ネットワークアプリケーション

第6回 Javaによるネットワークプログラミング

石井 健太郎

（423研究室・オフィスアワー火3限）

スケジュール

9月27日　第1回「TCP/IPプロトコルスイート」
10月4日　第2回「Javaによるウィンドウプログラミング」
10月11日　第3回「ネットワークアプリケーションのプログラミングモデル」
10月18日　第4回「Javaによるネットワークプログラミング」
10月25日　第5回「Javaによるネットワークプログラミング」
11月8日　第6回「Javaによるネットワークプログラミング」
11月15日　第7回「Javaによるネットワークプログラミング」　最終課題（1）
11月17日　第8回「ウェブプログラミングについて」

スケジュール

11月22日　第9回「JavaScriptによるクライアントサイドウェブプログラミング」
11月29日　第10回「JavaScriptによるクライアントサイドウェブプログラミング」
12月6日　第11回「JavaScriptによるクライアントサイドウェブプログラミング」
12月13日　第12回「JavaScriptによるクライアントサイドウェブプログラミング」
12月20日　第13回「JavaScriptによるクライアントサイドウェブプログラミング」
1月10日　第14回「JavaScriptによるクライアントサイド...」　**最終課題 (2)**
1月17日　第15回「まとめと演習」

- **演習課題を提出してください**

プログラム

- TCPTextApplicationServer
  - TCPTextApplicationServer.java
    - ウィンドウの描画・キーボード/マウス処理・main関数
  - TCPTextServer.java
    - ソケットの接続待ち受け(サーバソケット)
  - TCPTextSocket.java
    - クライアントとの通信
- TCPTextApplicationClient
  - TCPTextApplicationClient.java
    - ウィンドウの描画・キーボード/マウス処理・main関数
  - TCPTextSocket.java
    - サーバとの通信

TCPTextApplicationServer.java

SimpleTCPTextServerApplication.java | SimpleTCPTextServer.java | SimpleTCPTextSocket.java

```
44
45    // - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
46    SimpleTCPTextServerApplication(int serverPort) {
47        this.panel = new JPanel() {
48            @Override
49            protected void paintComponent(Graphics g) {
50                paintPanel((Graphics2D) g);
51            }
52        };
53        this.panel.setForeground(Color.WHITE);
54        this.panel.setBackground(Color.BLACK);
55
56        MouseAdapter mouseAdapter = new MouseAdapter() {
57            @Override
58            public void mouseMoved(MouseEvent me) {
59                mouseX = me.getX();
60                mouseY = me.getY();
61
62                // implement here
63
64                panel.repaint();
65            }
66
67            @Override
68            public void mousePressed(MouseEvent me) {
69                mouseX = me.getX();
70                mouseY = me.getY();
71                server.sendNumbers(mouseX, mouseY);
72
73                // implement here
74
75                panel.repaint();
76            }
77        };
78        this.panel.addMouseListener(mouseAdapter);
79        this.panel.addMouseMotionListener(mouseAdapter);
```

画面の描画

マウスの処理

mouseMoved () は
動かしたときの処理

mousePressed () は
クリックしたときの処理

```
81        this.server = new SimpleTCPTextServer(this, serverPort);
82        this.server.start();
83
84        this.number0 = -1;
85        this.number1 = -1;
86        this.string = "no data";
87
88        setTitle(getClass().getName());
89        setSize(INITIAL_WIDTH + 16, INITIAL_HEIGHT + 39);
90        setBackground(Color.BLACK);
91        setContentPane(this.panel);
92
93        addWindowListener(new WindowAdapter() {
94            @Override
95            public void windowClosing(WindowEvent we) {
96                quit();
97            }
98        });
99
100       addKeyListener(new KeyAdapter() {
101           @Override
102           public void keyPressed(KeyEvent ke) {
103               switch (ke.getKeyCode()) {
104               case KeyEvent.VK_ESCAPE:
105               case KeyEvent.VK_Q:
106                   quit();
107                   break;
108
109               // implement here
110
111
112               default:
113                   break;
114               }
115           }
116       });
117   }
```

