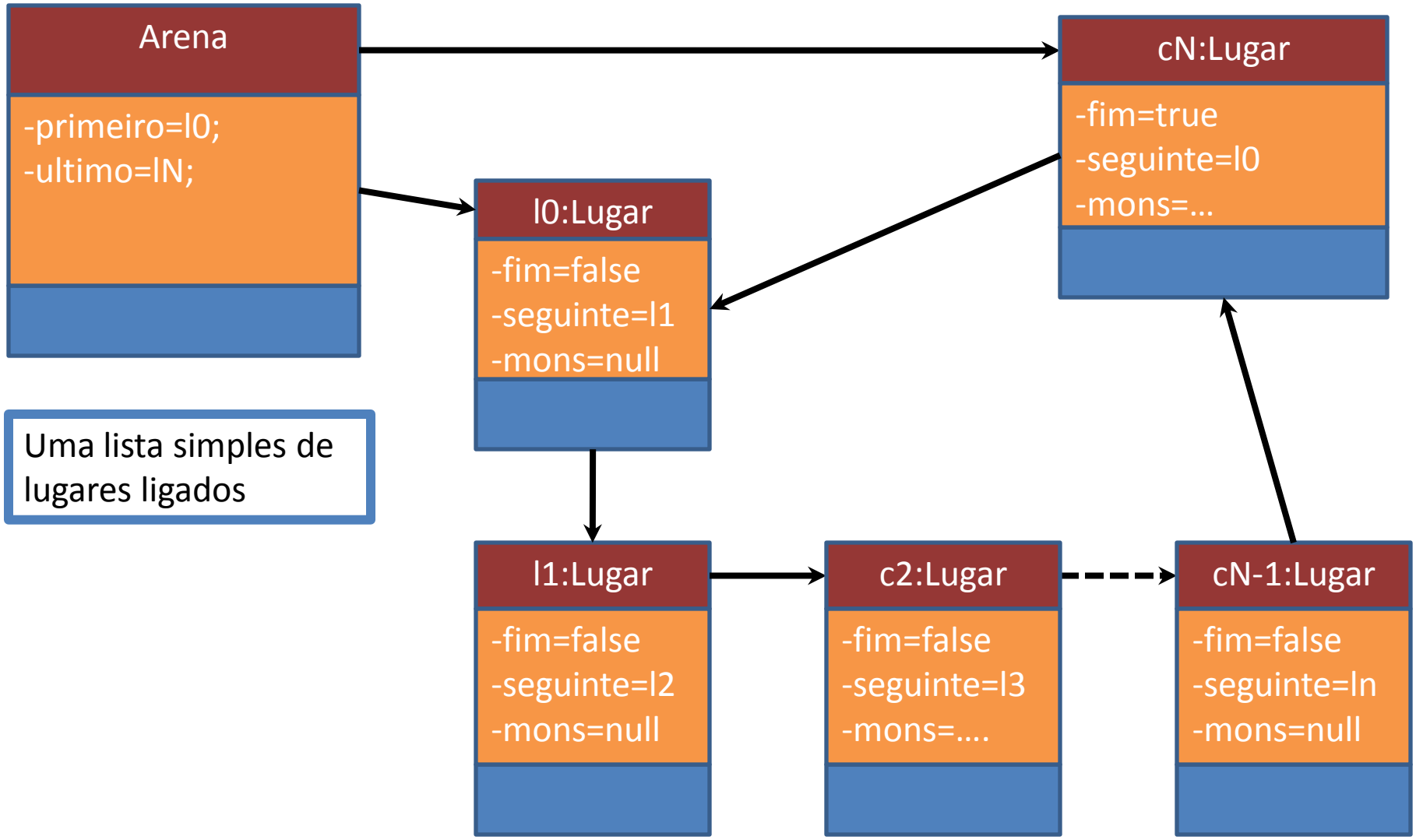
//Programa Principal

static final int *NUM_MONSTROS=10;*
static final int *DIM_ARENA=20;*
static Arena *arena=new Arena(DIM_ARENA);*

A Arena é um conjunto de lugares ligados de modo que qualquer lugar apenas conhece o seguinte

