CS550 Advanced Operating Systems
Programming Assignment 2
Source Code

Submitted by:
Chiranjeevi Ankamredy
A20359837

a. PeerClient.java

```
package PA2;
import java.io.*;
import java.net.*;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.util.Scanner;

class Hashtableop
{
    private int  maxSize;
    private String[] keys;
    public Hashtableop(int capacity)
    {

        maxSize = capacity;
        keys = new String[maxSize];

    }

    private int hash(String key) //return the hashcode to select the server
    {
        return key.hashCode() % maxSize;
    }

    public int find(String key)
    {
        int tmp = hash(key);
               return tmp;
    }

}

class PeerClient {
    public static void main(String args[])
        {

            try
              {

                 Scanner scan = new Scanner(System.in);
                 Hashtableop Serverslct = new Hashtableop(8);
                 //hashtable to connect server on kry selection

                 System.out.println("Enter key to connect server:");
                 int n=Serverslct.find(scan.next());
                 if (n < 0)
                 n = -n;
                System.out.println("connecting to the server "+n);
                int k=0;
                  BufferedReader br = new BufferedReader(new
                InputStreamReader(new FileInputStream("Config.txt")));
                String Peerdetls;
                 String line;
                 while ((line = br.readLine()) != null)
                {


                 if(n==k)
                 {
                   Peerdetls=line;
                   String words[] = Peerdetls.split(" ");
                  //  String firstTwo = words[0] + "  " + words[1];
                   System.out.println(words[0]);
                   System.out.println(words[1]);

                   int port = Integer.parseInt(words[1]);
                    Socket s=new Socket(words[0],port);
//connecting to the server
                    System.out.println("Peer1 Intitialized");
                       DataInputStream  inp = new
                      DataInputStream(s.getInputStream());
                       DataOutputStream oup = new
                      DataOutputStream(s.getOutputStream());

                   char ch;

               do
                {
                  System.out.println("\nHash Table Operations\n");
                        System.out.println("1. PUT ");
                        System.out.println("2. GET");
                        System.out.println("3. DELETE");
                //selecting the choice
                 int choice = scan.nextInt();
                        String choice1 = Integer.toString(choice);
                        oup.writeBytes(choice1);
                        oup.writeByte('\n');

                       switch (choice)
                       {
                        case 1 :
                //Doing put operation by selecting choice 1
                         System.out.println("Enter key and value");
                        oup.writeBytes(scan.next());
                                 oup.writeByte('\n');
                                 oup.writeBytes( scan.next());
                                 oup.writeByte('\n');
                                  String ip21 = inp.readLine();
                                   System.out.println(ip21);
                                  break;
                         case 2 :
                      //Doing get operation by selecting choice 2
                                   System.out.println("Enter key");
                                 oup.writeBytes(scan.next());
                                 oup.writeByte('\n');
                                   String ip6 = inp.readLine();
                                   System.out.println("Value = "+ip6 );
                                   break;
                         case 3 :

                       //Doing Delete operation by selecting choice 1
                                   System.out.println("Enter key");
                                    oup.writeBytes(scan.next());
                                    oup.writeByte('\n');
                                    String ip22 = inp.readLine();
                                   System.out.println(ip22);
                          break;

                  default :
                         System.out.println("Wrong Entry  ");
                         break;
                          }


           System.out.println("Do you want to continue (Type y or n) \n");
           ch = scan.next().charAt(0);
            String str = Character.toString(ch);
            oup.writeBytes(str);
            oup.writeByte('\n');
       } while (ch == 'Y'|| ch == 'y');





                 }
                 k++;
                }



               br.close();




               }

           catch (Exception e)
            {
           System.err.println("Error: " + e.getMessage());
            }
   }
```

```
}
```

