Artesanato de Aveiro

Sistemas Distribuídos – Trabalho 2 – Turma P2

Joel Pinheiro — 65151

Luís Assunção — 42967

3 de Maio de 2015

Conteúdo

Capítulo 1

Neste capítulo mostramos as parametrizações que existem na nossa solução, descrição das classes e diagrama de interação.

1 Parametrizações

- Nome do ficheiro de log
- Caminho do ficheiro de log
- Quantidade de matérias-primas em armazém
- Limite de produtos acabados na oficina para avisar a dona (recolha)
- Quantidade de produtos que a dona pode levar para a loja de cada vez
- Quantidade de matérias-primas que a dona pode levar para a oficina de cada vez
- Quantos artesões existem
- Quantos clientes existem
- Quantos produtos os artesões fabricam por matéria-prima
- Quanta matéria-prima os artesões levam para a sua mesa de trabalho (irá ser consumido para fabricar x produtos)
- Quanto tempo os artesões levam a fabricar x matérias-primas

Na nossa simulação é possível definir que x matérias-primas vão resultar em y produtos. Ao dizer que o artesão vai buscar 5 matérias-primas para a sua mesa e fabrica 1 produto estamos a dizer que são precisas 5 matérias-primas para produzir um único produto. Todas a matérias-primas na sua mesa são totalmente consumidas de uma vez.

Por outro lado, se dissermos que o artesão leva 3 matérias-primas, e produz 3 produtos, estamos a dizer que cada matéria-prima produz um produto. Podemos também definir que uma matéria-prima resulta em 5 produtos, ao definir que o artesão leva uma matéria-prima para a sua mesa, e produz cinco produtos por matéria-prima.

O programa de simulação é interativo, e irá guiar o utilizador por todos os parâmetros de configuração necessários.

2 Descrição das classes

2.1 EntrepeneurThread

```
public class EntrepreneurThread extends Thread{
   private final int amountProductsToPickUp;
   private final int numberOfReplenish;
   private String currentState;
   private final EntrepreneurStorageInterface STORAGEMONITOR;
   private final EntrepreneurStoreInterface STOREMONITOR;
   private final EntrepreneurWorkshopInterface
   WORKSHOPMONITOR;
}
```

2.2 CraftmanThread

```
public class CraftsmanThread extends Thread{
   private final int ID;
   private final CraftsmanWorkshopInterface WORKSHOPMONITOR;
   private int productStep;
   private int amountToCollect;
   private int elapsedTime;
   private String state;
}
```

2.3 CustomerThread

```
public class CustomerThread extends Thread{
   private final int ID;
   private final CustomerStoreInterface STOREMONITOR;
   private boolean insideTheStore;
   private boolean exitShop;
   private String state;
}
```

2.4 WorkshopMonitor

```
public class WorkshopMonitor implements
CraftsmanWorkshopInterface,
EntrepreneurWorkshopInterface{
    private int stockPrimeMaterials;
    private int stockManufacturedMaterials;
    private final WorkshopLogInterface LOGMONITOR;
    private final WorkshopStoreInterface STOREMONITOR;
    private int primeMaterialStockResuply;
    private int accumulatedSupliedPrimeMaterials;
    private int accumulatedProducedProducts;
    private boolean
    entrepreneurCalledReplenishPrimeMaterials;     private
    int productThreshold;
    private boolean entrepreneurCalledToCollectProducts;
    private int[] craftsmanAccumulatedManufacturedProducts;
}
```

2.5 StorageMonitor

```
public class StorageMonitor implements
EntrepreneurStorageInterface{
  private int stockPrimeMaterials;
  private final StorageLogInterface LOGMONITOR;
}
```

2.6 StoreMonitor

```
public class StoreMonitor implements
EntrepreneurStoreInterface,          CustomerStoreInterface,
WorkshopStoreInterface{
  private int numberOfAvailableProducts;
  private int customersInStore;
  private boolean isDoorOpen;
  private boolean primeMaterialAreNeeded;
  private boolean callCollectProducts;
  private Queue<Integer> customersThatWantToBuyQueue;
  private final StoreLogInterface LOGMONITOR;
  private int[] accumulatedBoughtProducts;
  private int[] amountToBuyCustomer;
}
```