キーボードの処理

ke.getKeyCode () で
入力されたキーを
取得できる

TCPTextApplicationServer.java

- 画面の描画
  - 次ページ
- マウスの処理
  - http://docs.oracle.com/javase/jp/8/api/java/awt/event/MouseEvent.html
  - http://docs.oracle.com/javase/jp/8/api/java/awt/event/MouseListener.html
  - http://docs.oracle.com/javase/jp/8/api/java/awt/event/MouseMotionListener.html
- キーボードの処理
  - http://docs.oracle.com/javase/jp/8/api/java/awt/event/KeyEvent.html
  - http://docs.oracle.com/javase/jp/8/api/java/awt/event/KeyListener.html

TCPTextApplicationServer.java

- paintPanel () で実際に何を描画するかを書いている

http://docs.oracle.com/javase/jp/8/api/java/awt/Graphics.html

```
//- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
private synchronized void paintPanel(Graphics2D g) {
    g.setColor(this.panel.getBackground());
    g.fillRect(0, 0, this.panel.getWidth(), this.panel.getHeight());

    g.setStroke(new BasicStroke(3.0f, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND));

    g.setColor(this.panel.getForeground());
    g.drawString("" + this.number0, this.panel.getWidth() / 2 - 50, this.panel.getHeight() / 2 - 25);
    g.drawString("" + this.number1, this.panel.getWidth() / 2 + 50, this.panel.getHeight() / 2 - 25);
    g.drawString(this.string, this.panel.getWidth() / 2, this.panel.getHeight() / 2 + 25);

    g.drawString(this.mouseX + ", " + this.mouseY, 25, 25);

    // implement here

}
```

TCPTextServer.java

- クライアントの待ち受け(this.serverSocket.accept())
- クライアントの相手をするスレッドの作成(client.start())
  - http://docs.oracle.com/javase/jp/8/api/java/net/ServerSocket.html
  - http://docs.oracle.com/javase/jp/8/api/java/lang/Thread.html

```
//----------------------------------
@Override
public void run() {
    try {
        System.out.println("started");
        this.serverSocket = new ServerSocket(this.port);
    } catch (IOException ioe) {
        ioe.printStackTrace();
        return;
    }

    this.enabled = true;
    while (this.enabled) {
        try {
            System.out.println("listening...");
            SimpleTCPTextSocket client = new SimpleTCPTextSocket(this.application, this.serverSocket.accept());
            System.out.println("accepted");

            client.start();
            this.clients.add(client);
        } catch (IOException ioe) {
            ioe.printStackTrace();
        }
    }
}
```

TCPTextServer.java

- ちなみに，複数クライアントを管理しており，
  すべてのクライアントにデータを送信する関数が用意されている
  - ので，自信のある人は，複数クライアントに挑戦するとよい

```
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
synchronized void sendNumbers(int numbers0, int numbers1) {
    for (SimpleTCPTextSocket client : this.clients)
        client.sendNumbers(numbers0, numbers1);
}

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
synchronized void sendString(String string) {
    for (SimpleTCPTextSocket client : this.clients)
        client.sendString(string);
}

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
synchronized void sendHoge() {
    for (SimpleTCPTextSocket client : this.clients)
        client.sendHoge();
}
```

TCPTextSocket.java

- 接続して通信に必要な BufferedReader と PrintWriter を生成する
  - http://docs.oracle.com/javase/jp/8/api/java/net/Socket.html
  - http://docs.oracle.com/javase/jp/8/api/java/io/BufferedReader.html
  - http://docs.oracle.com/javase/jp/8/api/java/io/PrintWriter.html

```
SimpleTCPTextSocket(SimpleTCPTextServerApplication application, Socket socket) {
    this.enabled = false;

    try {
        this.socket = socket;
        this.reader = new BufferedReader(new InputStreamReader(this.socket.getInputStream()));
        this.writer = new PrintWriter(this.socket.getOutputStream(), true);
    } catch (IOException ioe) {
        ioe.printStackTrace();

        if (this.writer != null) {
            this.writer.close();
            this.writer = null;
        }

        if (this.reader != null) {
            try {
                this.reader.close();
            } catch (IOException ioe2) {
                ioe2.printStackTrace();
            } finally {
                this.reader = null;
            }
        }

        if (this.socket != null) {
```

