Linguagem Funcional 4

IN1007 - Paradigmas de Linguagens de Programação

Gabriela Cunha (gcs)
Roberto Souto Maior (rsmbf)

URL do projeto: http://cin.ufpe.br/~rsmbf/PLP/

Visão geral

- Estende a Linguagem Funcional 2 com
  - Tuplas
  - Casamento de padrão
  - Comandos de concorrência para criar processos similares aos da linguagem Erlang
    - Spawn
    - Send
    - Receive

Ambiente do interpretador

- Existem dois tipos de ambiente na nossa linguagem
  - Ambiente do programa: gerencia a comunicação entre processos
  - Ambiente do processo: gerencia operações dentro de cada processo

Ambiente do Programa

- Funciona como um gerenciador de processos
- Inclui dois componentes:
  - Um processo principal (main)
  - Uma lista de mapeamentos de identificadores em processos
- Possui as seguintes operações:
  - Enviar mensagem
  - Criar processo
  - Encerrar processo

Ambiente do Processo

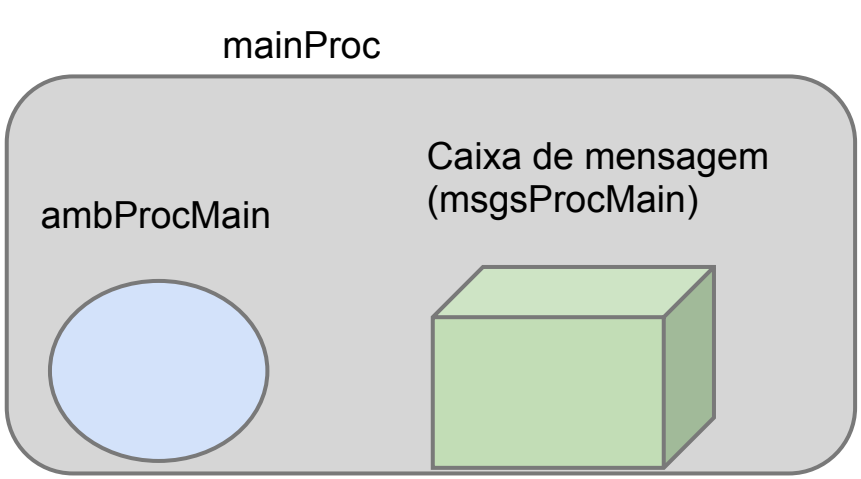
- Tem os seguintes componentes:
  - Um processo
  - Os componentes do ambiente da Linguagem Funcional 1
    - Uma pilha de Id - Valor
    - Uma pilha de Id - Def Funcao

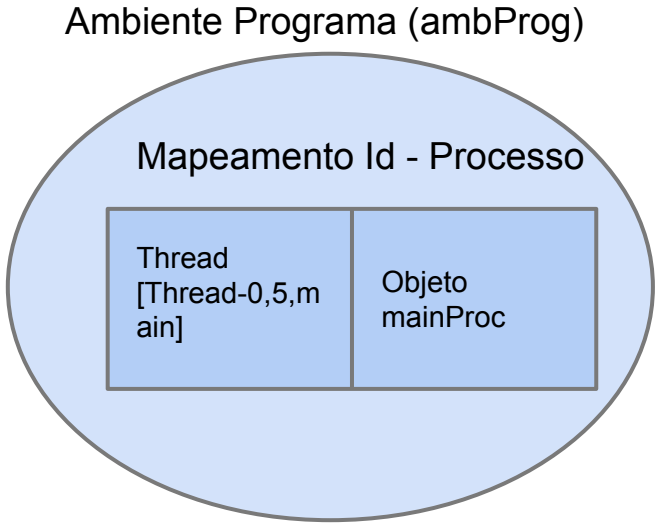
Processo

- Thread de Java
- Método run executa aplicação de função
- Comunicação via troca de mensagens
  - Quando outro processo envia uma mensagem, ela é adicionada na caixa de mensagens
  - Quando um receive é encontrado, damos wait no processo até que ele consiga ler alguma mensagem da caixa. Fazemos pattern matching aqui. Se o pattern matching conseguir casar, removemos a mensagem lida da caixa de mensagem.
  - Se outro processo for criado, o ambiente do processo criador é clonado para o novo processo