2.7 LogMonitor

```
public class LogMonitor implements StoreLogInterface,
StorageLogInterface, WorkshopLogInterface {
   // Craftsman
   private String[] crafsmanState;
   private int[] craftsmanAccManProducts;
   // Store
   private String storeState;
   private int customersInside;
   private int goodsInDisplay;
   private boolean callTransferProducts;
   private boolean callTransferPrimeMaterials;
   // Customer
   private String[] customersState;
   private int[] customerAccumulatedBoughtGoods;
   // Workshop
   private int stockPrimeMaterials;
   private int stockFinnishedProducts;
   private int resuplyPrimeMaterials;
   private int totalAmountPrimeMaterialsSupplied;
   private int totalAmountProductsManufactured;
   // Entrepreneur
   private String entrepreneurState;
   // Storage
   private int StorageHouseStock;
   private String fileName;
   private String filePath;
   private String previousLine;
   private int maxSpaceToIntegers;
}
```

2.8 StoreMonitor

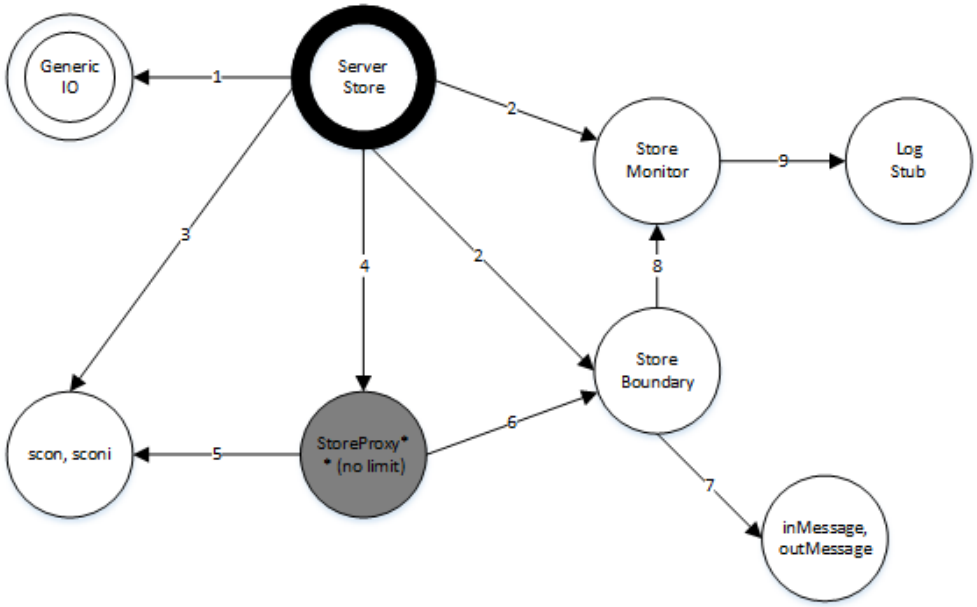
```
public class StoreMonitor implements
EntrepreneurStoreInterface,          CustomerStoreInterface,
WorkshopStoreInterface{
  private int numberOfAvailableProducts;
  private int customersInStore;
  private boolean isDoorOpen;
  private boolean primeMaterialAreNeeded;
  private boolean callCollectProducts;
  private Queue<Integer> customersThatWantToBuyQueue;
  private final StoreLogInterface LOGMONITOR;
  private int[] accumulatedBoughtProducts;
  private int[] amountToBuyCustomer;
}
```

3 Diagramas de Interação

Legenda:

3.1 Lado do Servidor

3.1.1 Store



1 – writeInString

2 - instantiate

3 - instantiate, start, accept

4 - instantiate, start

5 - readObject, writeObject, close

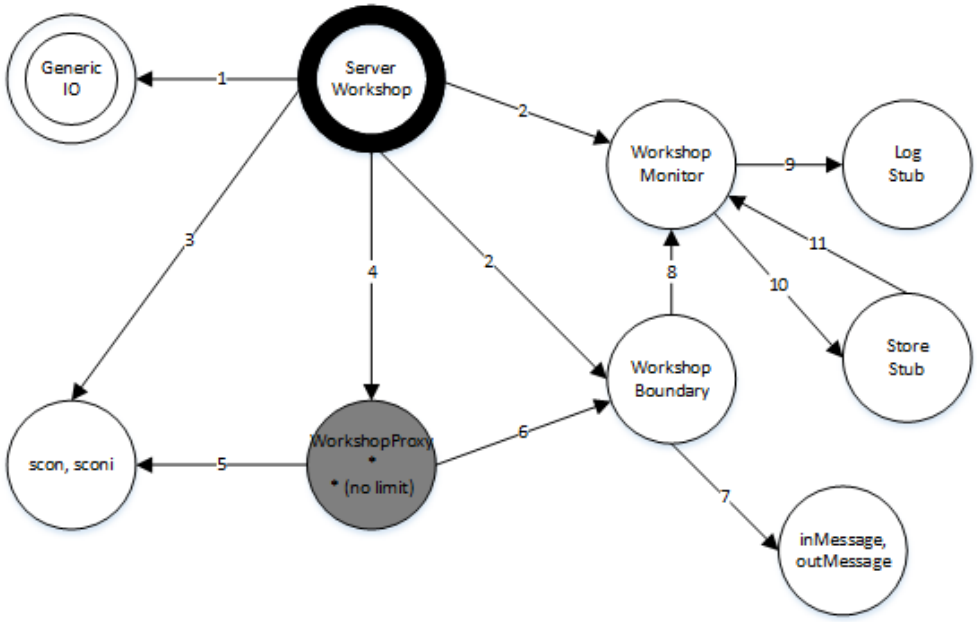
6 - processAndReply

7 - instantiate, getType, getId, get(...)

8 – set(...)

9 - instantiate

3.1.2 Workshop



1 – writeInString

2 - instantiate

3 - instantiate, start, accept

4 - instantiate, start

5 - readObject, writeObject, close

6 - processAndReply

7 - instantiate, getType, getId, get(...)

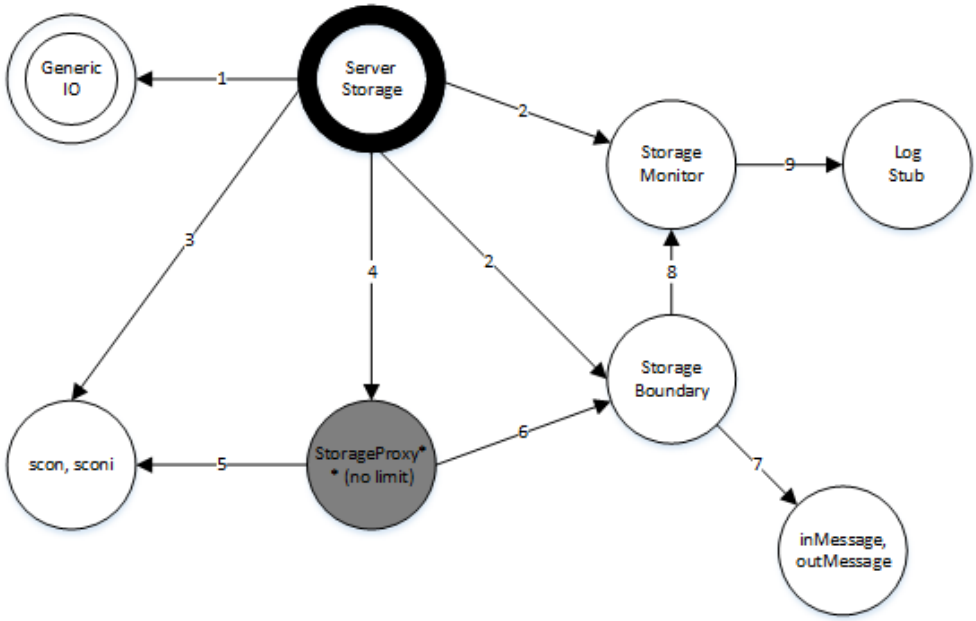
8 – set(...)

9 - instantiate

10 - instantiate

11 - getID, get(...)