サーバ側はこちらを使う

```
SimpleTCPTextSocket(SimpleTCPTextServerApplication application, String hostname, int port) {
    this.enabled = false;

    try {
        this.socket = new Socket(hostname, port);
        this.reader = new BufferedReader(new InputStreamReader(this.socket.getInputStream()));
        this.writer = new PrintWriter(this.socket.getOutputStream(), true);
    } catch (IOException ioe) {
        ioe.printStackTrace();

        if (this.writer != null) {
            this.writer.close();
            this.writer = null;
        }

        if (this.reader != null) {
            try {
                this.reader.close();
            } catch (IOException ioe2) {
                ioe2.printStackTrace();
            } finally {
                this.reader = null;
            }
        }

        if (this.socket != null) {
```

TCPTextSocket.java

- run () で行われているのが受信
  - this.reader.readLine ()

```
//- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
@Override
public void run() {
    if (this.socket == null || this.reader == null || this.writer == null)
        return;

    this.enabled = true;

    try {
        while (this.enabled) {
            String line = this.reader.readLine();
            if (line == null)
                break;

            StringTokenizer tokens = new StringTokenizer(line);
            if (!tokens.hasMoreTokens())
                continue;

            String command = tokens.nextToken();
            if (command.equals("Numbers")) {
                if (tokens.countTokens() < 2)
                    continue;
                int number0 = Integer.parseInt(tokens.nextToken());
                int number1 = Integer.parseInt(tokens.nextToken());

                this.application.setNumbers(number0, number1);
                sendString("received! '" + line + "'");
            } else if (command.equals("String")) {
                if (tokens.countTokens() < 1)
                    continue;
                String string = line.substring("String ".length());

                this.application.setString(string);
            } else if (command.equals("Hoge")) {
                // implement here

                this.application.setHoge();
            } else {
                System.err.println("Invalid command! : " + command);
            }
        }
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
}
```

- http://docs.oracle.com/javase/jp/8/api/java/io/BufferedReader.html

TCPTextSocket.java

- 個別の関数で行われているのが送信
  - this.writer.println ()

```
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
synchronized void sendNumbers(int numbers0, int numbers1) {
    if (this.writer == null)
        return;

    String line = "Numbers " + numbers0 + " " + numbers1;
    this.writer.println(line);
}

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
synchronized void sendString(String string) {
    if (this.writer == null)
        return;

    String line = "String " + string;
    this.writer.println(line);
}

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
synchronized void sendHoge() {
    if (this.writer == null)
        return;

    String line = "Hoge ";
    this.writer.println(line);
}
```

- http://docs.oracle.com/javase/jp/8/api/java/io/PrintWriter.html

TCPTextApplicationClient.java

- （このプログラムでは，）基本的にはサーバと同じ動作
  - 次ページ以降

TCPTextApplicationClient.java

```
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
SimpleTCPTextClientApplication(String serverHost, int serverPort) {
    this.panel = new JPanel() {
        @Override
        protected void paintComponent(Graphics g) {
            paintPanel((Graphics2D) g);
        }
    };
    this.panel.setForeground(Color.WHITE);
    this.panel.setBackground(Color.BLACK);

    MouseAdapter mouseAdapter = new MouseAdapter() {
        @Override
        public void mouseMoved(MouseEvent me) {
            mouseX = me.getX();
            mouseY = me.getY();

            // implement here

            panel.repaint();
        }

        @Override
        public void mousePressed(MouseEvent me) {
            mouseX = me.getX();
            mouseY = me.getY();
            socket.sendNumbers(mouseX, mouseY);

            // implement here

            panel.repaint();
        }
    };
    this.panel.addMouseListener(mouseAdapter);
    this.panel.addMouseMotionListener(mouseAdapter);

    this.socket = new SimpleTCPTextSocket(this, serverHost, serverPort);
    this.socket.start();
```