Exemplo Execução

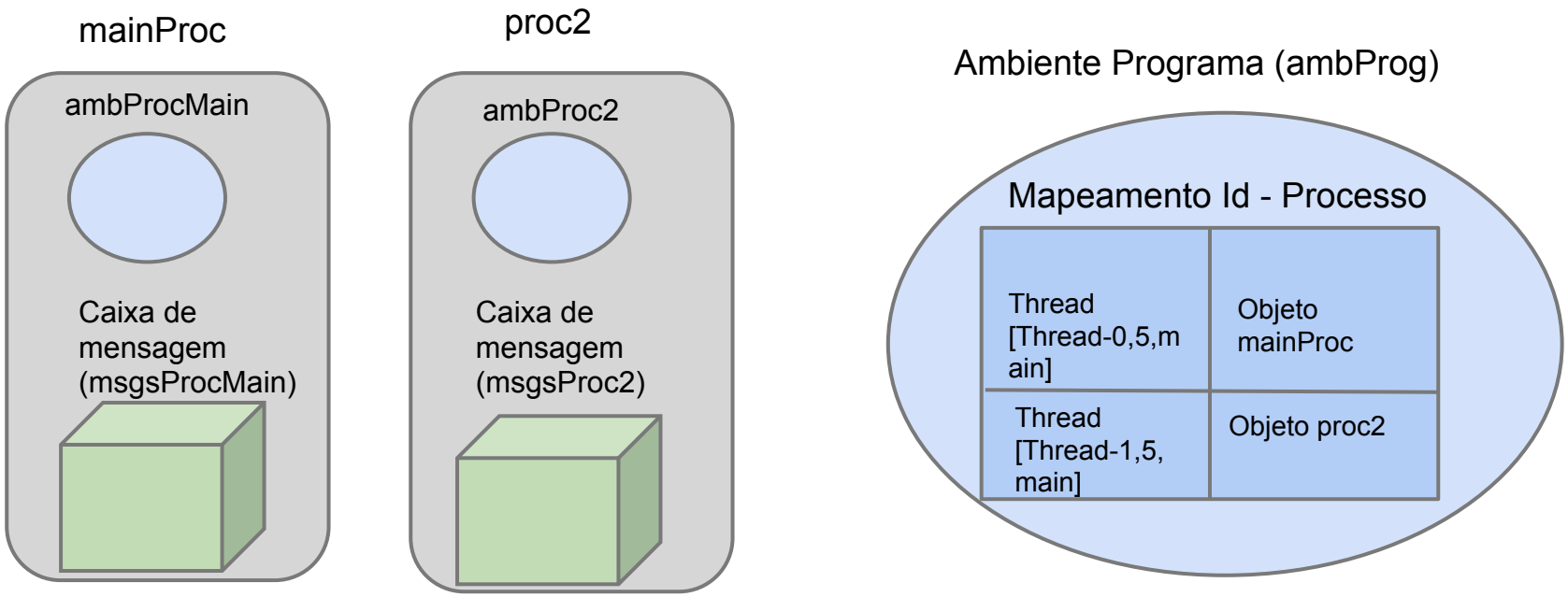
```
let fun soma pid =
        receive
                {x,y} -> pid ! (x + y)
        end
in
        let var pid = spawn(soma, [self()]) in
                pid ! {2,3},
                receive
                        x -> x
                end
```





