Módulo de auto-proteção em um sistema *peer-to-peer*

Acadêmico: Philipp Albert Schroeder
Orientador: Paulo Fernando da Silva

Roteiro

- Introdução
  - Objetivos
- Fundamentação teórica
  - *Peer-to-peer*
  - Ataques remotos
  - Computação autonômica
- Desenvolvimento
  - Especificação
  - Implementação
- Conclusão

Introdução

- Informação
- Sistemas complexos
- Automação
- Novo paradigma: Computação autonômica
- P2P

Objetivos

- Desenvolver um módulo de auto-proteção
  - procurar portas abertas não autorizadas pelo sistema;
  - impedir acesso de terceiros a arquivos não autorizados do sistema;
  - impedir a execução de arquivos não confiáveis, cuja procedência seja dúbia.

Fundamentação teórica

- P2P
  - modelo centralizado
  - modelo descentralizado
- Ataques remotos
  - recusa de serviço
  - spoofing
  - sniffers
  - cavalos de tróia
  - backdoors

Fundamentação teórica

- Computação autonômica
  - Auto-conhecimento
  - Auto-configuração
  - Auto-otimização
  - Auto-cura
  - Auto-proteção
  - Conhecimento do ambiente
  - Heterogeneidade
  - Antecipação

Trabalhos correlatos

- AntHill
  - computação autonômica
  - P2P
  - agentes
  - auto-organização e adaptação

Desenvolvimento

- Requisitos:
  - Procurar portas abertas não-autorizadas
  - Impedir acesso de terceiros a arquivos não-autorizados
  - Impedir execução de arquivos não-confiáveis
  - Implementado em Java

Especificação

- UML
  - Diagrama de classes
  - Diagrama de casos de uso
  - Diagrama de seqüência