画面の描画

マウスの処理

mouseMoved () は
動かしたときの処理

mousePressed () は
クリックしたときの処理

```
    this.number0 = -1;
    this.number1 = -1;
    this.string = "no data";

    setTitle(getClass().getName());
    setSize(INITIAL_WIDTH + 16, INITIAL_HEIGHT + 39);
    setBackground(Color.BLACK);
    setContentPane(this.panel);

    addWindowListener(new WindowAdapter() {
        @Override
        public void windowClosing(WindowEvent we) {
            quit();
        }
    });

    addKeyListener(new KeyAdapter() {
        @Override
        public void keyPressed(KeyEvent ke) {
            switch (ke.getKeyCode()) {
            case KeyEvent.VK_ESCAPE:
            case KeyEvent.VK_Q:
                quit();
                break;

            // implement here


            default:
                break;
            }
        }
    });
}
```

キーボードの処理

ke.getKeyCode () で
入力されたキーを
取得できる

TCPTextApplicationClient.java

- 画面の描画
  - 次ページ
- マウスの処理
  - http://docs.oracle.com/javase/jp/8/api/java/awt/event/MouseEvent.html
  - http://docs.oracle.com/javase/jp/8/api/java/awt/event/MouseListener.html
  - http://docs.oracle.com/javase/jp/8/api/java/awt/event/MouseMotionListener.html
- キーボードの処理
  - http://docs.oracle.com/javase/jp/8/api/java/awt/event/KeyEvent.html
  - http://docs.oracle.com/javase/jp/8/api/java/awt/event/KeyListener.html

TCPTextApplicationClient.java

- paintPanel () で実際に何を描画するかを書いている

http://docs.oracle.com/javase/jp/8/api/java/awt/Graphics.html

```
//- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
private synchronized void paintPanel(Graphics2D g) {
    g.setColor(this.panel.getBackground());
    g.fillRect(0, 0, this.panel.getWidth(), this.panel.getHeight());

    g.setStroke(new BasicStroke(3.0f, BasicStroke.CAP_ROUND, BasicStroke.JOIN_ROUND));

    g.setColor(this.panel.getForeground());
    g.drawString("" + this.number0, this.panel.getWidth() / 2 - 50, this.panel.getHeight() / 2 - 25);
    g.drawString("" + this.number1, this.panel.getWidth() / 2 + 50, this.panel.getHeight() / 2 - 25);
    g.drawString(this.string, this.panel.getWidth() / 2, this.panel.getHeight() / 2 + 25);

    g.drawString(this.mouseX + ", " + this.mouseY, 25, 25);

    // implement here

}
```

TCPTextSocket.java

- 接続して通信に必要な BufferedReader と PrintWriter を生成する
  - http://docs.oracle.com/javase/jp/8/api/java/net/Socket.html
  - http://docs.oracle.com/javase/jp/8/api/java/io/BufferedReader.html
  - http://docs.oracle.com/javase/jp/8/api/java/io/PrintWriter.html

```
SimpleTCPTextSocket(SimpleTCPTextClientApplication application, Socket socket) {
    this.enabled = false;

    try {
        this.socket = socket;
        this.reader = new BufferedReader(new InputStreamReader(this.socket.getInputStream()));
        this.writer = new PrintWriter(this.socket.getOutputStream(), true);
    } catch (IOException ioe) {
        ioe.printStackTrace();

        if (this.writer != null) {
            this.writer.close();
            this.writer = null;
        }

        if (this.reader != null) {
            try {
                this.reader.close();
            } catch (IOException ioe2) {
                ioe2.printStackTrace();
            } finally {
                this.reader = null;
            }
        }

        if (this.socket != null) {
```

```
SimpleTCPTextSocket(SimpleTCPTextClientApplication application, String hostname, int port) {
    this.enabled = false;

    try {
        this.socket = new Socket(hostname, port);
        this.reader = new BufferedReader(new InputStreamReader(this.socket.getInputStream()));
        this.writer = new PrintWriter(this.socket.getOutputStream(), true);
    } catch (IOException ioe) {
        ioe.printStackTrace();

        if (this.writer != null) {
            this.writer.close();
            this.writer = null;
        }

        if (this.reader != null) {
            try {
                this.reader.close();
            } catch (IOException ioe2) {
                ioe2.printStackTrace();
            } finally {
                this.reader = null;
            }
        }

        if (this.socket != null) {
```