3.1.3 Storage



1 – writelnString

2 - instantiate

3 - instantiate, start, accept

4 - instantiate, start

5 - readObject, writeObject, close

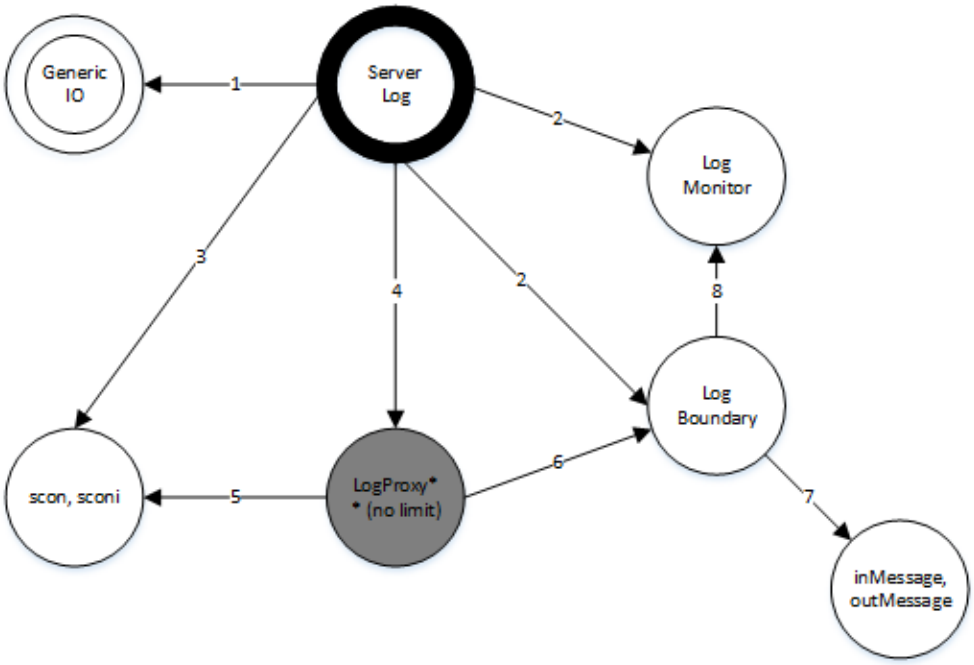
6 - processAndReply

7 - instantiate, getType, getId, get(...)

8 – set(...)

9 - instantiate

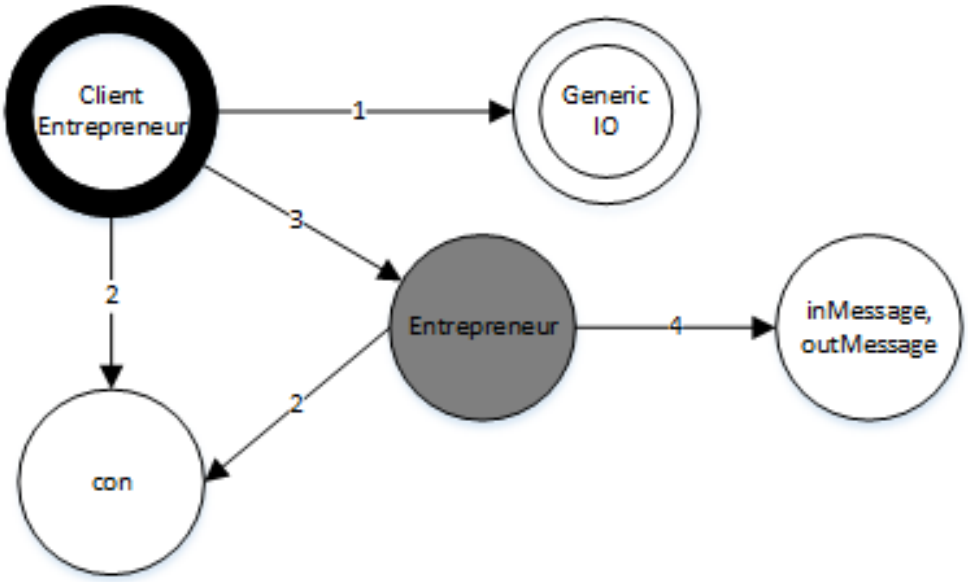
3.1.4 Log



1 – writeInString
2 - instantiate
3 - instantiate, start, accept
4 - instantiate, start
5 - readObject, writeObject, close
6 - processAndReply
7 - instantiate, getType, getId, get(...)
8 – set(...)
9 - instantiate