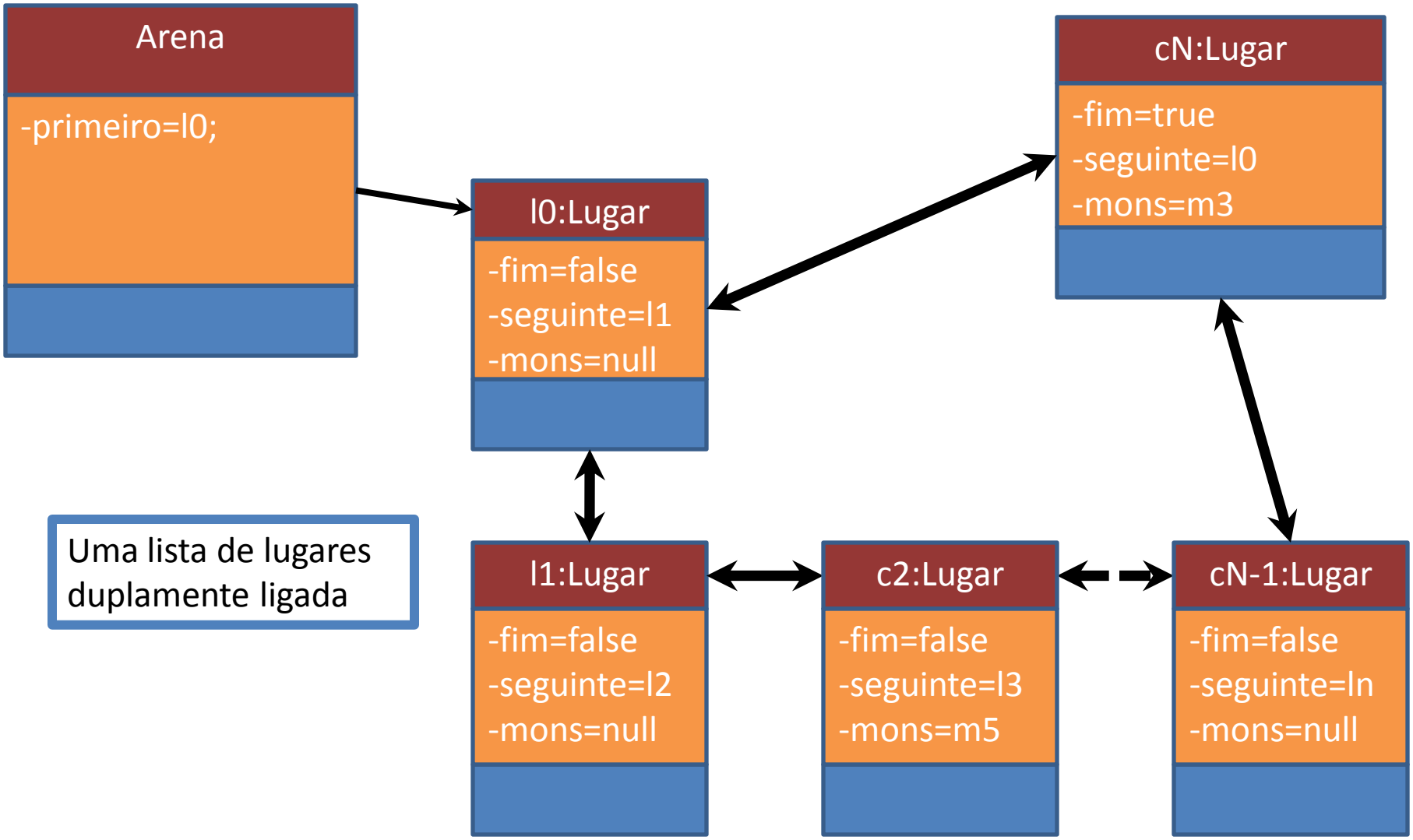
Exercício – Arena com Monstros

Diagrama de Objectos – Solução 2



Exercício – Arena com Monstros

Diagrama de Objectos – Solução 3

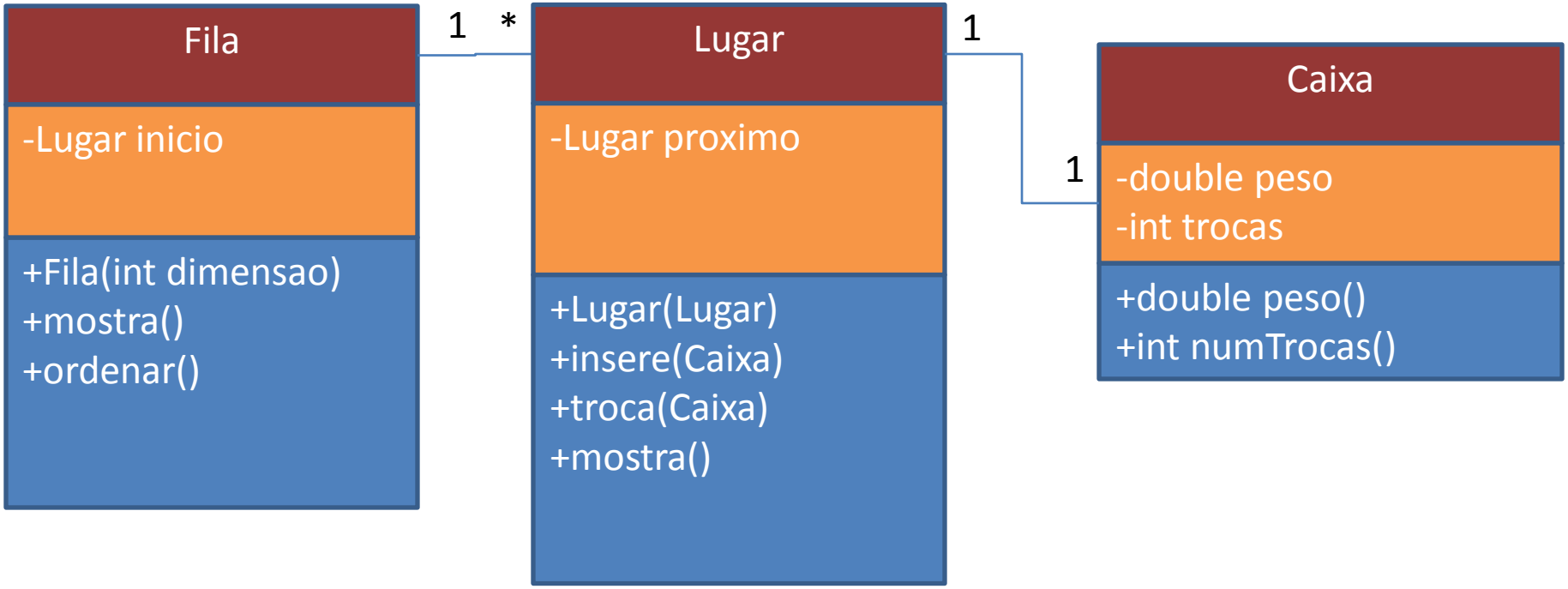


Exercício - Ordenar Fila de Caixas

- Coloque aleatoriamente numa fila um conjunto de caixas que se distinguem pelo peso que têm.
- Conte o número de mudanças que cada caixa tem de fazer para que fiquem todas ordenadas por ordem crescente do peso.