b. Peerserver.java

```
package PA2;
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.Map.Entry;
import java.util.Scanner;


class Hash
{

    public static int currentSize, maxSize;
    public static  String keys;
    public static String vals;
    static String resilence;

      public static Hashtable<String,String> data=  new
      Hashtable<String,String>(1000001);
      //Hashtable contains 1000001 keys.
    public  Hash()
    {  currentSize = 0;
       keys = new String();
       vals = new String();
    }

    //Inserting keys and values in hash table

   void insert(String key, String val)
    {
        keys=key;
        vals=val;
        data.put(keys,vals);
        return;

    }

   //geeting the valued based on key
   public  String get(String Name){

          keys=Name;
         return data.get(keys);
      }

    //Deleting the key value pair
   void delete(String key)
```

```
    {
        keys=key;
        data.remove(keys);


    }
   //printing the hash table values
    void printHashTable()
    {     System.out.println("Hash Table " );
          Iterator<Entry<String, String>> it = data.entrySet().iterator();
           while (it.hasNext())
                  {

                      Entry<String, String> pair = it.next();
      System.out.println(pair.getKey() + " " +
     pair.getValue());

                  }

    }


}
//creating thread for each peer
class ThreadHandler extends Thread
{


     private static final String port1 = null;
      Socket News;
     int n;

     ThreadHandler(Socket s,int v)
     {
       News=s;
       n=v;
     }

     public void run()
     {
      try
        {
       System.out.println("Thread created for peer" );
       Scanner scan = new Scanner(System.in);
       DataInputStream  inp = new DataInputStream(News.getInputStream());
       DataOutputStream oup = new DataOutputStream(News.getOutputStream());
            Hash h1 = new Hash();





             char ch;
             do
             {


            String ip = inp.readLine();
             int choice = Integer.parseInt(ip);
            switch (choice)
            {
            case 1 : //put operation
                String ip1 = inp.readLine();
                String ip2 = inp.readLine();
                h1.insert(ip1, ip2 );
                String ip15="Success";
                 oup.writeBytes(ip15);
                 oup.writeByte('\n');
                System.out.println("Key and values are inserted");

                break;

            case 2 : //get operation
                String ip3 = inp.readLine();
                 String ip11=(String) h1.get(ip3);
                 if(ip11==null)
                  { String i26="Invalid Key";
                   oup.writeBytes(i26);
                  oup.writeByte('\n');}
                 else
                  {oup.writeBytes(ip11);
                  oup.writeByte('\n');}
                break;
            case 3 :  //delete operation
                String ip4 = inp.readLine();
                h1.delete(ip4);
                 String ip18="Deleted";
                 oup.writeBytes(ip18);
                  oup.writeByte('\n');
                break;

            default :
                System.out.println("Wrong Entry  ");
                break;
            }

           h1.printHashTable();


            String ctr = inp.readLine();
            ch = ctr.charAt(0);
        } while (ch == 'Y'|| ch == 'y');



       // News.close();
        }
```

```
        catch(Exception e)
        {System.out.println(e);}

  }
}



public class PeerServer
 {
     public static void main(String[] args)
      {
           int req=1001;
         try
          {      int port=7777;

               ServerSocket ss=new ServerSocket(port);
              //server port is intilized as 7777

               for(;;)
               {
                 Socket s=ss.accept();   //establishes connection
                 System.out.println("Server started ");
                 Thread T =new ThreadHandler(s,req);
                 T.start();

                 req++;
               }

          }
          catch(Exception e)
          {System.out.println(e);}
        }

}
```

C.Config.txt

```
127.0.0.1 2222
127.0.0.1 3333
127.0.0.1 4444
127.0.0.1 5555
127.0.0.1 6666
127.0.0.1 7777
127.0.0.1 8888
127.0.0.1 9999
```