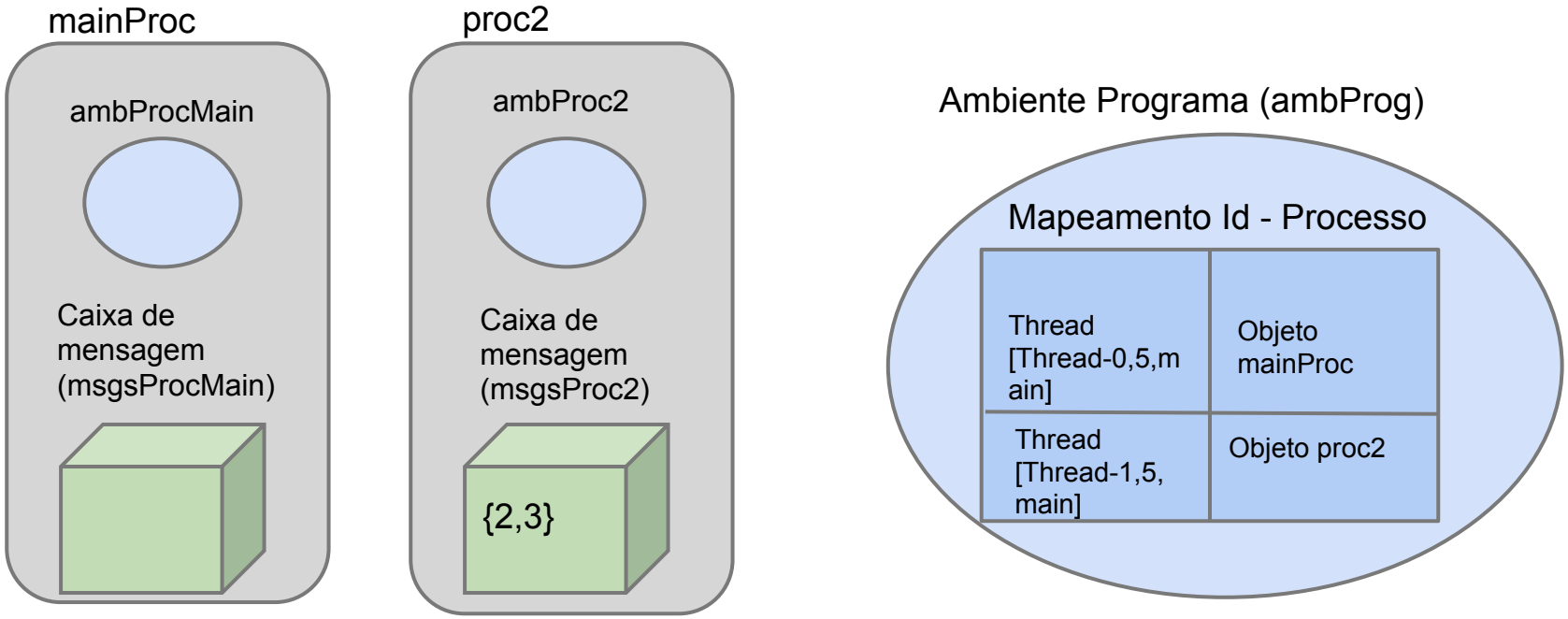

```
let fun soma pid =
        receive
            {x,y} -> pid ! (x + y)
        end
in ....
```

```
programa.executar(); //Func4Parser.java
ambProg.getMainProc().avaliar(exp); //Programa.java
processes.put(mainProc.toString(),mainProc); //ContextoPrograma.java
```



