Universidade Federal do Pampa – UNIPAMPA
Curso Bacharelado em Engenharia de Computação

Disciplina de Engenharia de Software
Prof. Carlos Michel Betemps

Documentação de Software - Gerenciador Dinâmico de Memória.

Integrantes do grupo: André Magalhães, Douglas Aquino, Ronaldo Amato.

Universidade Federal do Pampa

Este documento visa esclarecer os conceitos principais e métodos abordados durante a solução aplicada para criar um algoritmo paralelo/concorrente de gerenciamento dinâmico de memória. Como abordagem inicial, temos uma análise de requisitos globais do problema, buscando estabelecer os objetivos, com auxílio das ferramentas disponibilizadas pela disciplina de engenharia de software. Esta aplicação, por se tratar de um sistema de modelagem e simulação, inclui diversas classes e objetos separados que interagem entre si, colaborativamente. Diante disto, o modelo ideal para a elaboração do processo de desenvolvimento de software, foi selecionado o modelo de desenvolvimento incremental, auxiliado pela metodologia ágil e suas ferramentas, buscando tornar viável um ambiente de possíveis mudanças referentes à alterações durante a implementação do projeto.

A fim de definir os requisitos de usuário, visto que o software é estruturado para trabalhar de forma paralela, há parâmetros que devem ser definidos anteriormente ao seu início propriamente dito. O software deve fazer a gerência entre as requisições que serão criadas entre as filas de processamento e fazer a alocação e desalocação na memória. O usuário deverá inserir os valores correspondentes à quantidade, para o gerador de requisições. Sobre a alocação em memória, o uso mínimo e máximo para a chamada do algoritmo de desfragmentação.

Para a elaboração da engenharia de requisitos de sistema, foi feita a formulação de um fluxograma, buscando compreender a forma de execução da implementação, de forma funcional. Os dados podem ser inseridos na inicialização do sistema, de forma configurável:

- Tamanho da Heap;
- Limite de ocupação de memória como percentual, mínimo e máximo;
- Número de requisições a serem alocadas na memória.

Após a inserção destes dados, se iniciará a geração das requisições de forma aleatória, onde cada requisição ocupará um pedaço do vetor de requisições, com tamanho variável dentro do limite estipulado (mínimo e máximo).

Paralelamente, ao perceber a inserção destas requisições na fila de prontos do vetor de requisições, interno ao Process Control Block (PCB), o escalonador iniciará o seu fluxo, escalando as requisições conforme a ordem de chegada na fila, direcionando assim o seu processamento.

Durante este processo, o escalonador será responsável por transmitir as requisições, de acordo com a ordem de chegada, assim como durante a troca entre as filas. Esse fluxo obedece um limite definido para execução do quantum do cpu, visto que o algoritmo do escalonador fará o redirecionamento para a fila correspondente ready e terminated, caso o processo tenha seu quantum terminado ou não, respectivamente.

Enquanto estas operações são executadas, um monitor deve verificar o nível de memória disponível, onde fica responsável por executar as alocações de forma contígua, assim como verificar se o processo já está alocado, além da desalocação da Heap seguindo

o algoritmo Least Recent Used (LRU). Caso o processo não esteja alocado em Heap, o processo é carregado de memória subjacente.

Para a solução sequencial, utiliza-se somente um core para a CPU, a fim de realizar o processamento do processo assim como o seu quantum. Para a solução paralela, são definidas threads de processamento para cada core desejado na CPU, a partir de duas.

Caso o uso da memória supere o limite máximo, a memória swap implementada pode ser alocada como auxiliar, para a alocação dos processos.

O número de requisições, ao serem totalmente processados, definirá o final do programa.

Como diretrizes deste projeto, procurando atingir um bom nível de qualidade, os componentes foram arquitetados de forma a trabalharem modularmente, utilizando abstrações de arquitetura a fim de obter-se representações com características funcionais independentes, assim como o seu acoplamento fique em níveis aceitáveis. Cada classe do projeto foi criada com estes conceitos. As classes são:

- Alloc
  - Classe responsável por determinar o índice do processo alocado na Heap, assim como o seu offset;
- CPU
  - Classe construída para suprir ambas soluções, sequencial e multi-core, é capaz de armazenar uma lista de cores e selecionar aquele disponível para atender a requisição atual.
- Core
  - Representa um núcleo de processamento, executando as rotinas necessárias ao processo presente.
- Logger
  - Responsável por fazer os registros de movimentação de processos, assim como o seu timestamp. Executa os registros de alocação e desalocação em Heap, trocas de contexto entre filas, alocação e desalocação em swap.
- NoProcessException
  - Tem como função atuar como exceção na falha à busca de um processo, seja em qualquer memória.
- Swap
  - Atua como extensão da memória Heap. É constituída de uma lista de objetos da classe Proc, que simula um vetor de posições. Além dos métodos de adição e remoção, possui método de busca por ID do processo
- PCB
  - Está atribuído a esta classe, armazenar as filas de requisições de processos, assim como os métodos de adição, obtenção e mudança entre as filas.