Resilience

a.peerclient.java

```
package resilience;
import java.io.*;
import java.net.*;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.util.Scanner;

class Hashtableop
{
    private int  maxSize;
    private String[] keys;
    public Hashtableop(int capacity)
    {

        maxSize = capacity;
        keys = new String[maxSize];

    }

    private int hash(String key)
    {
        return key.hashCode() % maxSize;
    }

    public int find(String key)
    {
        int tmp = hash(key);
               return tmp;
    }

}

class PeerClient {
    public static void main(String args[])
        {

            try
              {

                 Scanner scan = new Scanner(System.in);
                 Hashtableop Serverslct = new Hashtableop(8);
                 System.out.println("Enter key to connect server:");
                 int n=Serverslct.find(scan.next());
                 if (n < 0)
                  n = -n;
                 System.out.println("connecting to the server "+n);
                 int k=0;
```

```
                  BufferedReader br = new BufferedReader(new
InputStreamReader(new FileInputStream("Config.txt")));
                  String Peerdetls;
                   String line;
                   while ((line = br.readLine()) != null)
                  {


                   if(n==k)
                   {
                     Peerdetls=line;
                     String words[] = Peerdetls.split(" ");
                    //  String firstTwo = words[0] + "  " + words[1];
                     System.out.println(words[0]);
                     System.out.println(words[1]);

                     int port = Integer.parseInt(words[1]);
                      Socket s=new Socket(words[0],port);
                      System.out.println("Peer1 Intitialized");
                      DataInputStream  inp = new
DataInputStream(s.getInputStream());
                      DataOutputStream oup = new
DataOutputStream(s.getOutputStream());

                     char ch;

                 do
                  {
                    System.out.println("\nHash Table Operations\n");
                          System.out.println("1. PUT ");
                          System.out.println("2. GET");
                          System.out.println("3. DELETE");

                   int choice = scan.nextInt();
                          String choice1 = Integer.toString(choice);
                          oup.writeBytes(choice1);
                          oup.writeByte('\n');

                         switch (choice)
                         {
                          case 1 :
                           System.out.println("Enter key and value");
                          oup.writeBytes(scan.next());
                                   oup.writeByte('\n');
                                   oup.writeBytes( scan.next());
                                   oup.writeByte('\n');
                                    String ip21 = inp.readLine();
                                     System.out.println(ip21);
                                    break;

                           case 2 :
                                     System.out.println("Enter key");
                                   oup.writeBytes(scan.next());
                                   oup.writeByte('\n');
```

```
                                    String ip6 = inp.readLine();
                                    System.out.println("Value = "+ip6 );
                                    break;
                        case 3 :
                                    System.out.println("Enter key");
                                     oup.writeBytes(scan.next());
                                     oup.writeByte('\n');
                                     String ip22 = inp.readLine();
                                    System.out.println(ip22);
                         break;

                   default :
                         System.out.println("Wrong Entry  ");
                         break;
                          }


             System.out.println("Do you want to continue (Type y or n) \n");
             ch = scan.next().charAt(0);
              String str = Character.toString(ch);
              oup.writeBytes(str);
              oup.writeByte('\n');
         } while (ch == 'Y'|| ch == 'y');





                  }
                  k++;
                 }



                br.close();




                }

             catch (Exception e)
              {
             System.err.println("Error: " + e.getMessage());
              }
     }
}
```

b.peerserver.java

```
   package Resilience;
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Hashtable;
import java.util.Iterator;
import java.util.Map.Entry;
import java.util.Scanner;
import Resilience.Resilience1;

class Hash
{

    public static int currentSize, maxSize;
    public static  String keys;
    public static String vals;
    static String resilence;

    public static Hashtable<String,String> data=  new
Hashtable<String,String>(1000001);
    public  Hash()
    {  currentSize = 0;
       keys = new String();
       vals = new String();
    }



   void insert(String key, String val)
    {
        keys=key;
        vals=val;
        data.put(keys,vals);
      return;

    }


   public  String get(String Name){

            keys=Name;
         return data.get(keys);
      }


   void delete(String key)
    {
         keys=key;
        data.remove(keys);


    }

    void printHashTable()
    {     System.out.println("Hash Table " );
         Iterator<Entry<String, String>> it = data.entrySet().iterator();
          while (it.hasNext())
```

```
                {

                      Entry<String, String> pair = it.next();
                      System.out.println(pair.getKey() + " " +
pair.getValue());

                }

    }


}

class ThreadHandler extends Thread
{


     private static final String port1 = null;
      Socket News;
     int n;

     ThreadHandler(Socket s,int v)
     {
       News=s;
       n=v;
     }

     public void run()
     {
      try
        {
           System.out.println("Thread created for peer" );
         Scanner scan = new Scanner(System.in);
           DataInputStream  inp = new
DataInputStream(News.getInputStream());
           DataOutputStream oup = new
DataOutputStream(News.getOutputStream());
            Hash h1 = new Hash();
            String q = "127.0.0.1";
            int k = n+ 1111;

            Resilience1 r=new Resilience1(q ,k);



              char ch;
              do
             {


            String ip = inp.readLine();
             int choice = Integer.parseInt(ip);
            switch (choice)
            {
```

```
            case 1 :
                String ip1 = inp.readLine();
                String ip2 = inp.readLine();
                h1.insert(ip1, ip2 );
                String ip15="Success";
                 oup.writeBytes(ip15);
                 oup.writeByte('\n');
                System.out.println("Key and values are inserted");

                break;

            case 2 :
                String ip3 = inp.readLine();
                 String ip11=(String) h1.get(ip3);
                 if(ip11==null)
                  { String i26="Invalid Key";
                   oup.writeBytes(i26);
                  oup.writeByte('\n');}
                 else
                  {oup.writeBytes(ip11);
                  oup.writeByte('\n');}
                break;
            case 3 :
                String ip4 = inp.readLine();
                h1.delete(ip4);
                 String ip18="Deleted";
                 oup.writeBytes(ip18);
                  oup.writeByte('\n');
                break;

            default :
                System.out.println("Wrong Entry  ");
                break;
            }

           h1.printHashTable();


            String ctr = inp.readLine();
            ch = ctr.charAt(0);
        } while (ch == 'Y'|| ch == 'y');



       // News.close();
        }

        catch(Exception e)
        {System.out.println(e);}

  }
}
```

```
public class PeerServer
 {
     public static void main(String[] args)
      {
           int req=1001;
         try
          {      int port=7777;

               ServerSocket ss=new ServerSocket(port);

               for(;;)
               {
                 Socket s=ss.accept();   //establishes connection
                 System.out.println("Server started ");
                 Thread T =new ThreadHandler(s,req);
                 T.start();

                 req++;
               }

          }
          catch(Exception e)
          {System.out.println(e);}
        }

}
```