Diagrama de classes

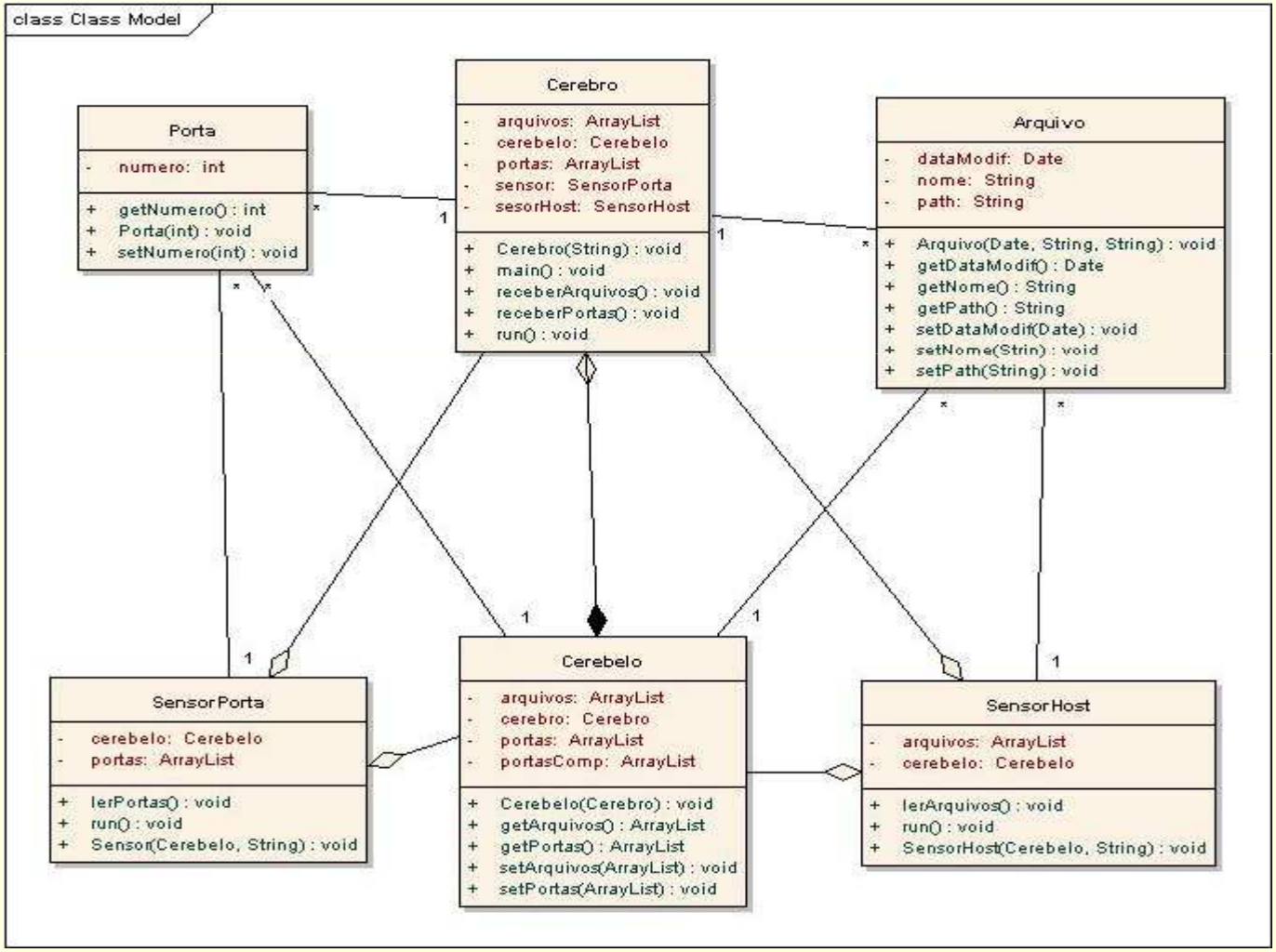