- Proc
    - Responsável por definir individualmente as características de cada processo através de seu construtor, com seu tamanho, tempo de execução, sua identificação e fila atual.
- RandomProcessGenerator
    - Classe responsável por criar os objetos da classe Proc com suas características, de forma individualizada.
- AllocProcess
    - Classe de controle, que tem propriedade de herança juntamente a classe Alloc, representando a alocação do processo para sua execução através de uma flag de verificação.
- Heap
    - Possui uma lista de objetos que simulam o funcionamento da memória, com tamanho configurável ao iniciar o sistema. Possui os métodos de alocação First-Fit e desalocação com algoritmo Least Recent Used (LRU) internos. Utiliza um vetor de controle de espaços ocupados, assim como um threshold customizável para o algoritmo de desfragmentação. Tanto para o algoritmo de desfragmentação quanto para alocação e desalocação, são definidas threads para cada situação.
- Scheduler
    - O escalonador Round Robin foi selecionado para realizar o escalonamento dos processos coletados na fila de requisições ready do PCB. Ao coletar um processo desta fila, automaticamente este processo é transferido para a fila running do PCB e o processo é escalado para a CPU tratar sua execução. Com o resultado do processamento, o escalonador define para qual fila o processo será direcionado. Caso seu quantum seja totalmente consumido, o processo é direcionado para fila terminated. Caso contrário, é redirecionado para o final da fila ready, ambas no PCB.

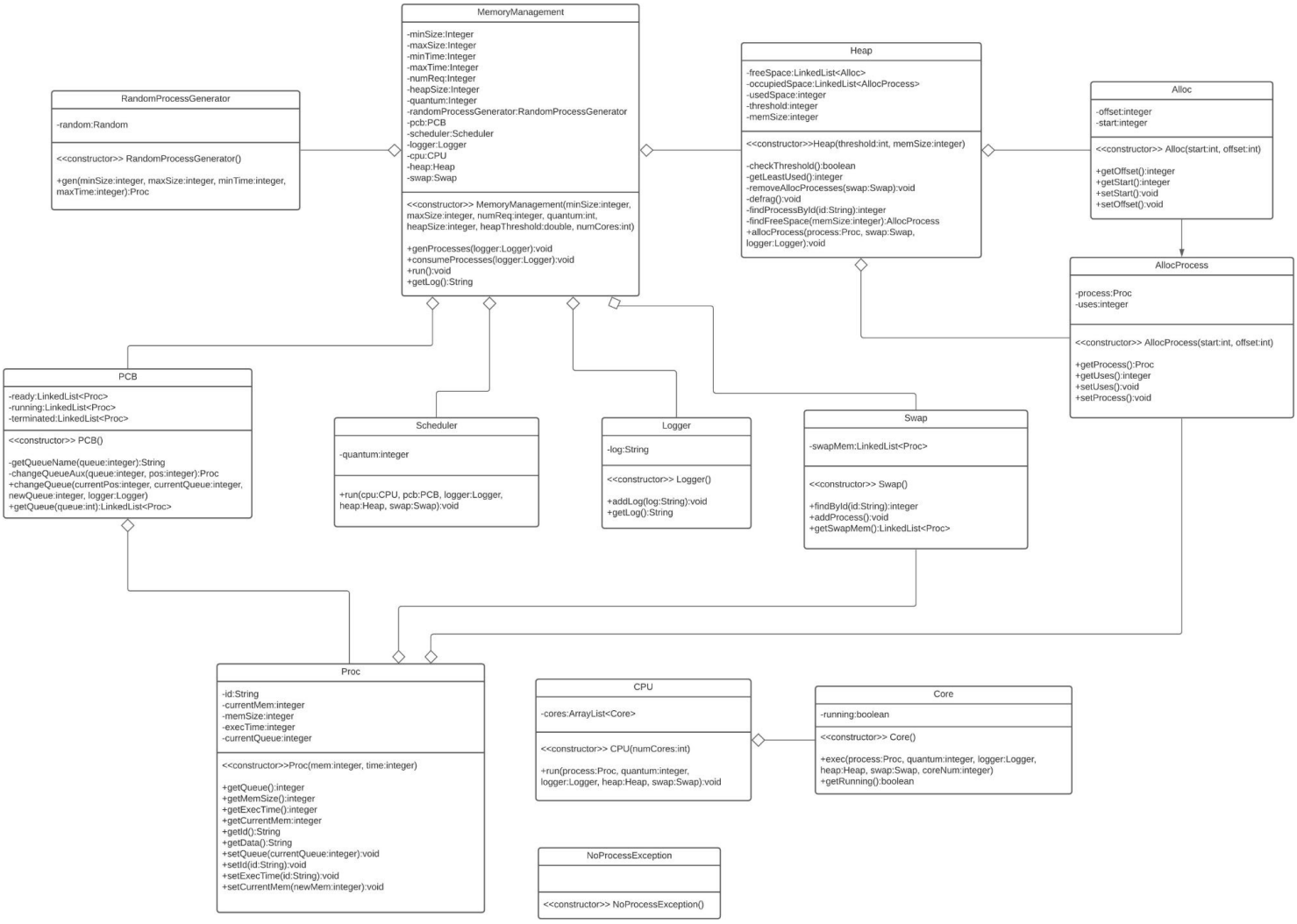
- MemoryManagement
    - Classe principal, responsável por atribuir os valores de entradas configurados entre todas as classes. Possui relação tem-um com grande parte das classes, assim como métodos capazes de definir qual solução será executada no momento, seja sequencial ou paralela.