c.Resilience.java

```
package Resilience;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.net.UnknownHostException;

public class Resilience1
 {
      private static String host;
      private static int port;

      public Resilience1(String message,int p)
      {
            this.host=message;
            this.port=p;
      }

     public static void main(String[] args)
      {


         try {
                 Socket s=new Socket(host,port);



         } catch (UnknownHostException e) {

                 e.printStackTrace();
           } catch (IOException e) {

                 e.printStackTrace();
           }

      }

}
```

Evaluation

a.Peerserver.java

```
import java.io.*;
import java.net.*;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.InputStreamReader;
import java.util.Scanner;
import java.util.Hashtable;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;
import java.util.Map.Entry;
import java.util.Set;


class Hash
{

    public static int currentSize, maxSize;
    public static  String keys;
    public static String vals;

    public static Hashtable<String,String> data=  new
Hashtable<String,String>(1000001);
    public  Hash()
    {  currentSize = 0;
       keys = new String();
       vals = new String();
    }



   void insert(String key, String val)
    {
        keys=key;
        vals=val;
        data.put(keys,vals);
      return;

    }


   public  String get(String Name){

            keys=Name;
         return data.get(keys);
      }


   void delete(String key)
    {
         keys=key;
        data.remove(keys);


    }

    void printHashTable()
    {     System.out.println("Hash Table " );
         Iterator<Entry<String, String>> it = data.entrySet().iterator();
          while (it.hasNext())
                 {

                       Entry<String, String> pair = it.next();
                       System.out.println(pair.getKey() + " " +
pair.getValue());

                 }

    }


}

class ThreadHandler extends Thread
{


     Socket News;
     int n;

     ThreadHandler(Socket s,int v)
```

```
      {
        News=s;
        n=v;
      }

      public void run()
      {
       try
         {
            System.out.println("Thread created for peer" );
          Scanner scan = new Scanner(System.in);
            DataInputStream  inp = new
DataInputStream(News.getInputStream());
            DataOutputStream oup = new
DataOutputStream(News.getOutputStream());
             Hash h1 = new Hash();

               char ch;
               do
              {


             String ip = inp.readLine();
              int choice = Integer.parseInt(ip);
             switch (choice)
             {
             case 1 :
                 for(int k=100000;k<200000;k++)
                {
                 String ip1 = inp.readLine();
                 String ip2 = inp.readLine();
                 h1.insert(ip1, ip2 );
                }
                 String ip15="Success";
                  oup.writeBytes(ip15);
                  oup.writeByte('\n');
                 System.out.println("Key and values are inserted");

                 break;

             case 2 :
                  for(int k=100000;k<200000;k++)
                {
                  String ip3 = inp.readLine();
                  String ip11=(String) h1.get(ip3);
                  if(ip11==null)
                   { String i26="Invalid Key";
                    oup.writeBytes(i26);
                   oup.writeByte('\n');}
                  else
                   {oup.writeBytes(ip11);
                   oup.writeByte('\n');}
                  }
                 break;
             case 3 :
                  for(int k=100000;k<200000;k++)
                {
                 String ip4 = inp.readLine();
                 h1.delete(ip4);
                 }
                  String ip18="Deleted";
                  oup.writeBytes(ip18);
                   oup.writeByte('\n');
                 break;

             default :
                 System.out.println("Wrong Entry  ");
                 break;
             }

            h1.printHashTable();


             String ctr = inp.readLine();
             ch = ctr.charAt(0);
         } while (ch == 'Y'|| ch == 'y');



        // News.close();
         }

         catch(Exception e)
         {System.out.println(e);}

   }
 }



 public class PeerSerevr
  {
      public static void main(String[] args)
       {
            int req=1001;
          try
           {  int port=7777;
           ServerSocket ss=new ServerSocket(port);

                for(;;)
                {
                  Socket s=ss.accept();   //establishes connection
                  System.out.println("Server started ");
                  Thread T =new ThreadHandler(s,req);
                  T.start();

                  req++;
                }
           }
           catch(Exception e)
           {System.out.println(e);}
         }

 }
```