クライアント側はこちらを使う

TCPTextSocket.java

- run () で行われているのが受信
  - this.reader.readLine ()

```
//- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
@Override
public void run() {
    if (this.socket == null || this.reader == null || this.writer == null)
        return;

    this.enabled = true;

    try {
        while (this.enabled) {
            String line = this.reader.readLine();
            if (line == null)
                break;

            StringTokenizer tokens = new StringTokenizer(line);
            if (!tokens.hasMoreTokens())
                continue;

            String command = tokens.nextToken();
            if (command.equals("Numbers")) {
                if (tokens.countTokens() < 2)
                    continue;
                int number0 = Integer.parseInt(tokens.nextToken());
                int number1 = Integer.parseInt(tokens.nextToken());

                this.application.setNumbers(number0, number1);
                sendString("received! '" + line + "'");
            } else if (command.equals("String")) {
                if (tokens.countTokens() < 1)
                    continue;
                String string = line.substring("String ".length());

                this.application.setString(string);
            } else if (command.equals("Hoge")) {
                // implement here

                this.application.setHoge();
            } else {
                System.err.println("Invalid command! : " + command);
            }
        }
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
}
```

- http://docs.oracle.com/javase/jp/8/api/java/io/BufferedReader.html

TCPTextSocket.java

- 個別の関数で行われているのが送信
  - this.writer.println ()

```
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
synchronized void sendNumbers(int numbers0, int numbers1) {
    if (this.writer == null)
        return;

    String line = "Numbers " + numbers0 + " " + numbers1;
    this.writer.println(line);
}

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
synchronized void sendString(String string) {
    if (this.writer == null)
        return;

    String line = "String " + string;
    this.writer.println(line);
}

// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
synchronized void sendHoge() {
    if (this.writer == null)
        return;

    String line = "Hoge ";
    this.writer.println(line);
}
```

- http://docs.oracle.com/javase/jp/8/api/java/io/PrintWriter.html

プログラム

- TCPBinaryApplicationServer
  - TCPBinaryApplicationServer.java
    - ウィンドウの描画・キーボード/マウス処理・main関数
  - TCPBinaryServer.java
    - ソケットの接続待ち受け(サーバソケット)
  - TCPBinarySocket.java
    - クライアントとの通信
- TCPBinaryApplicationClient
  - TCPBinaryApplicationClient.java
    - ウィンドウの描画・キーボード/マウス処理・main関数
  - TCPBinarySocket.java
    - サーバとの通信

TCPBinaryApplication {Server,Client} について

- 通信方式以外は TCPTextApplication {Server,Client} と同じ
  - 次ページ以降では異なる部分を説明する

TCPBinarySocket.java

- 接続して通信に必要な DataInputStream と DataOuputStream を生成する
  - http://docs.oracle.com/javase/jp/8/api/java/net/Socket.html
  - http://docs.oracle.com/javase/jp/8/api/java/io/DataInputStream.html
  - http://docs.oracle.com/javase/jp/8/api/java/io/DataOutputStream.html

```
SimpleTCPBinarySocket(SimpleTCPBinaryServerApplication application, Socket socket) {
    this.enabled = false;

    try {
        this.socket = socket;
        this.input = new DataInputStream(this.socket.getInputStream());
        this.output = new DataOutputStream(this.socket.getOutputStream());
    } catch (IOException ioe) {
        ioe.printStackTrace();

        if (this.output != null) {
            try {
                this.output.close();
            } catch (IOException ioe2) {
                ioe2.printStackTrace();
            } finally {
                this.output = null;
            }
        }

        if (this.input != null) {
            try {
                this.input.close();
            } catch (IOException ioe2) {
                ioe2.printStackTrace();
            } finally {
```