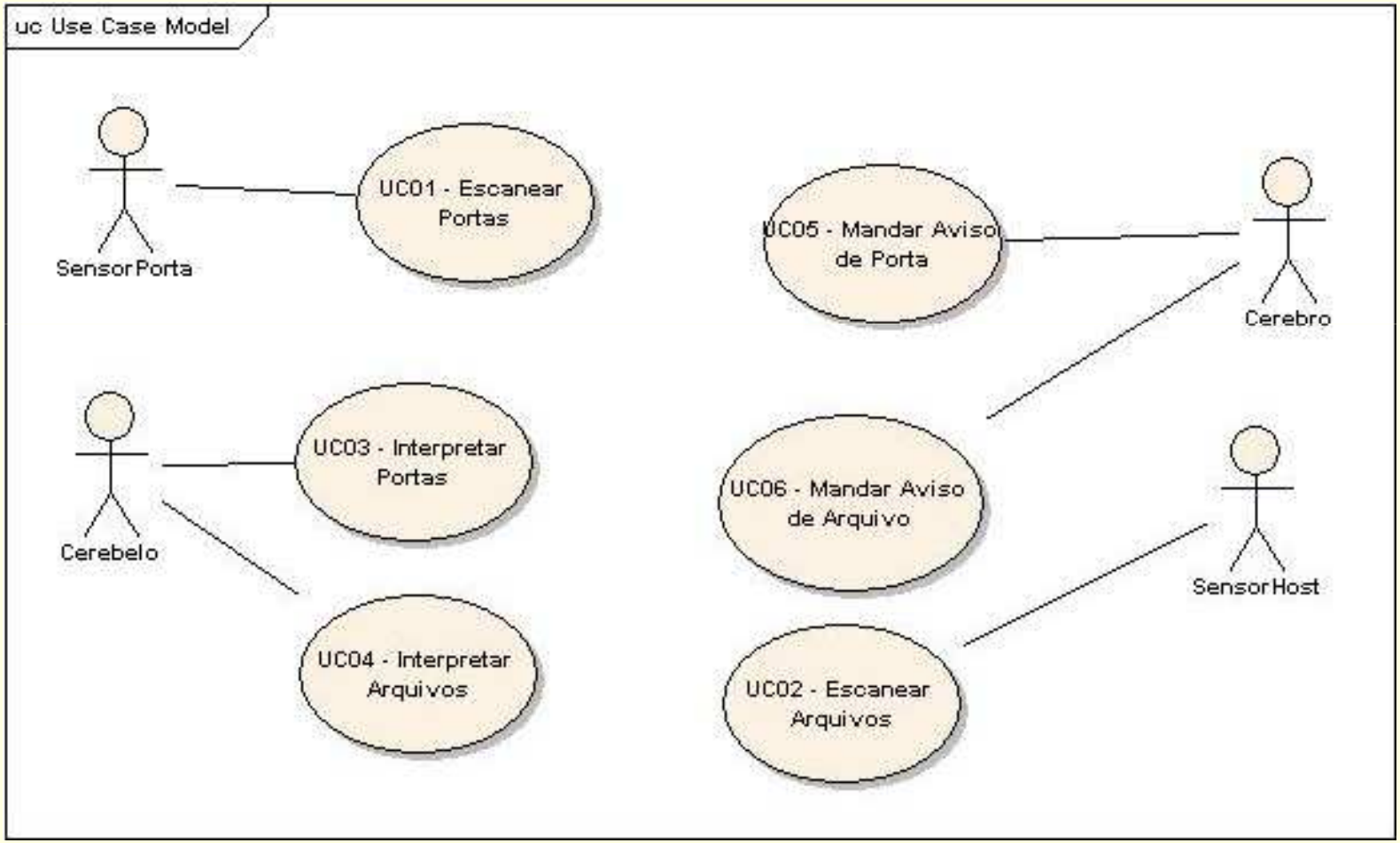
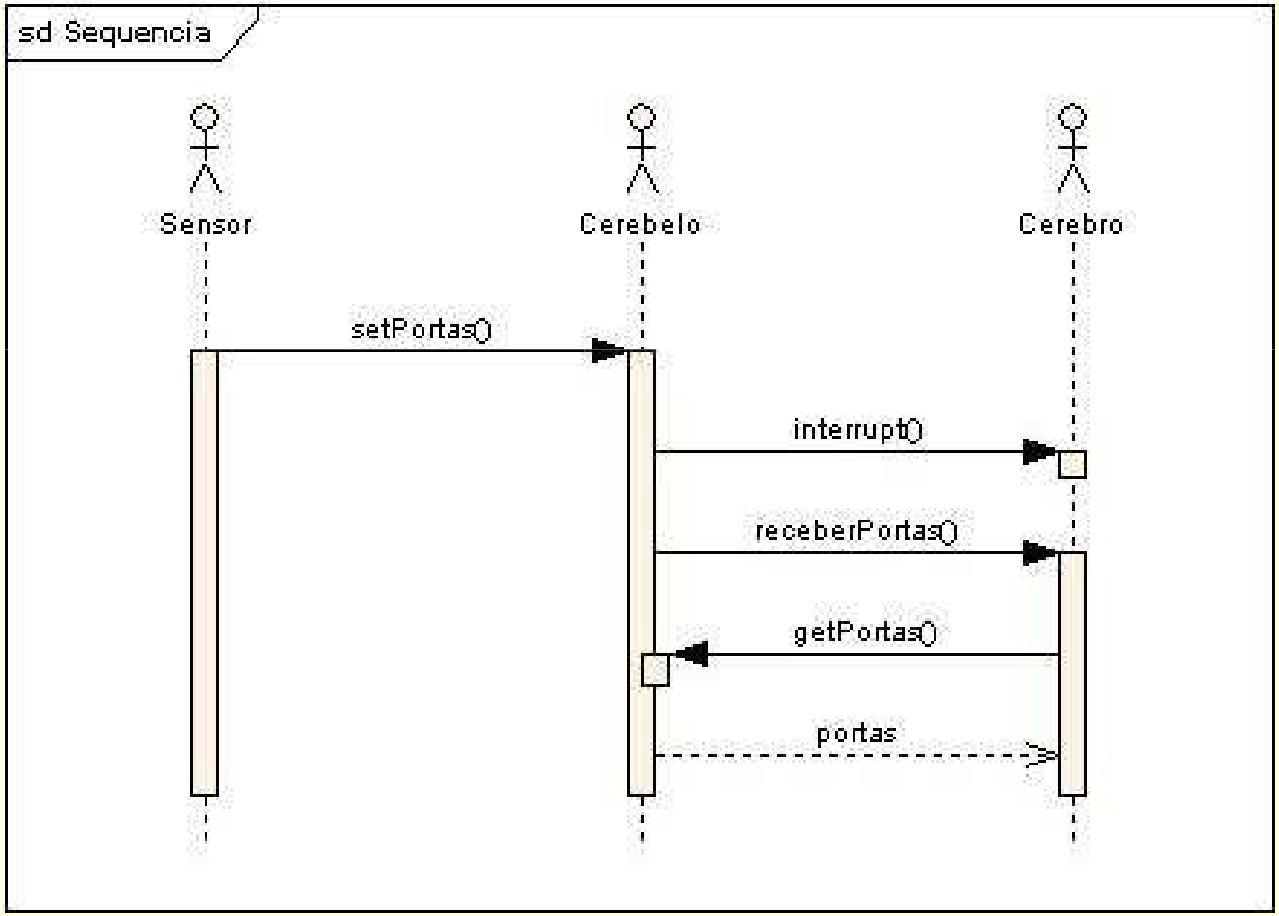