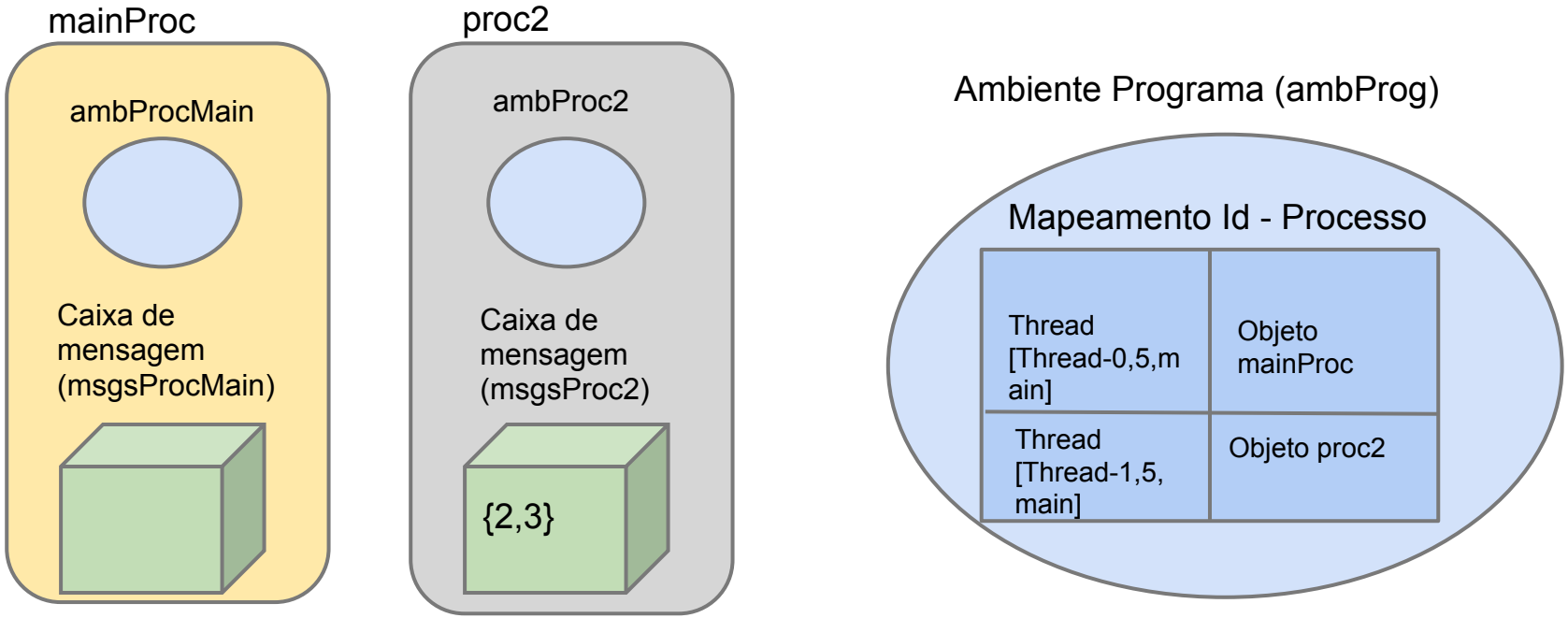

```
… in
let var pid = spawn(soma, [self()]) in
              pid ! {2,3},
              receive
                    x -> x
              end
```

```
ObjExpSpawn.avaliar(ambProcMain);
ambProcMain.createProcess(func, args); //ExpSpawn.java
ambProg.createProc(...); //Processo.java
processes.put(proc2.toString(), proc2); //ContextoPrograma.java
soma.avaliar(ambProc2) //Processo.java
```



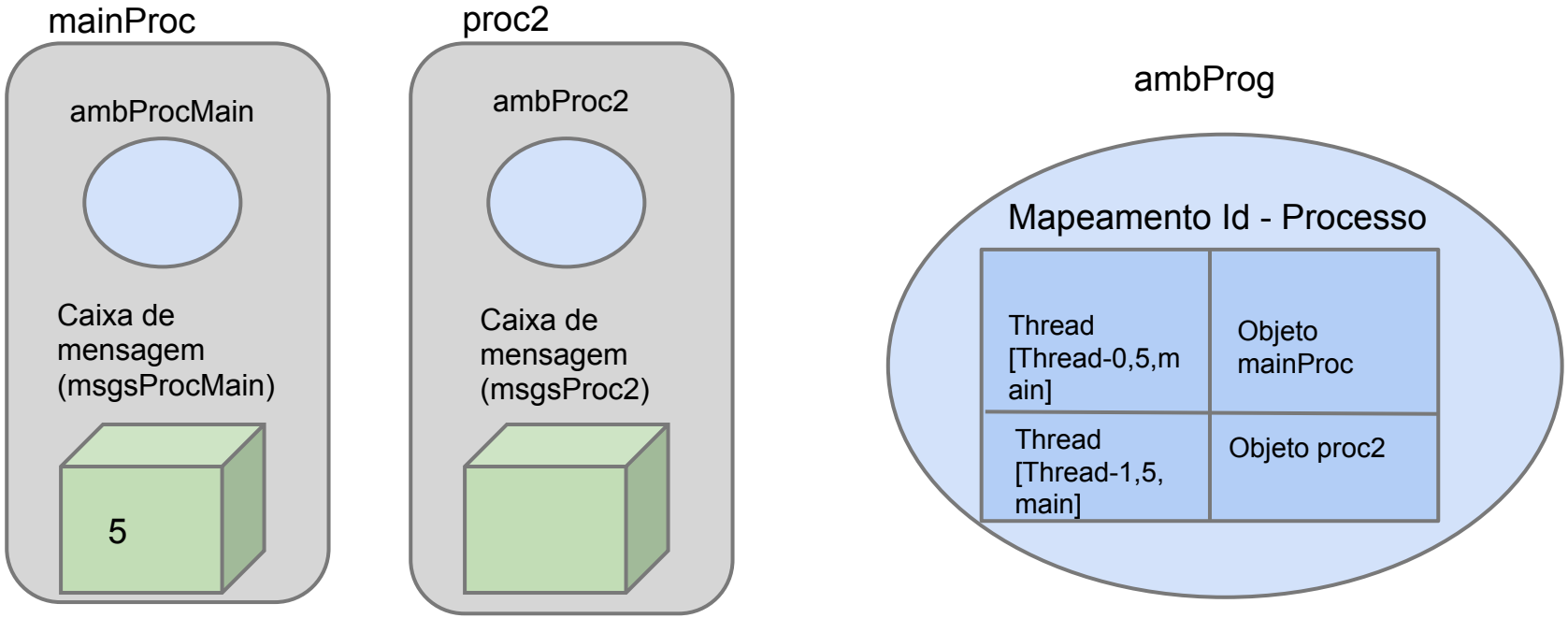

```
… in
let var pid = spawn(soma, [self()]) in
              pid ! {2,3},
              receive
                    x -> x
              end
```

```
ObjExpSend.avaliar(ambProcMain)
ambProcMain.send(Thread[Thread-1,5,main], {2,3}) //ExpSend.java
ambProg.sendMsg(Thread[Thread-1,5,main], {2,3}) //Processo.java
proc2.includeMessage({2,3}) //AmbientePrograma.java
msgsProc2.add({2,3}) //Processo.java
```



