

# **Trabalho Prático - Programação Modular**

**Allan V. Mori<sup>1</sup>, Bruno L. Sousa<sup>1</sup>**

<sup>1</sup>Departamento de Ciência da Computação – Universidade Federal de Minas Gerais (UFMG)  
UFMG – Avenida Antônio Carlos, 6627, 31.270-010, Belo Horizonte, MG, Brasil.

allanmori@ymail.com, bruno.luan.sousa@gmail.com

## **1. Introdução**

Um sistemas de telefonia tem por objetivo gerenciar ligações entre assinantes. Para isso é necessário prover uma infraestrutura capaz de suportar diversas operações e falhas que possam ocorrer. Neste trabalho foi implementado um sistema de telefonia que desempenha varias funções, como adição de assinantes, ligação entre assinantes, adição de centrais, procura de rotas, geração de contas entre várias outras que simulam um ambiente real e controlado que pode ser assistido através de seus registros de controle. Na implementação do sistema, foi considerado os requisitos pré-levantados no enunciado do trabalho e adaptações baseadas no mundo real para o correto funcionamento deste sistema.

Devido ao grande número de funcionalidade necessárias para o sistemas funcionar como planejado, foram utilizados diagramas para expressar de maneira objetiva a arquitetura e buscar atingir um dos principais requisitos para programas que são grandes e escaláveis, a Modularidade. Com essa meta, foram utilizados recursos que possibilitam planejar e implementar incrementalmente para diminuir o acoplamento e aumentar a coesão, características essenciais para o correto desenvolvimento do projeto e programação do sistema proposto. Nesse contexto a aplicação de padrões de projetos, foi um dos pilares no planejamento deste sistema.

Neste trabalho definimos o sistema de gerenciamento de telefonia sendo uma rede heterogênea que conecta as centrais telefônicas aos assinantes, com cada assinante conectado a uma única central, que é capaz de receber seus pedidos e encaminhá-los para o restante da rede. A comunicação começa com o pedido de conexão, esta solicitação é roteada através da rede. Se existir caminho de conexão, a chamada será realizada e desta forma o caminho se torna exclusivo para o receptor e transmissor até que a ligação seja concluída. Se não existir caminho, ou a linha já estiver ocupada, ou ainda a central está temporariamente fora de serviço, esta ligação não pode ser completada. Além das ligações este sistema realiza o gerenciamento de criação e exclusão de novas linhas para assinantes e outras centrais. Outra parte importante para um sistema de telefonia é a capacidade de gerar contas com periodicidade para seus assinantes.

Nas próximas seções serão detalhadas as arquiteturas utilizadas, os padrões de projeto aplicados. Na Seção 3 é demonstrado o funcionamento deste sistema e alguns testes com entradas variadas aplicadas sobre o sistemas cobrindo os critérios solicitados.

## **2. Arquiteturas e Padrões de Projetos Utilizados**

No desenvolvimento deste trabalho foram necessários artefatos que sustentassem a implementação de forma consistente. Os diagramas desta seção mostram as abstrações

Outro ponto que vale destacar foi o desenvolvimento utilizando a técnica de código comentado para que os desenvolvedores sempre estivessem amparados sobre o que uma classe ou método pode fazer e deve fazer, evitando eventuais problemas de comunicação entre os desenvolvedores.

Na Figura 1, é apresentado o diagrama que contém os principais elementos implementados no Sistema de Gerenciamento de Telefonia. Começando pelo pacote "search", neste foi utilizado o padrão visitor em sua arquitetura, pois este padrão desacopla a utilização da busca implementada, permitindo que o sistema possa ser estendido com outras implementações de busca, seguindo os princípios e práticas recomendadas de arquitetura de software.