3.3 Lado do Cliente

3.3.1 Entrepreneur



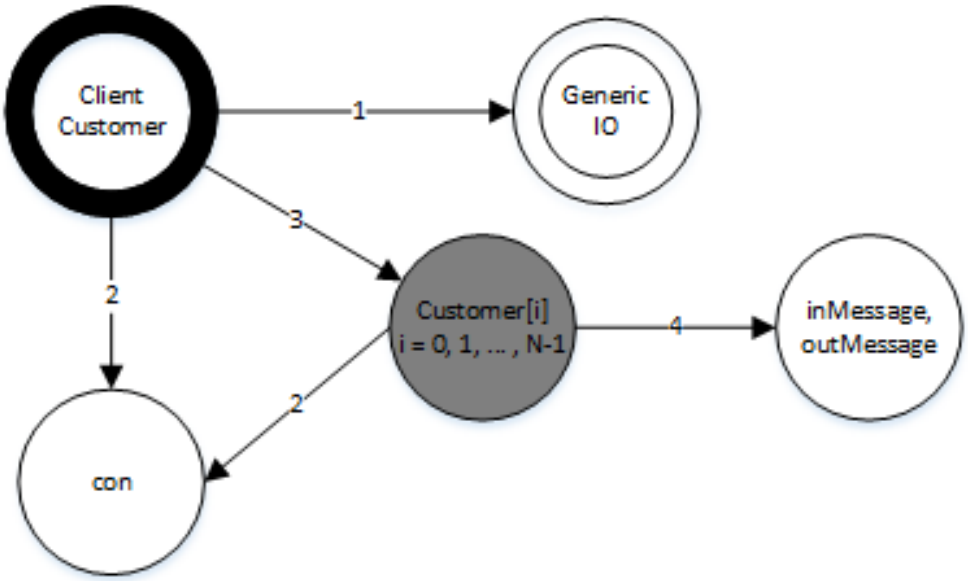
1 - readlnInt, readlnString, writeString, writelnString

2 - instantiate, open, readObject, writeObject, close

3 - instantiate, start, join

4 - instantiate, getType

3.3.2 Customer



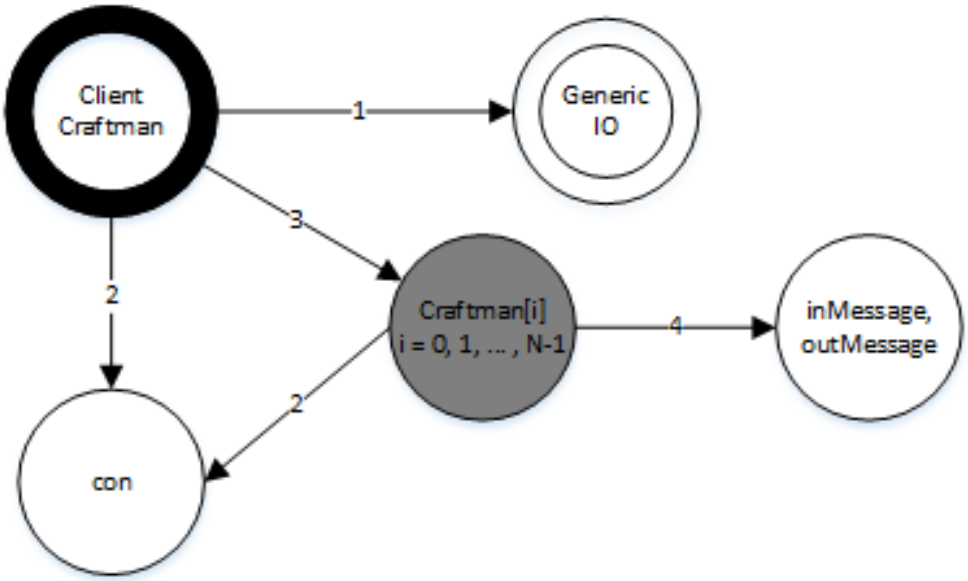
1 - readlnInt, readlnString, writeString, writelnString

2 - instantiate, open, readObject, writeObject, close

3 - instantiate, start, join

4 - instantiate, getType

3.3.3 Craftman



1 - readlnInt, readlnString, writeString, writelnString

2 - instantiate, open, readObject, writeObject, close

3 - instantiate, start, join

4 - instantiate, getType