Data de Entrega: 26/09/2016
O que deve ser entregue: 1. Deve ser enviado pelo SIGAA um arquivo zipado contendo os códigos fontes dos programas implementados. Use o nome especificado em cada questão para nomear o seu programa;
Como será avaliado o trabalho: A avaliação deste trabalho levará em conta a correção e qualidade das questões respondidas. Cada questão descrita abaixo possui uma pontuação definida.

Laboratório 3 – Comunicação entre processos

1) Baixe o arquivo comprimido (CK069 SOCKET & RMI) disponível no SIGAA, o qual contém os exemplos que você pode utilizar neste laboratório;
2) Compile e execute os códigos apresentados nas Figuras 1 e 2 (disponíveis no diretório ”socket” do arquivo comprimido, classes DataClient.java e DataServer.java) referentes aos códigos de um cliente e servidor que se comunicam via sockets;
3) Modifique o servidor/cliente mostrado nas Figuras 1 e 2 para criar um servidor que receba uma expressão matemática no formato “operador:operando1:operando2”. O servidor devera’ permitir a execução das operações de adição, subtração, multiplicação e divisão.
4) Utilize os códigos desenvolvidos anteriormente para implementar uma comunicação entre processos através de mecanismos de comunicação indireta, através do mecanismo de mailbox. Desta maneira, o servidor deve funcionar como uma mailbox e a sua função será armazenar ou recuperar mensagens pelos programas clientes. Neste exercício, um cliente poderá enviar mensagem para um determinado cliente (cada cliente deve possuir um identificador) registrado no servidor. Um cliente poderá também requisitar as mensagens enviadas para ele;
5) Implemente a solução desenvolvida no item 4 utilizando agora o método RMI. Utilize os arquivos exemplo (diretório “rmi” e subdiretórios “cliente” e “server”) contidos no arquivo comprimido, baixado no item 1, como base para sua implementação.

Date Server (Fig 1)

```
public class DateServer
{
   public static void main(String[] args) {
      try {
         ServerSocket sock = new ServerSocket(6013);

         // now listen for connections
         while (true) {
            Socket client = sock.accept();

            PrintWriter pout = new
             PrintWriter(client.getOutputStream(), true);

            // write the Date to the socket
            pout.println(new java.util.Date().toString());

            // close the socket and resume
            // listening for connections
            client.close();
         }
      }
      catch (IOException ioe) {
         System.err.println(ioe);
      }
   }
}
```

Client Server (Fig 2)

```
public class DateClient
{
   public static void main(String[] args) {
      try {
         //make connection to server socket
         Socket sock = new Socket("127.0.0.1",6013);

         InputStream in = sock.getInputStream();
         BufferedReader bin = new
            BufferedReader(new InputStreamReader(in));

         // read the date from the socket
         String line;
         while ( (line = bin.readLine()) != null)
            System.out.println(line);

         // close the socket connection
         sock.close();
      }
      catch (IOException ioe) {
         System.err.println(ioe);
      }
   }
}
```