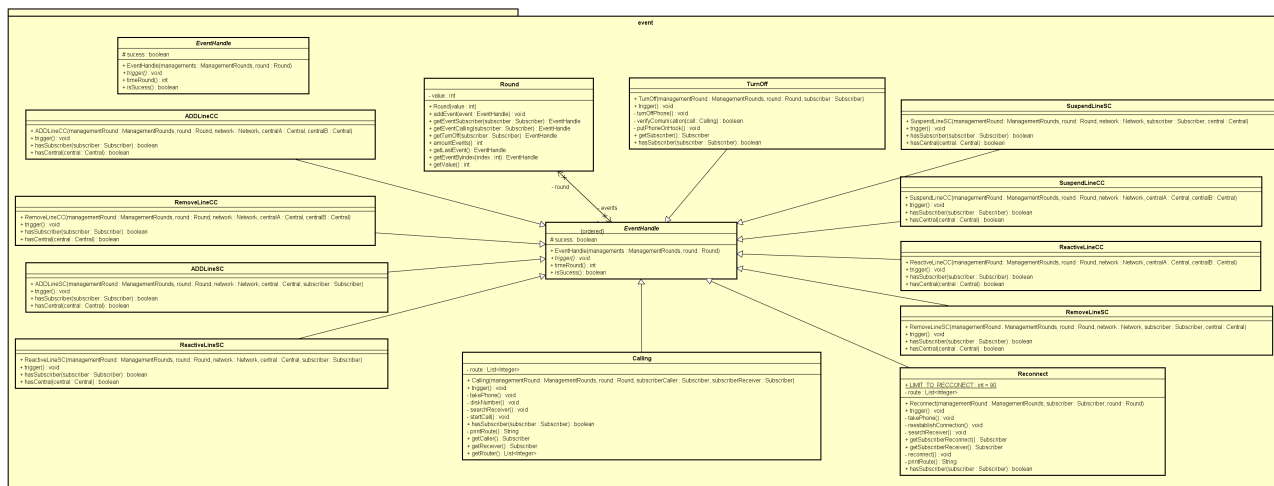
Seguindo da esquerda para a direita temos a parte central da aplicação, com os pacotes que gerenciam os eventos localizados no centro e acima, o pacote "account" contém classes, que tem por principal função o gerenciamento e a contabilização de pulsos. Neste

mesmo pacote também são implementados padrões como no pacote "event" onde está implementado uma fábrica dos objetos relacionados com os requisitos pré definidos para este tipo de sistema, Figura 2.

No centro e abaixo o pacote "structure" encontra-se as classes que pertencem a estrutura do sistema e gerenciamento da rede, mantendo o funcionamento e manutenção da rede gerada. Já na parte direita desse diagrama temos os pacotes que realizam a parte de ignição do sistema no pacote "programacaomodular" e o pacote que realiza a leitura dos símbolos na entrada do sistema. Algumas classes foram suprimidas dessa representação, pois interferiam na visualização do diagrama, além do pacote EventHandler que está na próxima subseção.

## 2.2. Diagrama de Classes para o EventHandler

A Figura 2 apresenta o diagrama de Classes para EventHandler, onde estão implementadas as fábricas que podem criar novas linhas, entre outras funcionalidades que fazem parte dos requisitos desse sistema. Vale destacar a classe Round, pois esta classe foi criada com o objetivo de controlar a passagem de períodos de tempo dentro da simulação, desta forma a cada round acontecem eventos e a cada indicação de troca de round é indicado qual o tempo que passou em unidades de tempo, ou como considerado semanticamente, segundos. Essas considerações foram úteis para a geração de contas e medição de pulsos utilizados pelos assinantes, permitindo assim ter uma emissão de fatura por assinantes e período de tempo pré definido.



### Figura 2. Diagrama de Classes - EventHandler

### 2.3. Padrões de Projetos utilizados

Devido ao tamanho do projeto e a necessidade de expansão com consistência, foram aplicados os padrões de projetos para auxiliar na sustentação e desenvolvimento incremental. Abaixo uma breve definição dos padrões e seu uso dentro deste projeto.