unipampa

Universidade Federal do Pampa

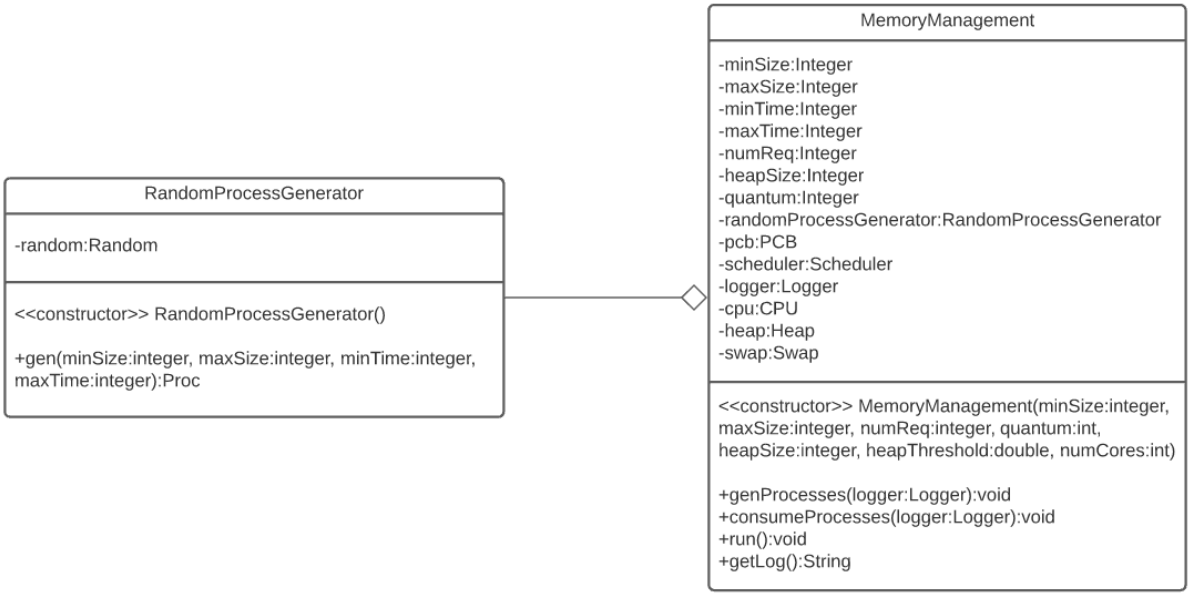
Diagrama de classes UML

O detalhamento das relações entre classes, pode ser observado através do diagrama de classes UML pela figura abaixo:

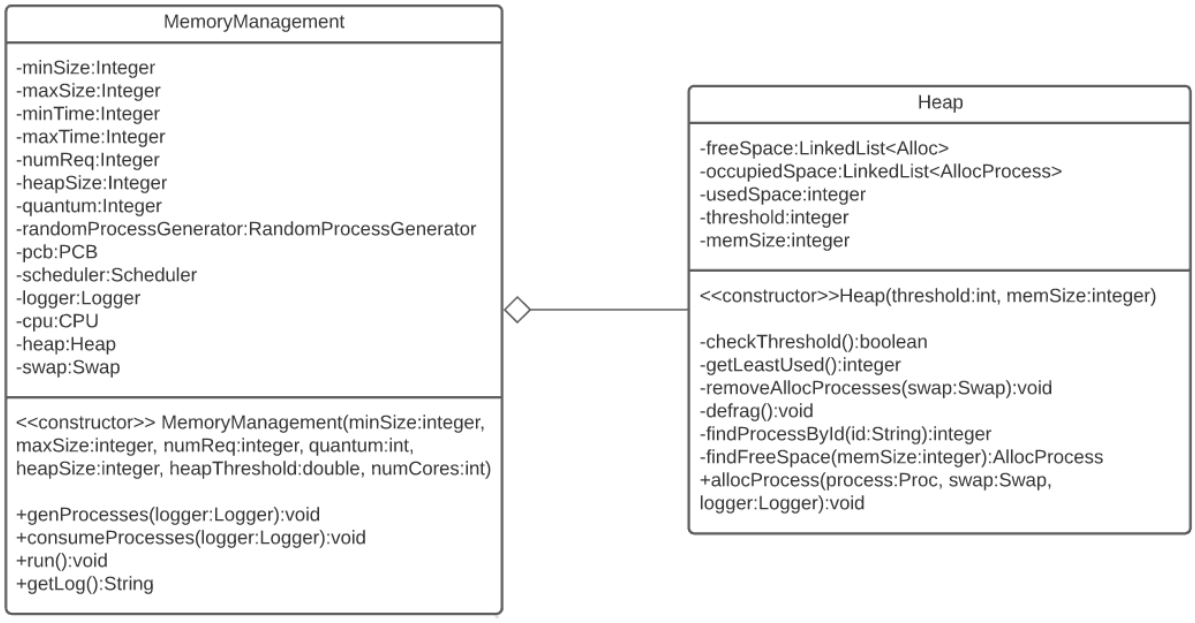


O detalhe de cada relacionamento, pode ser visto abaixo:

Relação entre RandomProcessGenerator e MemoryManagement

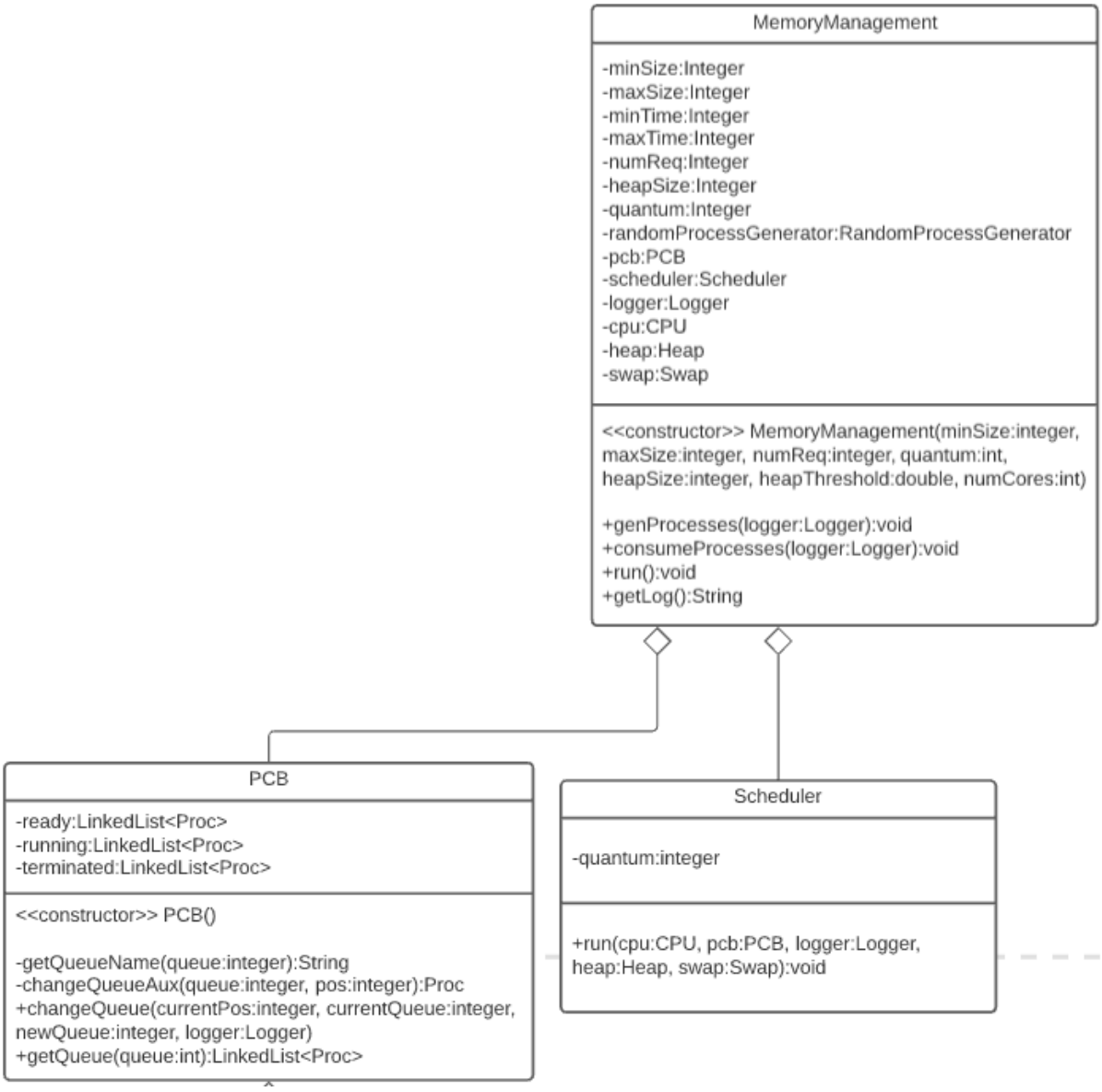


Relação entre MemoryManagement e Heap



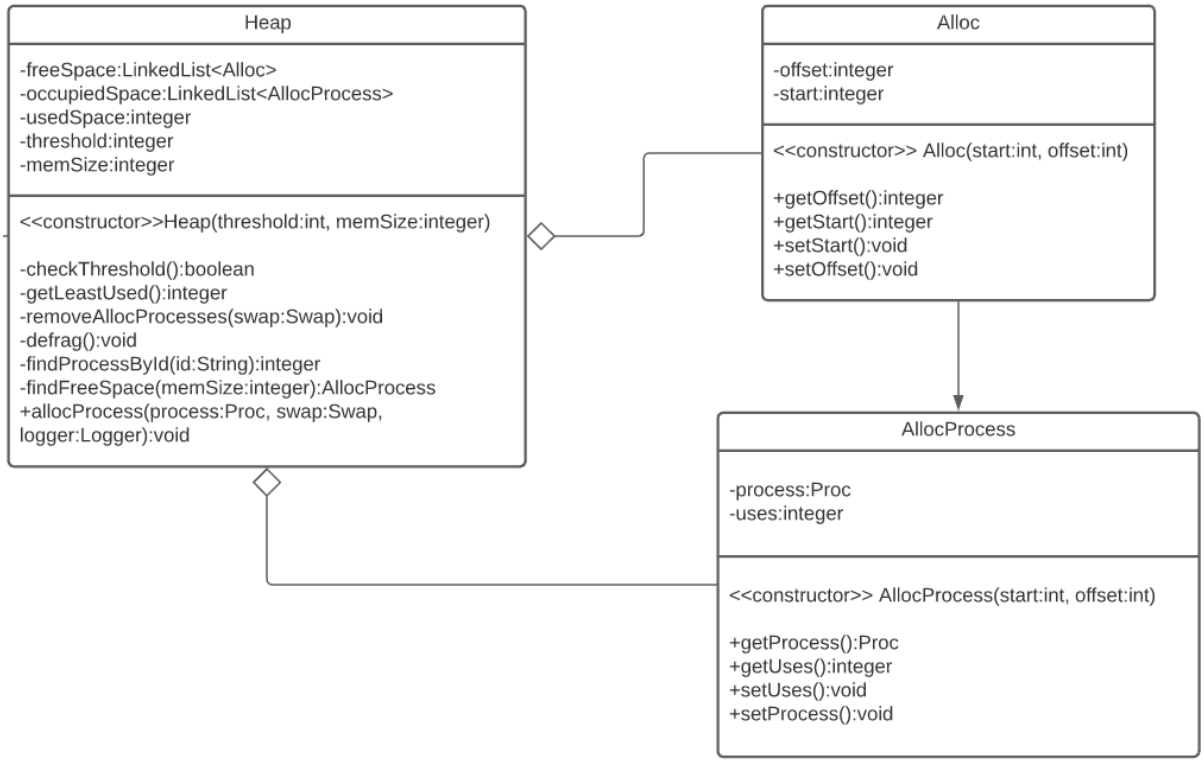
Universidade Federal do Pampa