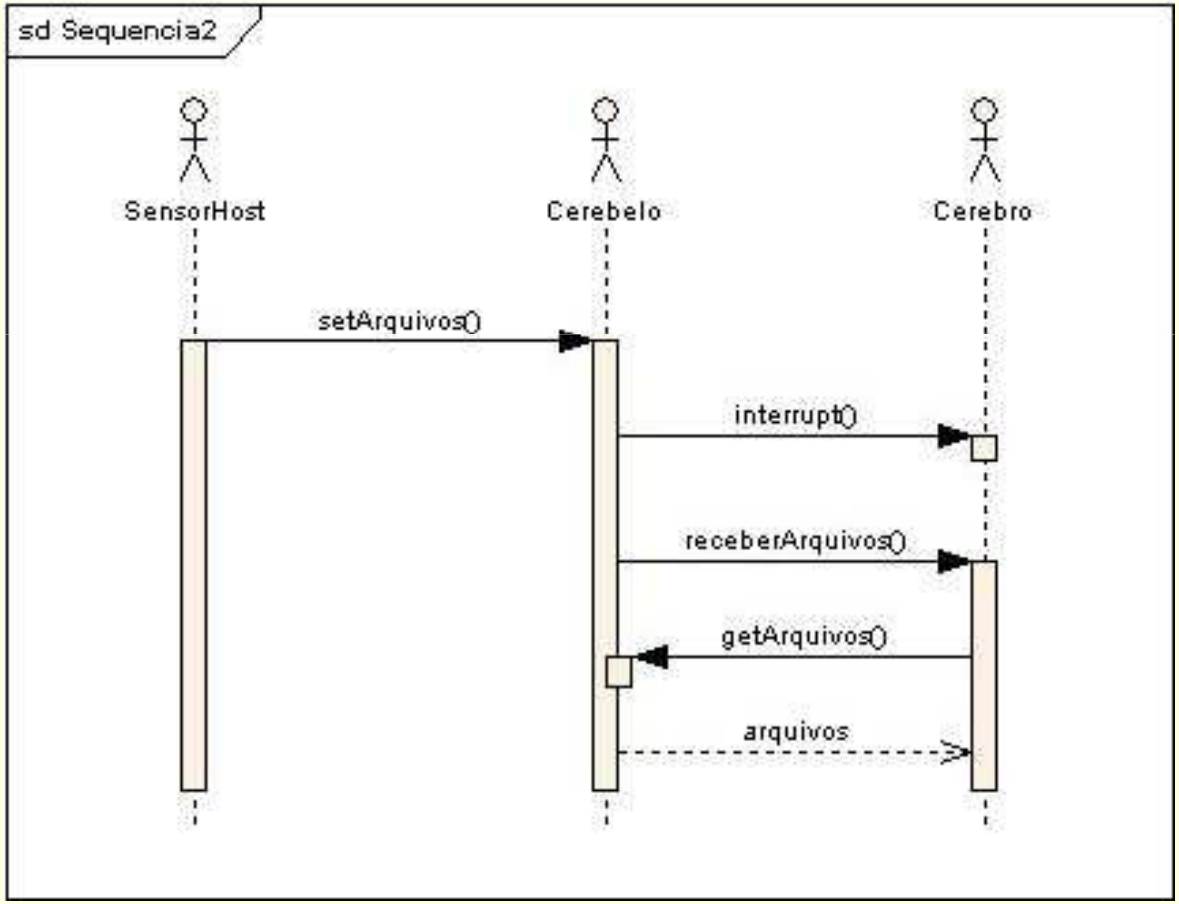
Diagrama de Casos de Uso



Diagramas de Seqüência



Diagramas de Seqüência



Implementação

- Java
- Eclipse 3.2
- Computação-autonômica
  - Auto-proteção

Implementação

- Sensor de porta:

```
for(int i = 0; i <= 6553; i++)
{
  try{
      ServerSocket port = new ServerSocket(i);
      port.close();
  }catch(java.io.IOException e){
      Porta p = new Porta(i);
      portas.add(p);
  }
}
```

Implementação

- Sensor de host:

```
for(int i = 0; i < listaArquivos.length; i++)
{
  Arquivo arq = new Arquivo(listaArquivos[i].getName(),
                    listaArquivos[i].getPath(),
                    new Date(listaArquivos[i].lastModified()));
  arquivos.add(arq);
}
```

Implementação

- Teste de portas:

```
public void
    setPortas(ArrayList<Porta>
    port) {
  boolean tem = false;
  while(port.size()>0)
{
    tem = false;
   Porta po = port.get(0);
   for(int i=0; i<portasComp.size();
     i++)
   {
   Porta q = portasComp.get(i);
   if
     (po.getNumero()==q.getNumero
     ()){
      tem = true;
   }
}
   if (!tem){
     portas.add(po);
   }
   port.remove(0);
}
//interrompe o cerebro
cerebro.interrupt();
//manda o cerebro receber os
    dados
cerebro.receberPortas();
}
```

Implementação

- Teste de Arquivos:

```
public void setArquivos(ArrayList<Arquivo> arq){
  while(arq.size()>0){
    Arquivo ar = arq.get(0);
    java.util.Date agora = new java.util.Date();
    if (ar.getDataModif().getTime()+tempo > agora.getTime()){
      arquivos.add(ar);
    }
    arq.remove(0);
  }
  //interrompe o cerebro
  cerebro.interrupt();
  //manda o cerebro receber os dados
  cerebro.receberArquivos();
}
```

Implementação

- Aviso de portas:

```
//se houver recebido as portas
if(portas.size()>0){
   //para cada porta recebida
   //imprime a porta aberta
   while(portas.size()>0){
      Porta po = portas.get(0);
      JOptionPane.showMessageDialog(null,"Atenção! \nA porta
     "+po.getNumero()        +" não deveria estar aberta!");
      portas.remove(0);
   }
}
```

Implementação

- Aviso de arquivos:

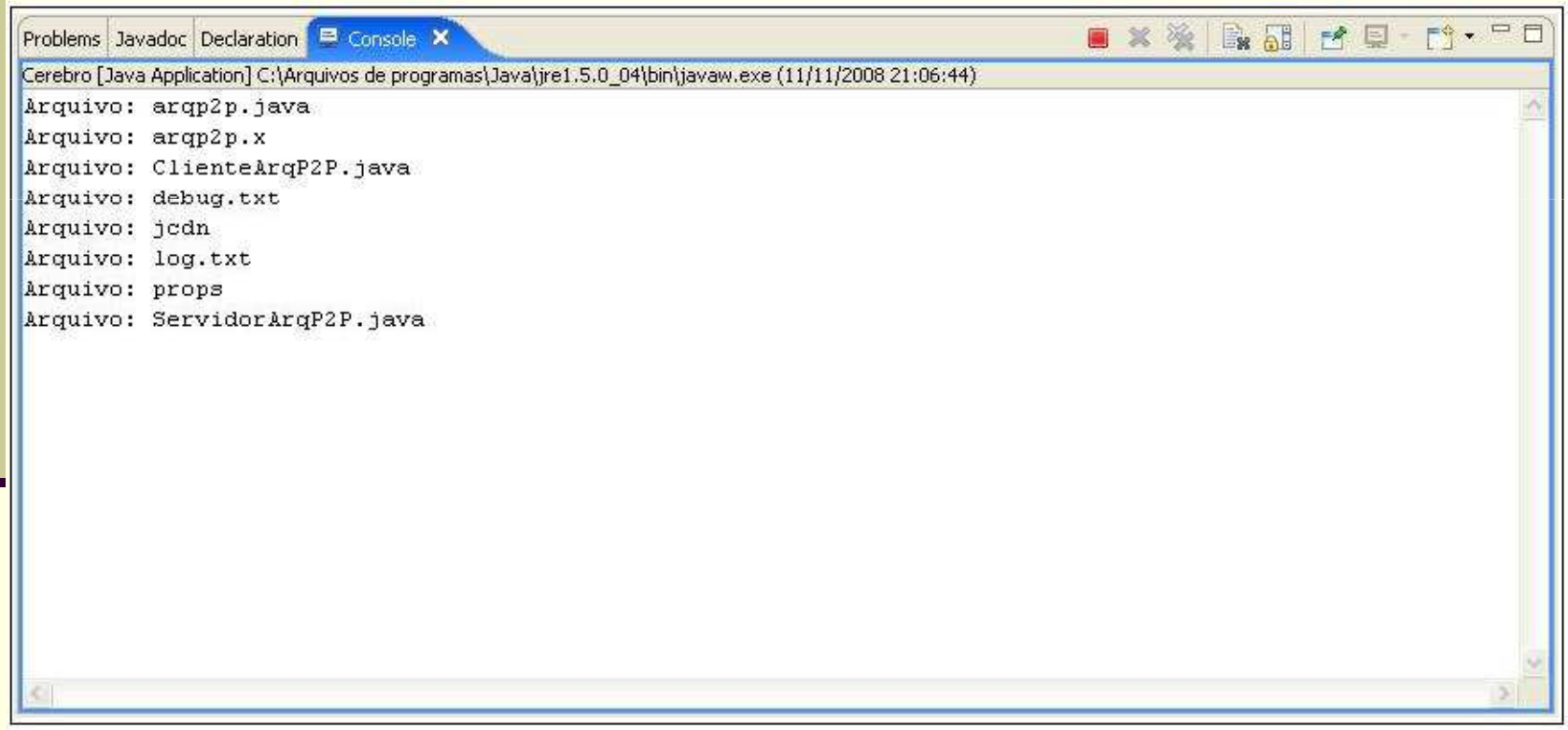
```
//se recebeu o array de arquivos
if(arquivos.size()>0){
   //para cada arquivo
   //imprime o arquivo modificado
   while(arquivos.size()>0){
      Arquivo ar = arquivos.get(0);
      JOptionPane.showMessageDialog(null,"Atenção! \nO arquivo
     "+ar.getNome()        +" foi modificado!");
      arquivos.remove(0);
   }
}
```

Operacionalidade

- Tela inicial:

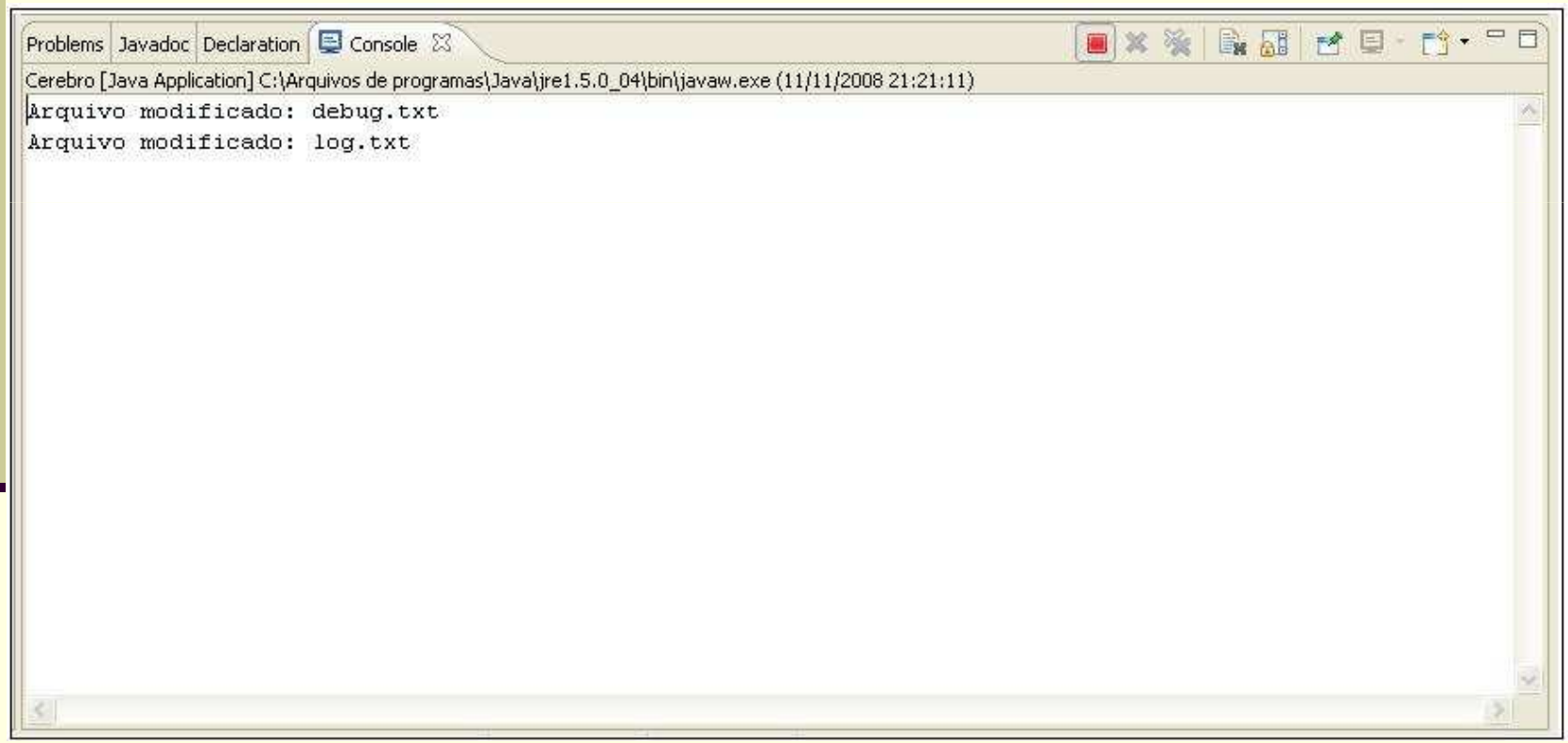
Operacionalidade

- Exemplo de lista de arquivos:



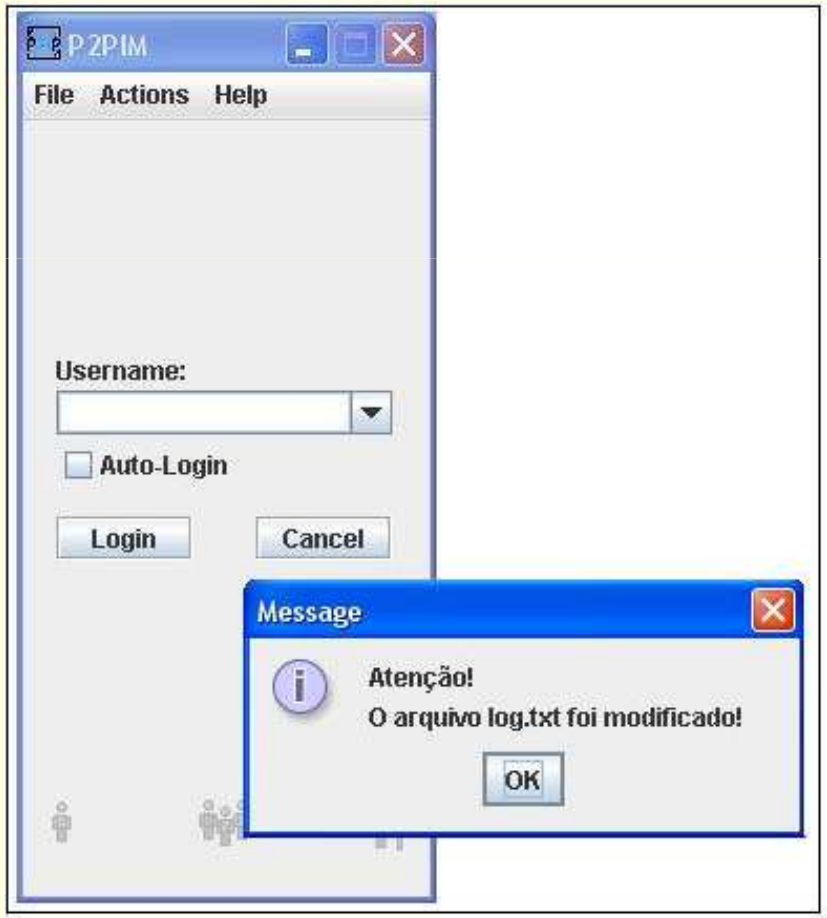
Operacionalidade

- Exemplo de lista de arquivos modificados:



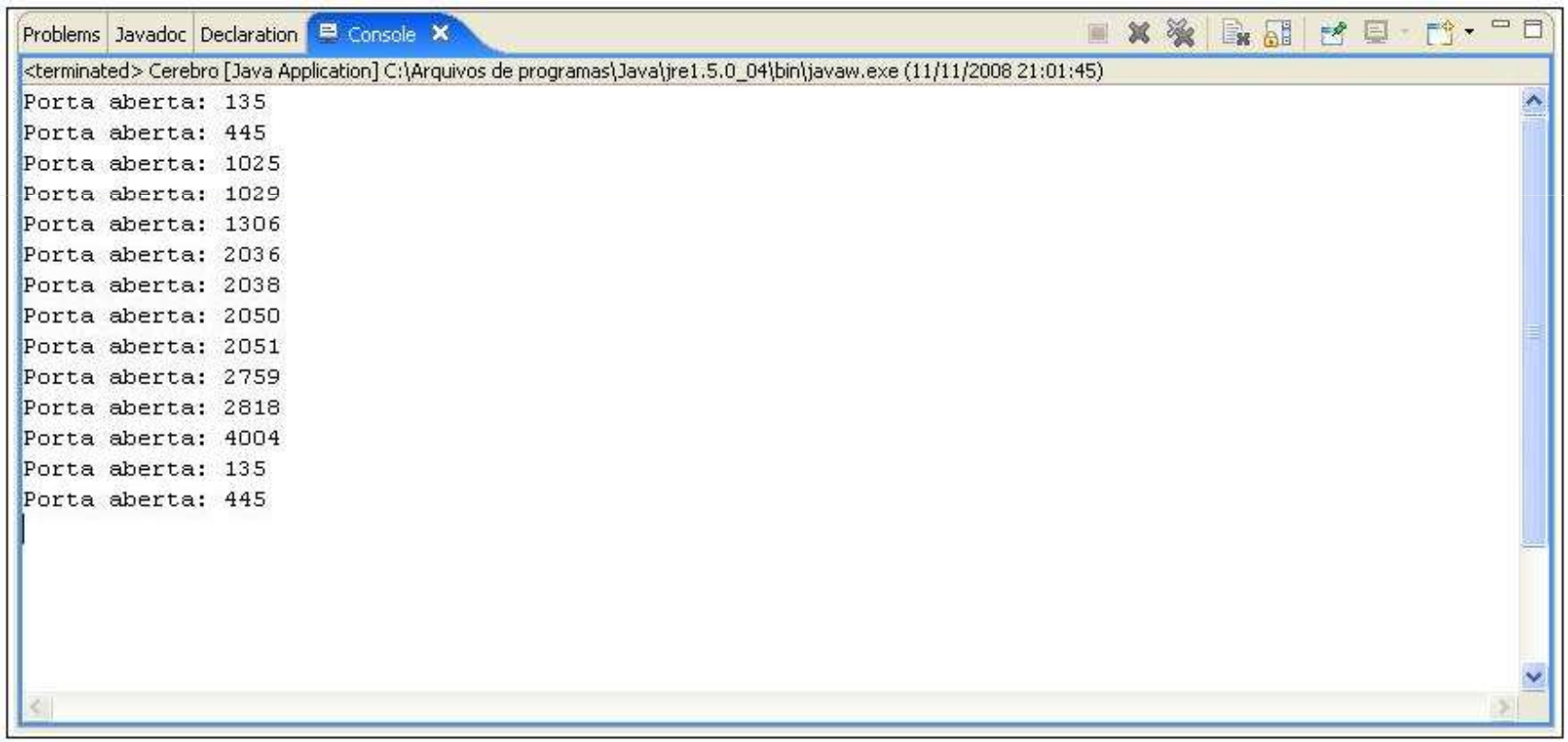
Operacionalidade

- Exemplo de aviso de arquivo modificado:



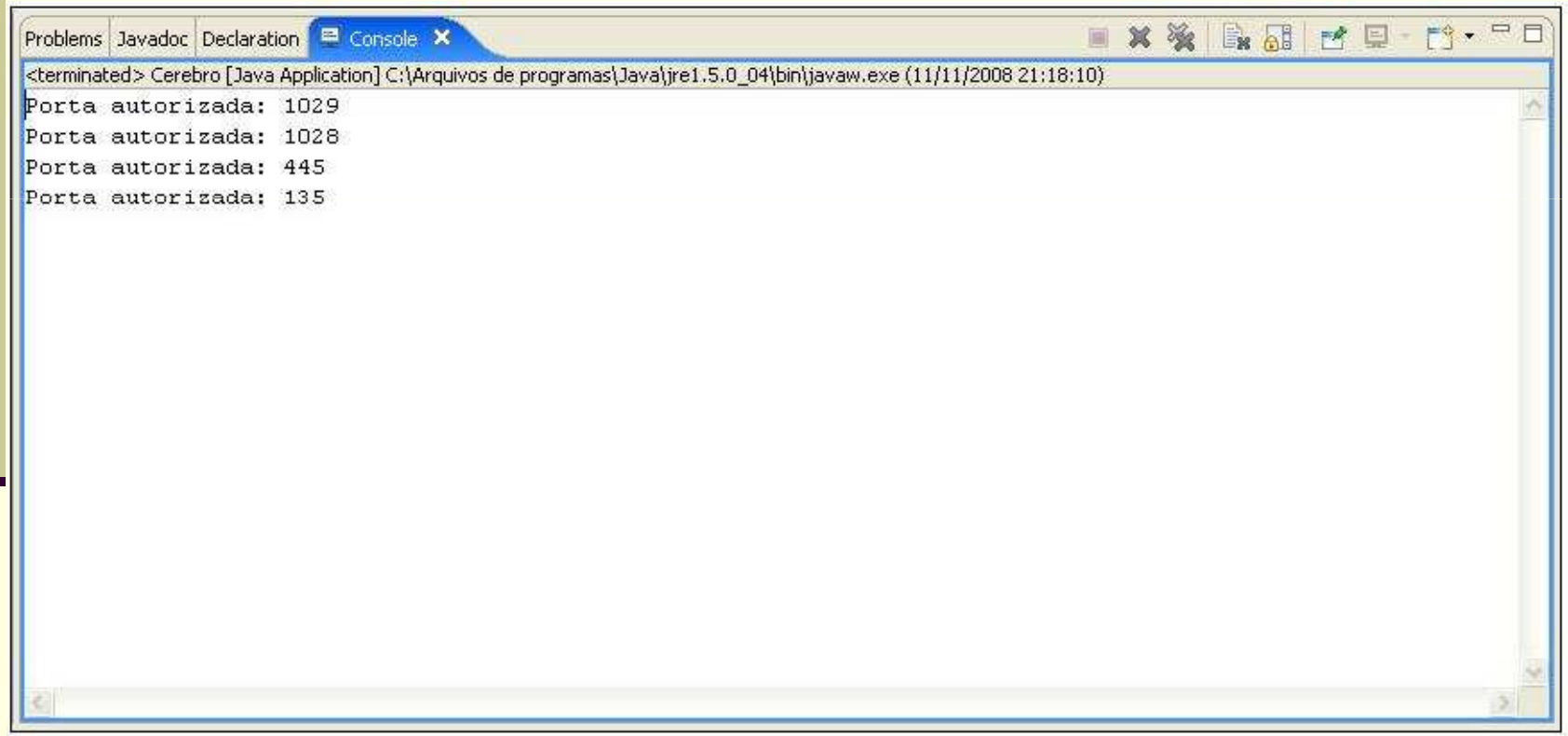
Operacionalidade

- Exemplo de lista de portas abertas:



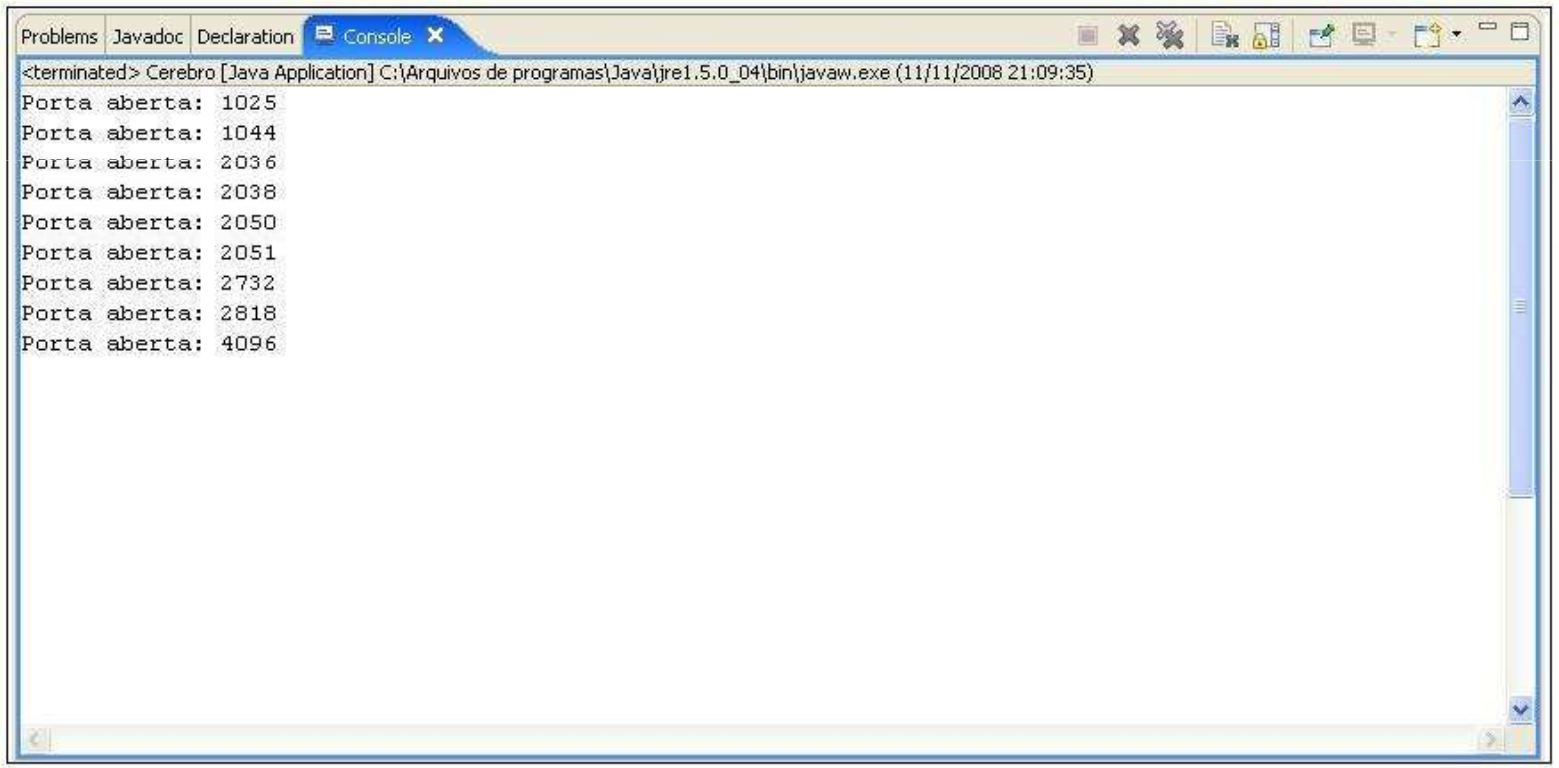
Operacionalidade

- Exemplo de lista de portas autorizadas:



Operacionalidade

- Exemplo de lista de portas abertas não-autorizadas:



Operacionalidade

- Exemplo de aviso de porta não-autorizada aberta:

Resultados

- Aviso sobre portas abertas
  - Backdoor
- Aviso sobre arquivos modificado
  - Invasão
- Módulo independente

Conclusões

- Conceitos de
  - peer-to-peer
  - ataques remotos
  - computação autonômica
- Técnicas e ferramentas adequadas
- Verificação correta
  - portas abertas
  - arquivos modificados

Extensões

- Agentes
- Uso de memória
- Procedência de arquivos