### **2.3.1. Command**

Esse padrão tem por objetivo oferecer um maior controle da execução de uma requisição. Para isso Encapsula uma solicitação como objeto, para permitir parametrizar o cliente com diferentes solicitações, enfileirando ou fazendo o registro de solicitações e suportando operações que podem ser desfeitas. Sua principal vantagem é a facilidade de extensão da arquitetura, permitindo adicionar novos commands sem efeitos colaterais. Além de um bom nível de desacoplamento entre objetos, separando os objetos que possuem os dados dos que manipulam os dados.

### **2.3.2. Builder**

No Builder o objetivo é separar a construção de um objeto complexo de sua representação de modo que o mesmo processo de construção possa criar diferentes representações, permitindo separar os passos de construção de um objeto em pequenos métodos. Este padrão está sendo utilizado para a criação da rede, ele é representado pelo método `init()` da classe `ManageNetwork`.

### **2.3.3. Chain of Responsibility**

Para encadear os objetos receptores, passando a solicitação ao longo da cadeia até que um objeto a trate, este padrão foi utilizado ao longo da implementação, evitando o acoplamento do remetente de uma solicitação ao seu receptor, pois quem puder tratar a solicitação irá tratá-la, permitindo que mais de um objeto trate essa solicitação. A estrutura usualmente descrita é uma classe que possui cadeia de objetos, adaptamos no sistema para utilizar uma lista de objetos ao invés de existir um objetos dentro do outro.

### **2.3.4. Factory Method**

O padrão Factory Method facilita a inclusão de novos objetos. Não é necessário alterar código, apenas criar a assinatura do objeto e a sua fábrica, assim todo o código já escrito não precisará de alteração. O pacote `event` possui as classes que representam os eventos em si, e o pacote `event.Factory` contém as classes que implementam as fábricas de cada um dos eventos, como é possível observar nos diagramas acima.

### **2.3.5. Visitor**

Para representar uma operação a ser executada nos elementos de uma estrutura de objetos o visitor permite definir uma nova operação sem mudar as classes dos elementos sobre os quais opera. Neste projeto foi utilizado sobre a estrutura que permite realizar a busca de um assinante, evitando alteração nas chamadas de busca, caso sejam implementadas novas buscas. Atualmente esta é representada pela classe `Visitor` e o `DepthFirstSearch`, sendo os clientes que utilizam essa classe, `Calling` e `Reconnect`.

### 2.3.6. Facade

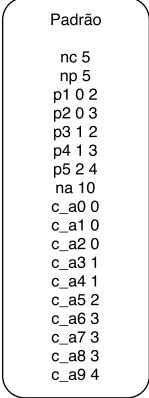
Facade define uma interface de nível mais alto que torna o subsistema mais fácil de ser usado. Este padrão também pode fornecer uma interface unificada para um conjunto de interfaces num subsistema. No sistema é representada pela classe SystemFacade, que unifica vários módulos do sistema com um acesso unificado para o cliente. Os módulos que são unificados: Gerenciamento da Rede, Módulo de Leitura, Gerenciamento dos Rounds, Emissão de Contas e Criação dos Arquivos de Saída.

### 2.4. Tratamento de Exceções

Na implementação realizada as exceções são tratadas pelo código, não havendo especificamente um módulo para isso. O motivo para isso, é que esse requisito eleva a robustez do sistema e também sua complexidade. Os principais tratamentos de exceções estão relacionados aos erros e exceções mais comuns, como erros de leitura, a não criação ou existência de um objeto ou a emissão de contas, para isso são realizadas verificações em lugares pontuais do código. No caso das exceções relacionadas com o sistema de telefonia os tratamentos devidos e seguem a mesma ideologia, sendo tratadas quando necessárias durante o código em lugares estratégicos.