Relação entre PCB, Scheduler e MemoryManagement

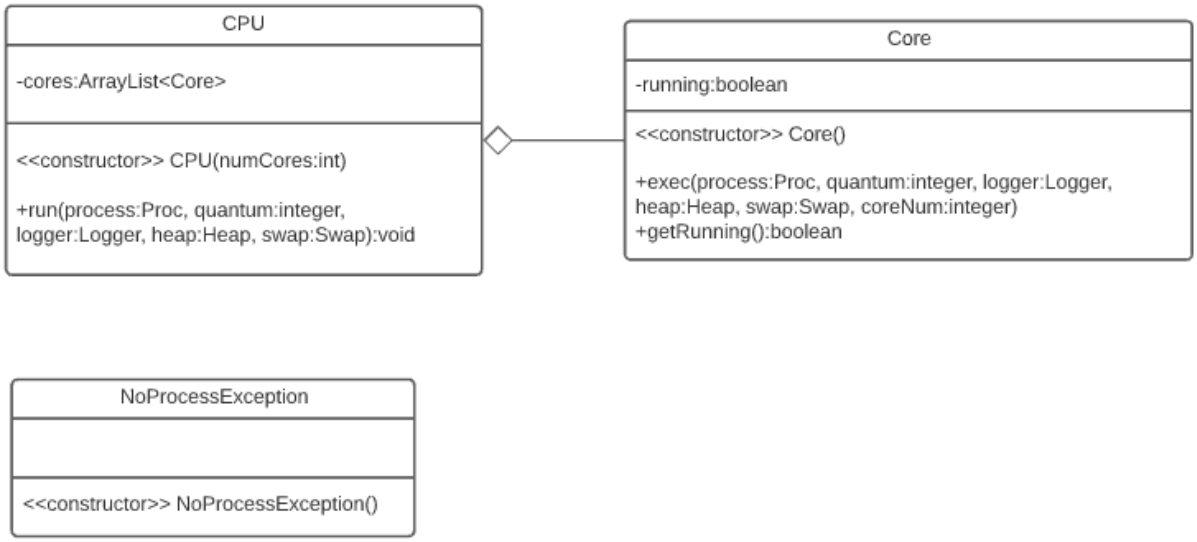


Universidade Federal do Pampa

Relação entre Heap, Alloc e AllocProcess

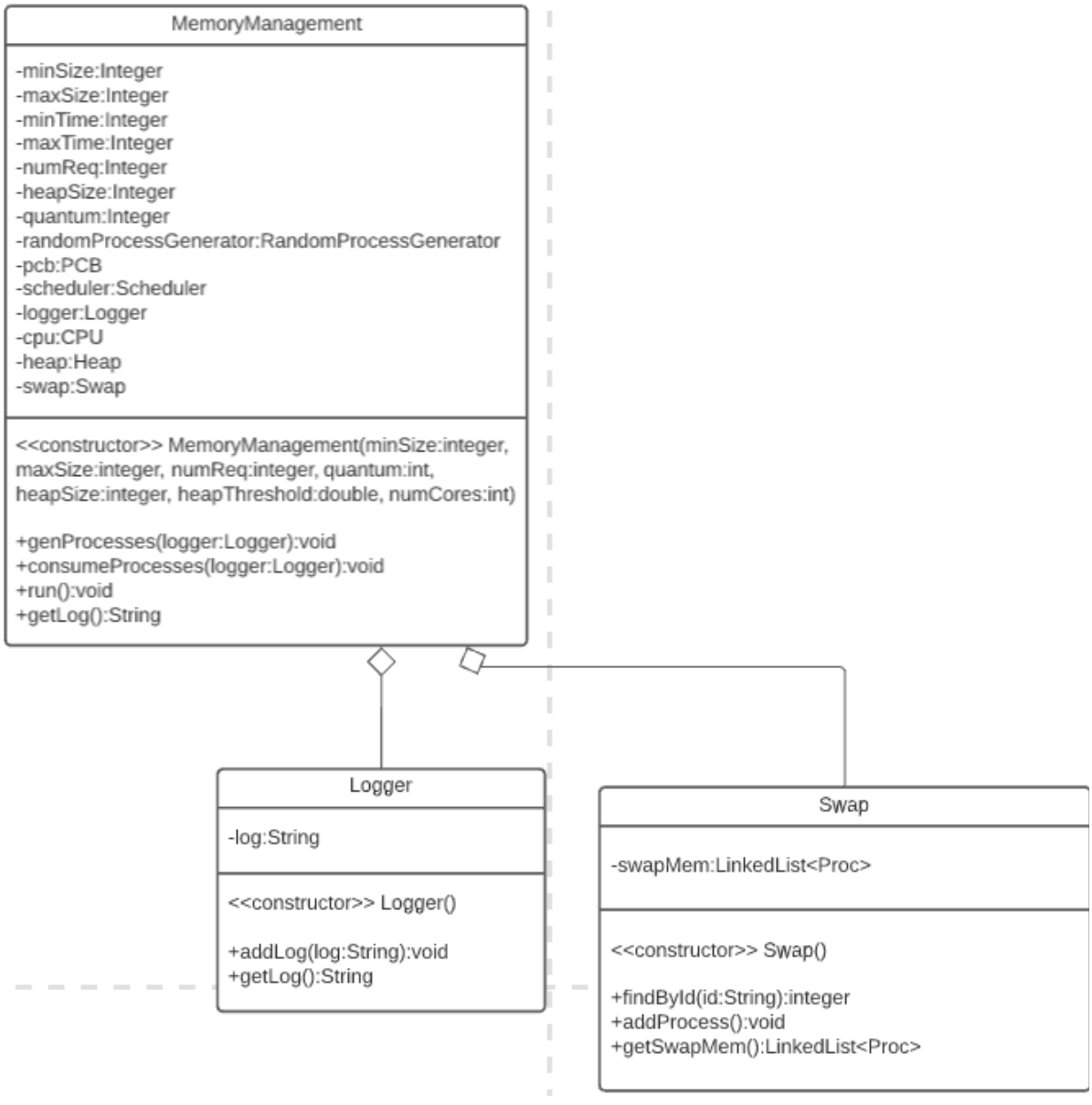