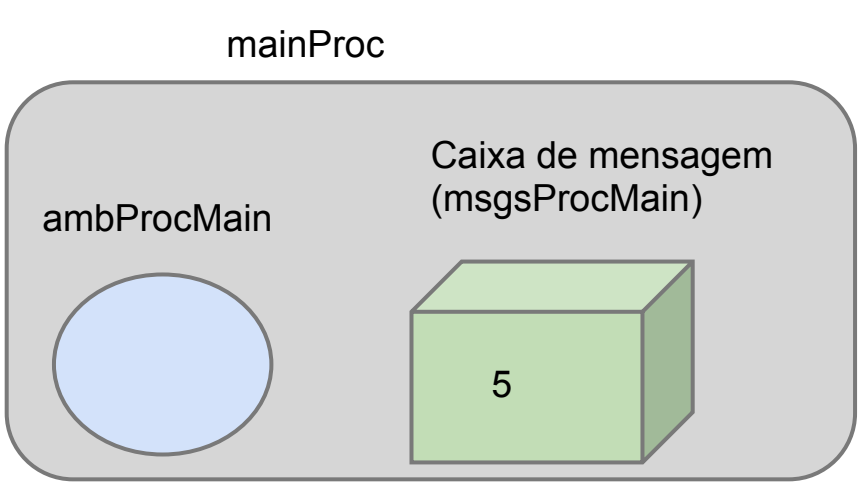

```
… in
let var pid = spawn(soma, [self()]) in
        pid ! {2,3},
        receive
                x -> x
        end
```

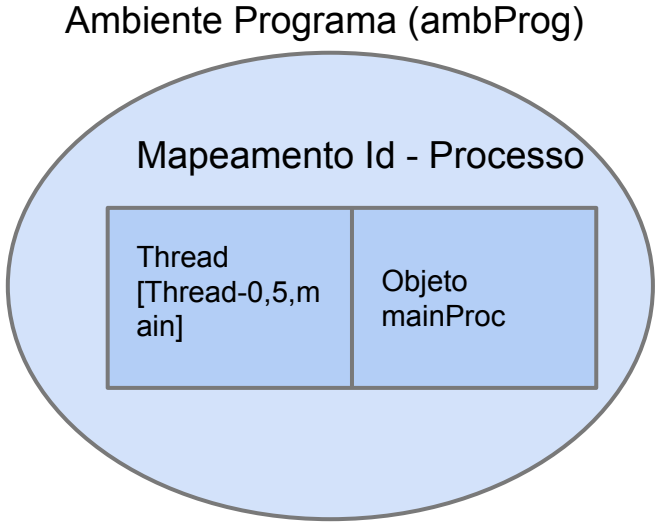
ObjExpReceive.avaliar(ambProcMain)
mainProc.wait() //ExpReceive.java




```
let fun soma pid =
        receive
                {x,y} -> pid ! (x + y)
        end
in ....
```