## 3. Demonstração de Funcionamento e seus Artefatos

Nesta seção são apresentados os cenários requisitados nos critérios de avaliação. Nós primeiros tópicos serão abordados os principais artefatos utilizados na entrada que são gerados de saída. Para padronizar, a configuração do sistema nos casos de teste. A Figura 3 apresenta os comandos iniciais que serão padrões para execução de cada teste.



```
Padrão  
  
nc 5  
np 5  
p1 0 2  
p2 0 3  
p3 1 2  
p4 1 3  
p5 2 4  
na 10  
c_a0 0  
c_a1 0  
c_a2 0  
c_a3 1  
c_a4 1  
c_a5 2  
c_a6 3  
c_a7 3  
c_a8 3  
c_a9 4
```

**Figura 3. Configuração Padrão de Execução dos Testes**

### 3.1. Arquivo de Entrada

Para padronizar a entrada de dados o programa recebe como entrada um arquivo de texto contendo a definição do grafo que representa o sistema telefônico. Neste arquivo existe uma sequência de eventos por round. A cada linha do arquivo de entrada existe uma tag que indica o tipo de informação que será fornecida. Durante a próxima seção mais detalhes serão fornecidos através de exemplos de execução.

### 3.2. Executando o Sistema

O sistema de telefonia foi implementado na linguagem de programação Java. Sendo assim, foi criado um arquivo de formato `jar`, cujo qual corresponde ao executável desse programa. Esse arquivo está situado dentro da pasta `dist` do projeto. Sendo assim, para executá-lo e realizar testes nesse programa, é necessário seguir os seguintes passos:

- certifique-se que a máquina virtual Java (JVM), versão 7 ou superior esteja instalada em seu computador. Caso não esteja instalada, efetue sua instalação;
- abra o terminal Linux ou console do Mac ou Windows;
- pelo terminal ou console, vá até a pasta onde está localizado o arquivo `.jar` do programa;
- dentro desse diretório execute o comando de execução Java, passando o arquivo de entrada para o programa. Certifique-se que o arquivo de entrada fornecido está correto e não possui inconsistências. O comando `java` utilizado para executar a aplicação é mostrado no final dessa seção;
- após a execução do programa, são gerados três arquivos `txt`, no mesmo diretório cuja aplicação `.jar` está armazenada. Esses arquivos são as saídas emitidas pelo programa. Essas saídas referem-se a: eventos, faturas e sinais.

Ao utilizar essa aplicação, o usuário deve fornecer um arquivo `txt` contendo as informações de entrada. O comando utilizado para executar o arquivo `jar` é o seguinte:

```
java -jar ProgramacaoModular.jar arquivo_de_entrada
```

### 3.3. Arquivos de Saída

Ao terminar a execução do sistema, três novos arquivos de texto existirão. No arquivo "Sinais.txt" poderão ser encontrados informações sobre os sinais de controle que o sistema utiliza, como no caso de um telefone ser retirado do gancho. Em "Eventos.txt" existirão a lista de eventos requisitados durante o funcionamento do sistema, facilitando a conferência de resultados e comparação com a entrada fornecida. O terceiro arquivo "Fatura.txt", é o arquivo de saída em que estarão contabilizadas os pulsos telefônicos.

### 3.4. Ligações e Sinais

Para mostrar a dinâmica das ligações as Figuras 4 e 5 possuem respectivamente, os comandos de entrada e o resultado. Uma das regras propõe que seja respeitado o fim da ligação após 90s caso o receptor desligue e permita ao receptor retomar a ligação que desligou em até 90s. É também importante que um sistema seja transparente em suas operações para o usuário com sinais de aviso e para a empresa com sinais de controle e situação da rede e suas linhas, sendo possível a realização da auditorias para atestar o correto funcionamento. As Figuras desta seção mostram exemplos do envio de sinais aos terminais. Nas Figuras 6 e 7 é possível verificar que esse sistema respeita a regra de tempo máximo de religação apresentando respectivamente a entrada executada e a saída para uma tentativa de religação mal sucedida.