Relação entre CPU, Core e NoProcessException



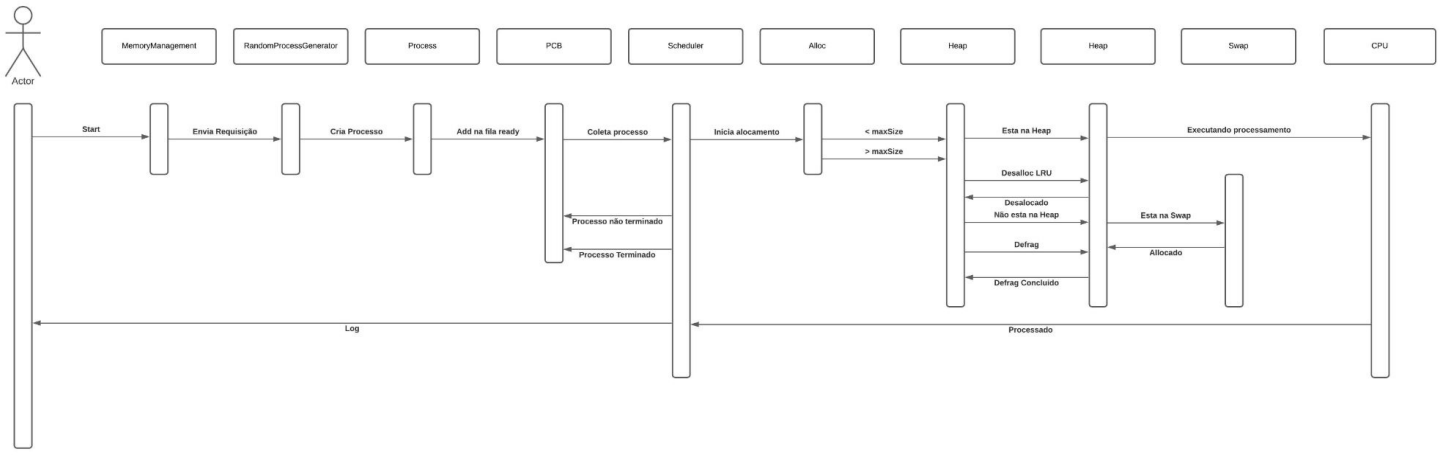
Universidade Federal do Pampa

Relação entre MemoryManagement, Logger e Swap

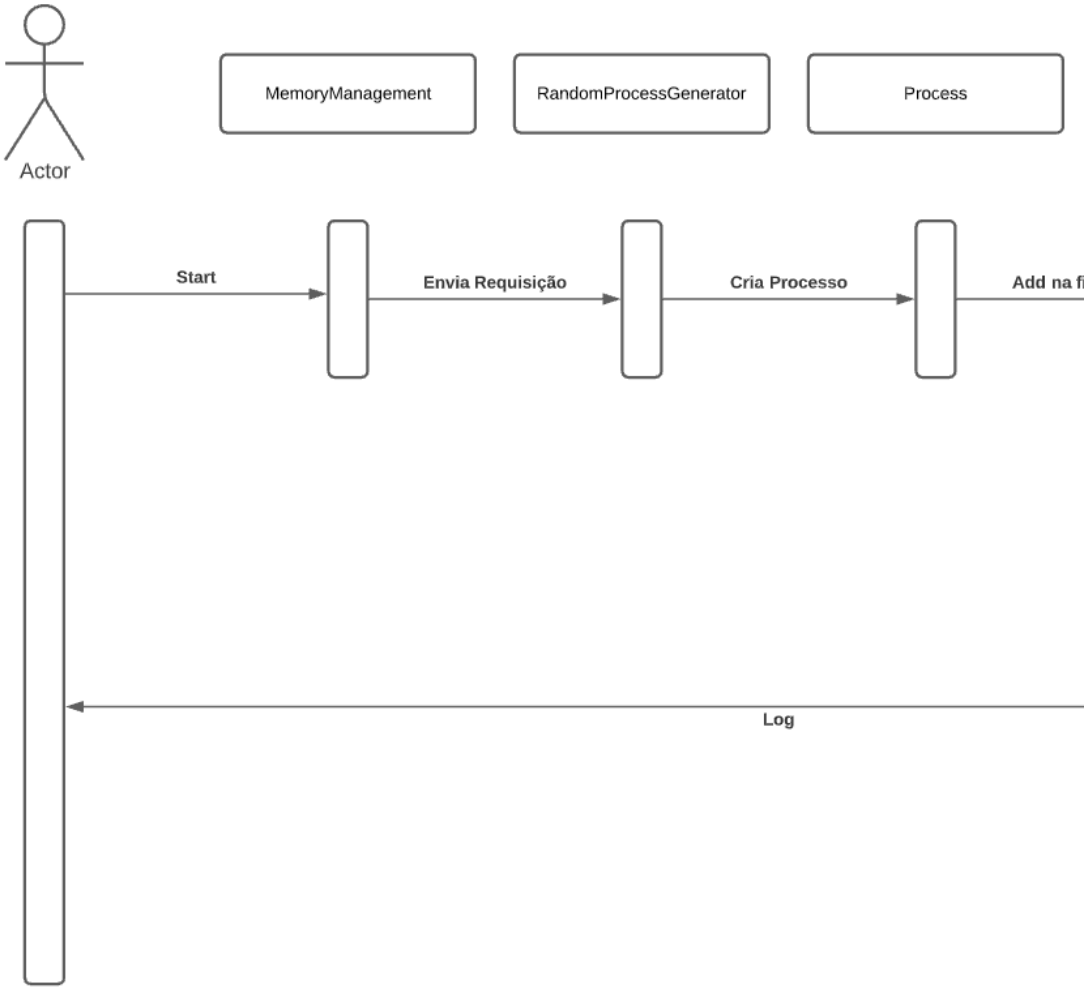


Universidade Federal do Pampa

Pelo diagrama de sequência, verifica-se o ciclo de execução de um processo. Este ciclo pode ser aplicado em ambas as soluções, tanto sequencial quanto paralela multithread.