c. EvaluationClient.java

```
package Evaluation;
import java.io.*;
import java.net.*;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.io.InputStreamReader;
import java.util.Scanner;

class Hashtableop
{
    private int  maxSize;
    private String[] keys;
    public Hashtableop(int capacity)
    {

        maxSize = capacity;
        keys = new String[maxSize];

    }

    private int hash(String key)
    {
        return key.hashCode() % maxSize;
    }

    public int find(String key)
    {
        int tmp = hash(key);
               return tmp;
    }

}

class Evaluationclient1 {
    public static void main(String args[])
        {
            try
              {

                 Scanner scan = new Scanner(System.in);
                 Hashtableop Serverslct = new Hashtableop(8);
                 System.out.println("Enter key to connect server:");
                 int n=Serverslct.find(scan.next());
                 if (n < 0)
                  n = -n;
                 System.out.println("connecting to the server "+n);
                 int k=0;

                 BufferedReader br = new BufferedReader(new
InputStreamReader(new FileInputStream("Config.txt")));
                 String Peerdetls;
                  String line;
                  while ((line = br.readLine()) != null)
                 {


                  if(n==k)
                  {
                    Peerdetls=line;
                    String words[] = Peerdetls.split(" ");
                   //  String firstTwo = words[0] + "  " + words[1];
                    System.out.println(words[0]);
                    System.out.println(words[1]);

                    int port = Integer.parseInt(words[1]);
                     Socket s=new Socket(words[0],port);
                     System.out.println("Peer1 Intitialized");
                     DataInputStream  inp = new
DataInputStream(s.getInputStream());
                     DataOutputStream oup = new
DataOutputStream(s.getOutputStream());

                    char ch;

                do
                 {
                   System.out.println("\nHash Table Operations\n");
                         System.out.println("1. PUT ");
                         System.out.println("2. GET");
                         System.out.println("3. DELETE");

                  int choice = scan.nextInt();
                         String choice1 = Integer.toString(choice);
                         oup.writeBytes(choice1);
                         oup.writeByte('\n');
                        int i;
                        switch (choice)
                        {
                         case 1 :
```

```
                          System.out.println("Enter key and value");
                                   long millis = System.currentTimeMillis()
% 1000;
                                 for( i=100000;i<200000;i++)
                         { String j=Integer.toString(i);
                                   oup.writeBytes(j);
                                  oup.writeByte('\n');
                                  oup.writeBytes(j);
                                  oup.writeByte('\n');
                                   }
                                long millis1 = System.currentTimeMillis() %
1000;
                                 long pute=millis1-millis;
                                  System.out.println("Put time"+pute);
                               String ip21 = inp.readLine();
                                    System.out.println(ip21);
                                   break;

                          case 2 :
                                    System.out.println("Enter key");
                                    long millis2 =
System.currentTimeMillis() % 1000;
                                 for( i=100000;i<200000;i++)
                         { String j=Integer.toString(i);
                                  oup.writeBytes(j);
                                  oup.writeByte('\n');
                                    String ip6 = inp.readLine();
                                    System.out.println("Value = "+ip6 );
                                   }
                                    long millis3 =
System.currentTimeMillis() % 1000;
                                 long gett=millis3-millis2;
                                  System.out.println("gett time:"+gett);
                                    break;
                          case 3 :
                                    System.out.println("Enter key");
                                     long millis4 =
System.currentTimeMillis() % 1000;
                                     for( i=100000;i<200000;i++)
                            {String j=Integer.toString(i);
                                     oup.writeBytes(j);
                                     oup.writeByte('\n');
                                      }
                                      long millis5 =
System.currentTimeMillis() % 1000;
                                 long delt=millis5-millis4;
                                  System.out.println("dlt time"+delt);
                                     String ip22 = inp.readLine();
                                    System.out.println(ip22);
                           break;

                   default :
                          System.out.println("Wrong Entry  ");
                          break;
```

```
                       }


            System.out.println("Do you want to continue (Type y or n) \n");
            ch = scan.next().charAt(0);
             String str = Character.toString(ch);
             oup.writeBytes(str);
             oup.writeByte('\n');
        } while (ch == 'Y'|| ch == 'y');






                  }
                  k++;
                 }




                br.close();






                }

            catch (Exception e)
             {
            System.err.println("Error: " + e.getMessage());
             }
    }
}
```