Exemplo de  
religação aos 90s

em 4  
r 1  
e 0 5 9  
r 2  
e 0 2 8  
r 5  
e 1 9  
r 80  
e 2 9  
r 82  
e 1 5  
e 1 8

**Figura 4. Entrada para Demonstração de Religações**

----- Eventos acionados -----

Evento 0: Iniciar Ligação entre o Assinante 5 e o Assinante 9  
Evento 1: Iniciar Ligação entre o Assinante 2 e o Assinante 8  
Evento 2: Assinante 9 solicitou um evento de desligar uma ligação  
Evento 3: Assinante 9 deseja reconectar sua última ligação  
Evento 4: Assinante 5 solicitou um evento de desligar uma ligação  
Evento 5: Assinante 8 solicitou um evento de desligar uma ligação

----- Sinais retornados -----

----- Evento 0 -----  
Telefone Retirado do Gancho...  
Discando Número...  
Completando Ligação...  
Rota da Ligação: 2 - 4  
Ligação Completada!

----- Evento 1 -----  
Telefone Retirado do Gancho...  
Discando Número...  
Completando Ligação...  
Rota da Ligação: 0 - 2 - 1 - 3  
Ligação Completada!

----- Evento 2 -----  
Ligação Desligada pelo Assinante 9!  
Telefone Colocado no Ganho...

----- Evento 3 -----  
Telefone Retirado do Ganho...  
Verificando Rota...  
Rota Encontrada: 4 - 2  
Ligação entre assinante 9 e 5 reconectada.

----- Evento 4 -----  
Ligação Desligada pelo Assinante 5!  
Telefone Colocado no Ganho...

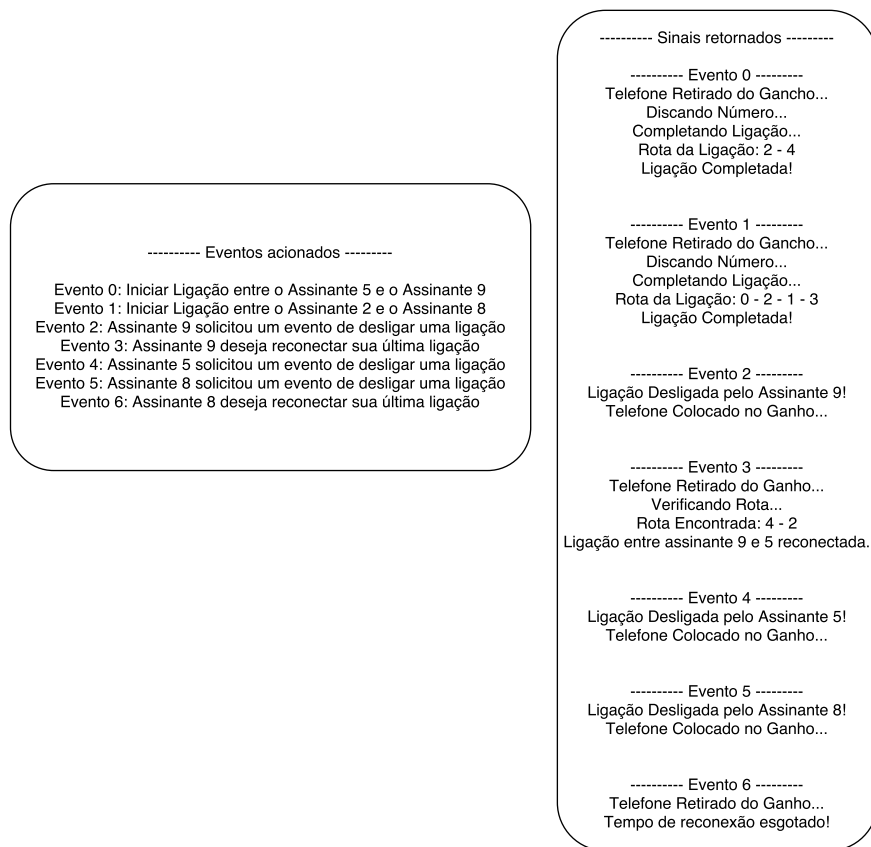
----- Evento 5 -----  
Ligação Desligada pelo Assinante 8!  
Telefone Colocado no Ganho...