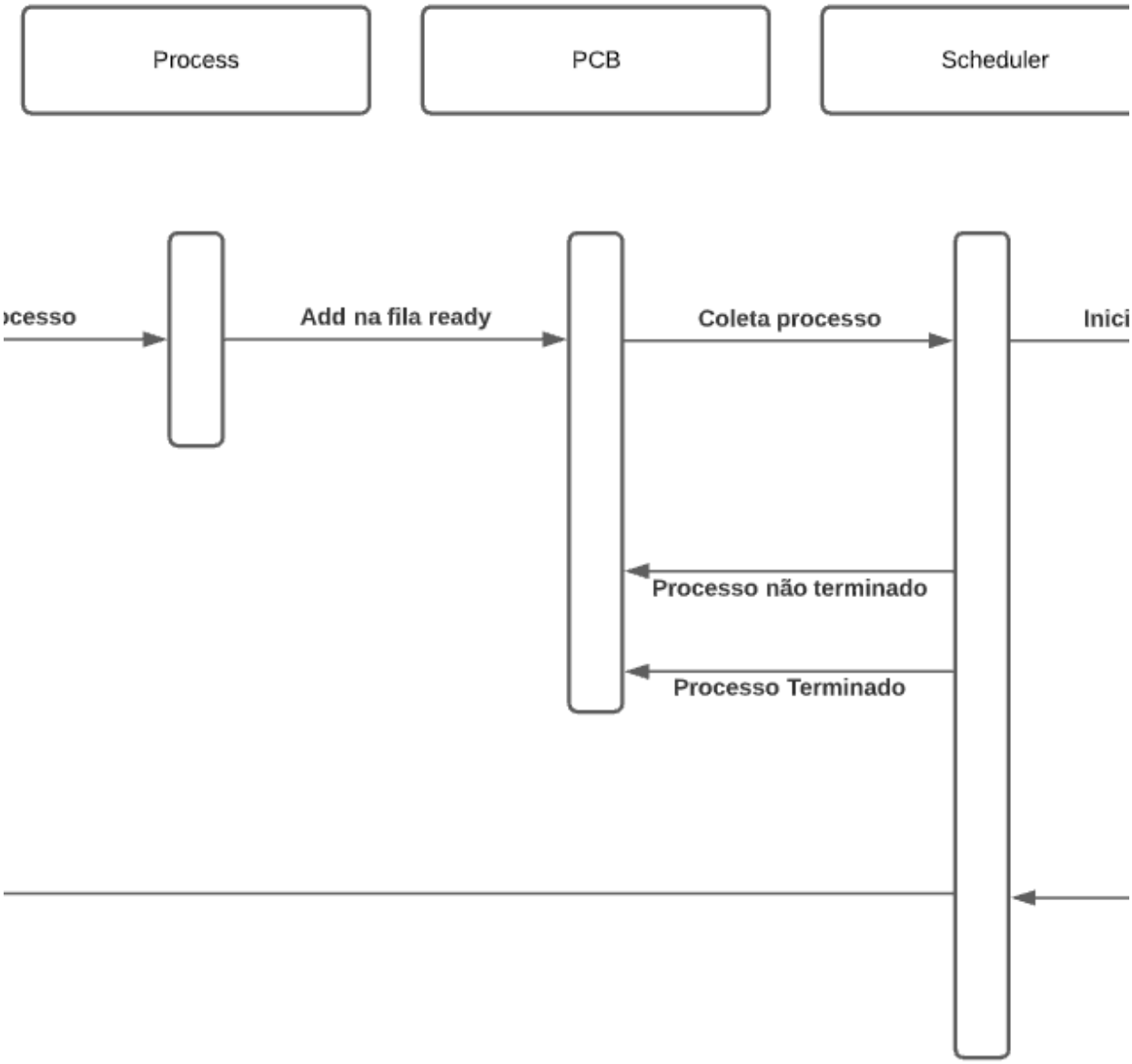


Abaixo, temos cada passo no detalhe:



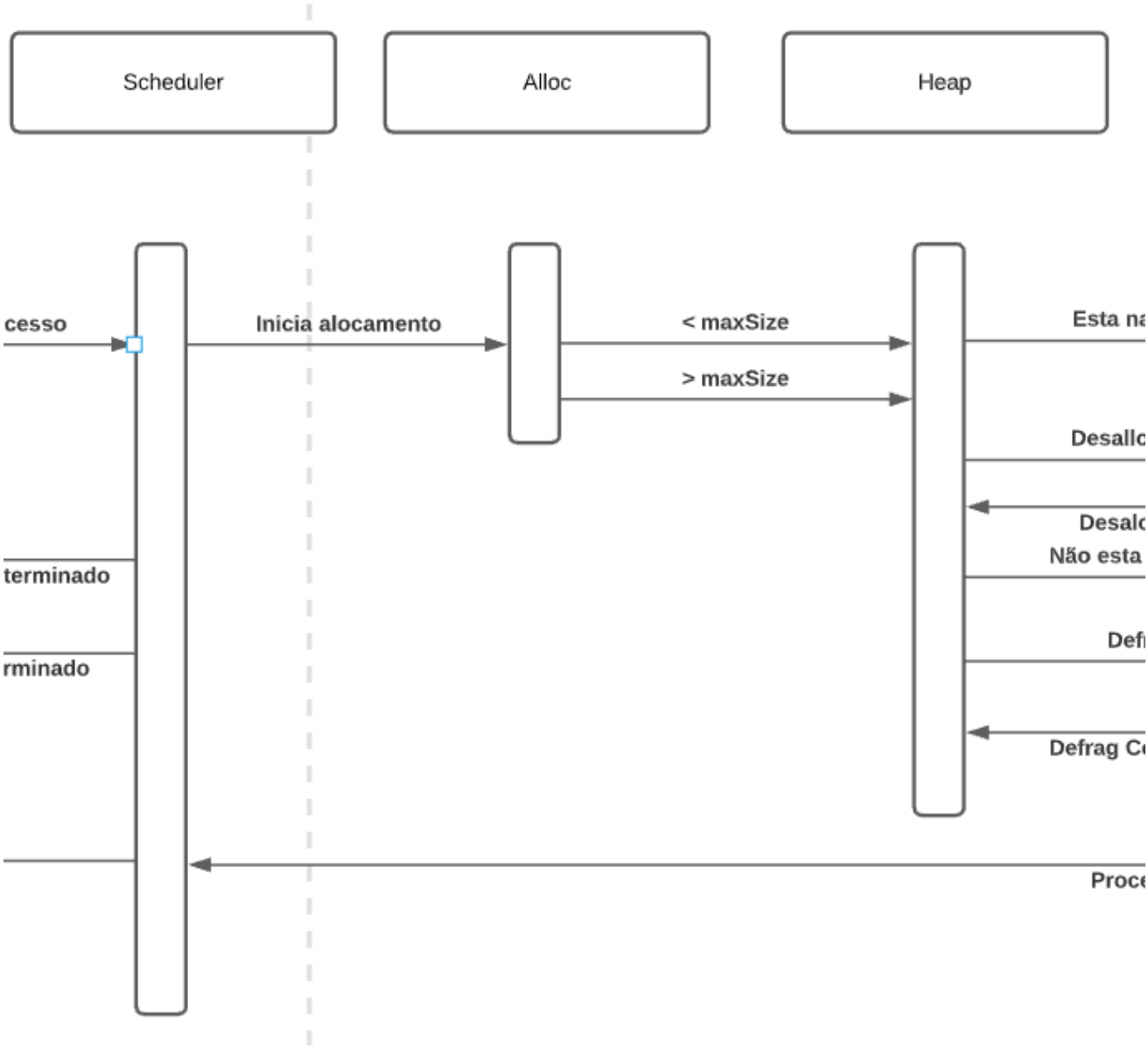
unipampa

Universidade Federal do Pampa



unipampa

Universidade Federal do Pampa



unipampa

Universidade Federal do Pampa