サーバ側はこちらを使う

```
SimpleTCPBinarySocket(SimpleTCPBinaryServerApplication application, String hostname, int port) {
    this.enabled = false;

    try {
        this.socket = new Socket(hostname, port);
        this.input = new DataInputStream(this.socket.getInputStream());
        this.output = new DataOutputStream(this.socket.getOutputStream());
    } catch (IOException ioe) {
        ioe.printStackTrace();

        if (this.output != null) {
            try {
                this.output.close();
            } catch (IOException ioe2) {
                ioe2.printStackTrace();
            } finally {
                this.output = null;
            }
        }

        if (this.input != null) {
            try {
                this.input.close();
            } catch (IOException ioe2) {
                ioe2.printStackTrace();
            } finally {
```

クライアント側はこちらを使う

TCPBinarySocket.java

- run () で行われているのが受信
  - this.input.readChar ()

```
// - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
@Override
public void run() {
    if (this.socket == null || this.input == null || this.output == null)
        return;

    this.enabled = true;

    try {
        while (this.enabled) {
            char command = this.input.readChar();
            if (command == 'N') {
                //implement here

            } else if (command == 'S') {
                //implement here

            } else if (command == 'H') {
                // implement here

                this.application.setHoge();
            } else {
                System.err.println("Invalid command! : " + command);
            }
        }
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
}
```

- http://docs.oracle.com/javase/jp/8/api/java/io/DataInputStream.html

TCPBinarySocket.java

- 個別の関数で行われているのが送信
  - this.output.writeChar ()
  - this.output.writeInt ()
  - this.output.write ()

```
//----------------------------------------------------------------
synchronized void sendNumbers(int numbers0, int numbers1) {
    if (this.output == null)
        return;

    try {
        this.output.writeChar('N');
        this.output.writeInt(numbers0);
        this.output.writeInt(numbers1);
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
}

//----------------------------------------------------------------
synchronized void sendString(String string) {
    if (this.output == null)
        return;

    byte[] bytes = string.getBytes();
    try {
        this.output.writeChar('S');
        this.output.writeInt(bytes.length);
        this.output.write(bytes);
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
}

//----------------------------------------------------------------
synchronized void sendHoge() {
    if (this.output == null)
        return;

    try {
        this.output.writeChar('H');
    } catch (IOException ioe) {
        ioe.printStackTrace();
    }
}
```

- http://docs.oracle.com/javase/jp/8/api/java/io/DataOutputStream.html

演習(再掲＆提出期限変更)

- TCPによるデータの送受信を用いて，ウィンドウアプリケーションを作成する
  - まず，テキストデータにするか，バイナリデータにするか，選択する
  - 基本課題
    - マウスのクリック位置を相手がたプログラムに送信して，そのデータを文字で表示させる
    - さらに，受信したデータに応じた位置(相手のクリック位置)に図形を描画させるなど，見た目におもしろいプログラムに仕上げよう
  - 発展課題
    - クリック位置だけじゃなく，いろんなデータを送受信して，おもしろいアプリケーションを考えよう
      - 例えば，ドラッグすると相手がたに線が引ける協調お絵かきソフトとするなど

- プログラム(TCP*.java)の内容を，1つのテキストファイルにコピーして，第8回(提出期限変更)の講義開始時までに，提出フォルダ(Xドライブ)にファイルで提出
  - X:¥IN科専門¥石井講師¥ネットワークアプリケーション¥第5回
  - ファイル名は「<学生番号>.txt」とする(ハイフンなし)　例: N14999.txt