**Figura 5. Saída para Demonstração de Religações**

Tentativa de  
religação depois de  
90s

em 4  
r 1  
e 0 5 9  
r 2  
e 0 2 8  
r 5  
e 1 9  
r 80  
e 2 9  
r 82  
e 1 5  
e 1 8  
r 190  
e 2 8

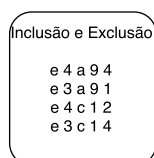
**Figura 6. Entrada para Demonstração de Religações Mal Sucedidas**



**Figura 7. Saída para Demonstração de Religações Mal Sucedidas**

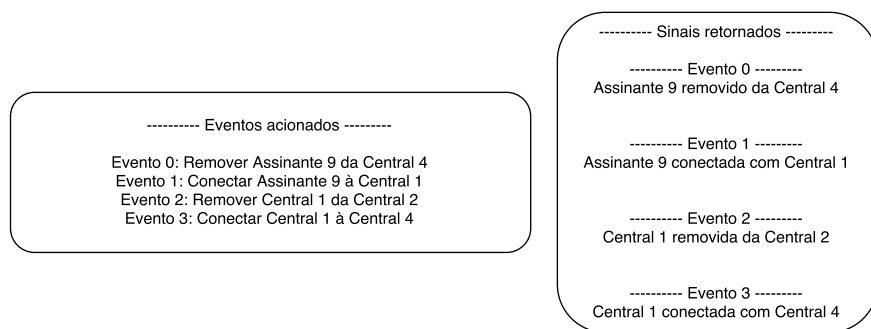
### 3.5. Inclusão e Exclusão de Linhas

Um sistema de telefonia deve gerenciar a inclusão e remoção de linhas, sendo uma das operações mais básicas e importantes na manutenção da rede e para correto funcionamento. Nas Figuras 8 e 9 abaixo pode ser observado um caso típico.



**Figura 8. Entrada para Demonstração de Inclusão e Exclusão de Linhas**





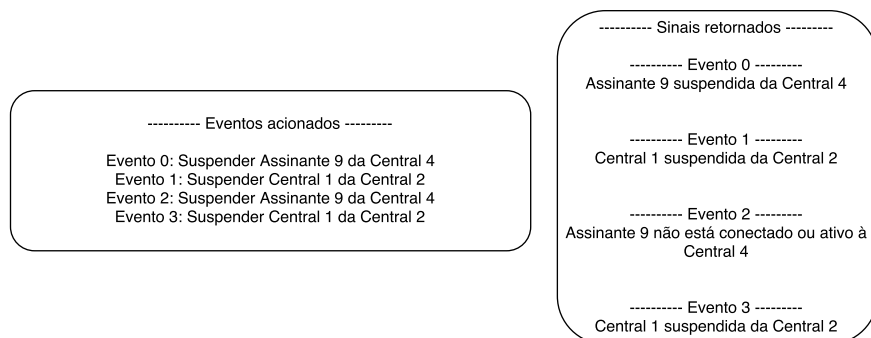
**Figura 9. Saída para Demonstração de Inclusão e Exclusão de Linhas**

### 3.6. Suspensão e Reativação de Linhas

Um sistema de telefonia deve gerenciar a inclusão e remoção de linhas, sendo uma das operações mais básicas e importantes na manutenção da rede e para correto funcionamento. Nas Figuras 10 e 11 abaixo pode ser observado um caso de teste para este cenário.



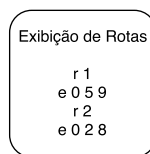
**Figura 10. Entrada para Demonstração de Inclusão e Exclusão de Linhas**



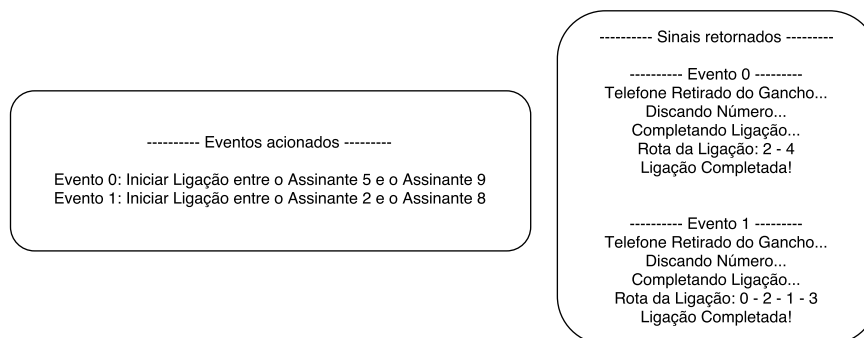
**Figura 11. Saída para Demonstração de Inclusão e Exclusão de Linhas**