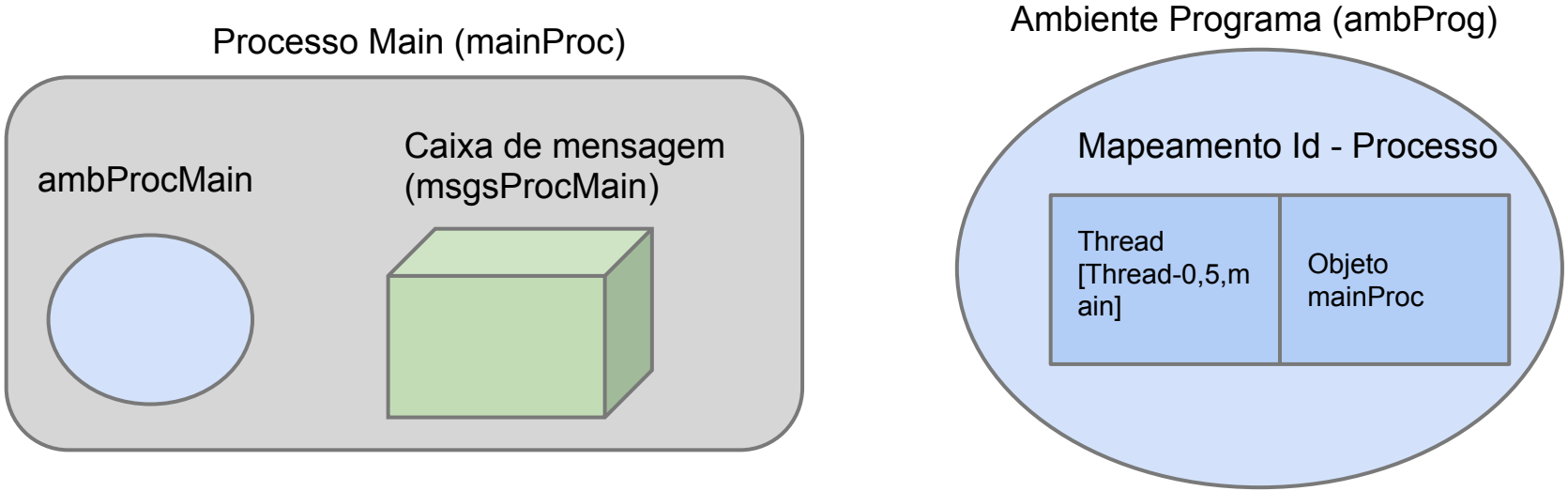
```
ObjExpReceive.avaliar(ambProc2)
proc2.readMessage(...) //ExpReceive.java
msgsProc2.remove(...)    //Processo.java -- REMOVE {2,3}
procMain.includeMessage(2 + 3) //AmbientePrograma.java
procMain.notify() //AmbientePrograma.java
```






```
let fun soma pid =
        receive
                {x,y} -> pid ! (x + y)
        end
in ....
```

ambProg.killProc(Thread[Thread-1,5,main]) //Processo.java
processes.remove(Thread[Thread-1,5,main]) //AmbientePrograma.java




```
… in
let var pid = spawn(soma, [self()]) in
        pid ! {2,3},
        receive
                x -> x
        end
```

ObjExpReceive.avaliar(ambProc2)
proc2.readMessage(...) //ExpReceive.java
msgsProc2.remove(...) //Processo.java -- REMOVE 5
//PROGRAMA RETORNA 5

BNF

http://www.cin.ufpe.br/~rsmbf/PLP/bnf.html

Linguagem Funcional 4

IN1007 - Paradigmas de Linguagens de Programação

Gabriela Cunha (gcs)
Roberto Souto Maior (rsmbf)

URL do projeto: http://cin.ufpe.br/~rsmbf/PLP/