Exercício – Ordenar Fila de Caixas

Diagrama de Classes



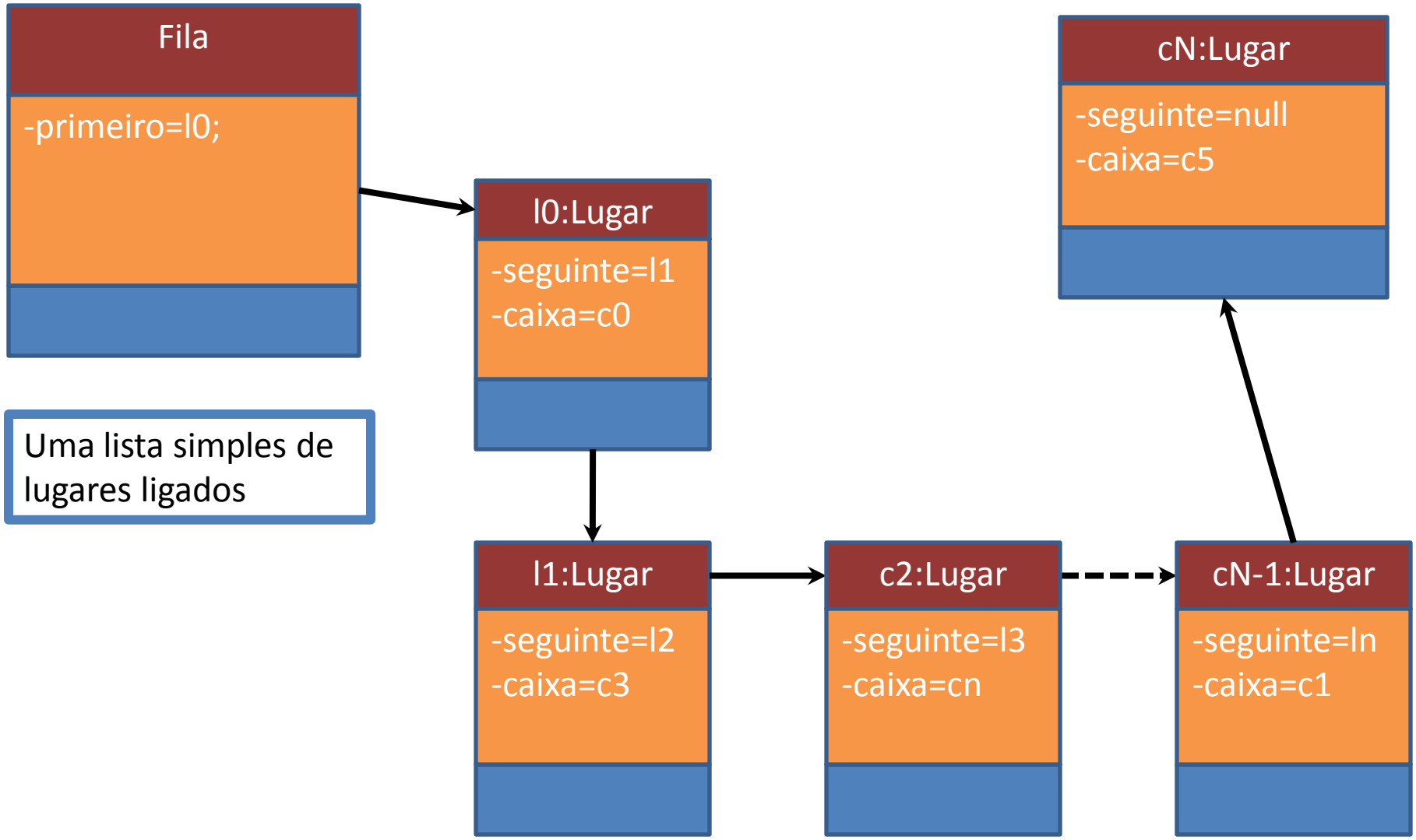
//Programa Principal

static final int *NUM_CAIXAS=10;*
static Fila *fila=new Fila(NUM_CAIXAS);*

Uma lista é um conjunto de lugares ligados de modo que um apenas conhece o próximo. Em cada lugar há uma caixa.

Exercício – Ordenar Fila de Caixas

Diagrama de Objectos



Exercício – Mundo Celular

- Num mundo celular, cada organismo pode ter 8 células vizinhas. Cada célula mantém o seu próprio estado excepto nas condições seguintes:
  - A célula nasce caso tenha exactamente 3 células vizinhas
  - A célula morre por falta de recursos caso tenha 4 ou mais células vizinhas
  - A célula morre de solidão caso tenha duas ou menos células vizinhas
- O mundo celular forma uma toroide

Exercício - MasterMind

- O jogo MasterMind consiste na escolha aleatória de uma sequência de cores que tem de ser adivinhada pelo utilizador.
- Cada tentativa tem como resultado dois números, o primeiro indica o número de cores certas em lugares certos, o segundo o número de cores certas em lugares errados.

Referências

http://Tektonia.com