### 3.7. Exibição de Rota

As Figuras 12 e 13, exibem o passo a passo das operações e a rota das ligações. Na segunda é possível atestar que o sistema consegue lidar com rotas que passam por mais de duas centrais. A exibição de rotas faz parte do sistema para facilitar a auditoria e uso das linhas, num caso real auxiliam a empresa a descobrir onde a infraestrutura da rede deve ser aprimorada.



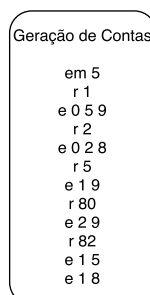
**Figura 12. Entrada para Demonstração de Exibição de Rota**



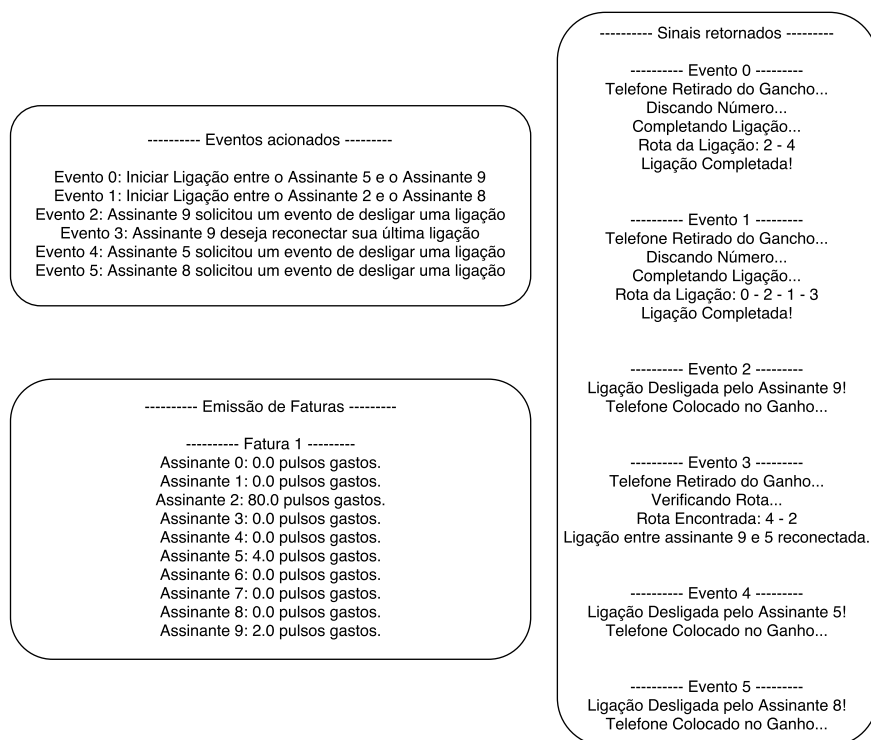
**Figura 13. Saída para Demonstração de Exibição de Rota**

### 3.8. Geração de Contas

A geração de faturas, é uma importante métrica para o usuário e a empresa que possui o sistema de gerenciamento, pois permite ao usuário controle sobre o que gasta, e no caso da empresa o ajuste de tarifas para manter o sistema em funcionamento. Para esse sistema foi considerado a passagem de tempo indicado na mudanças de round, e a realização da contagem de pulsos a cada unidade de tempo, por fim imprimindo essa conta. Um exemplo nas Figuras 14 e 15.



**Figura 14. Entrada para Demonstração de Geração de Contas**



**Figura 15. Saída para Demonstração de Geração de Contas**