ネットワークアプリケーション

第5回 アプリケーションプロトコルの設計 (3)

石井 健太郎
(423研究室・オフィスアワー水3限)

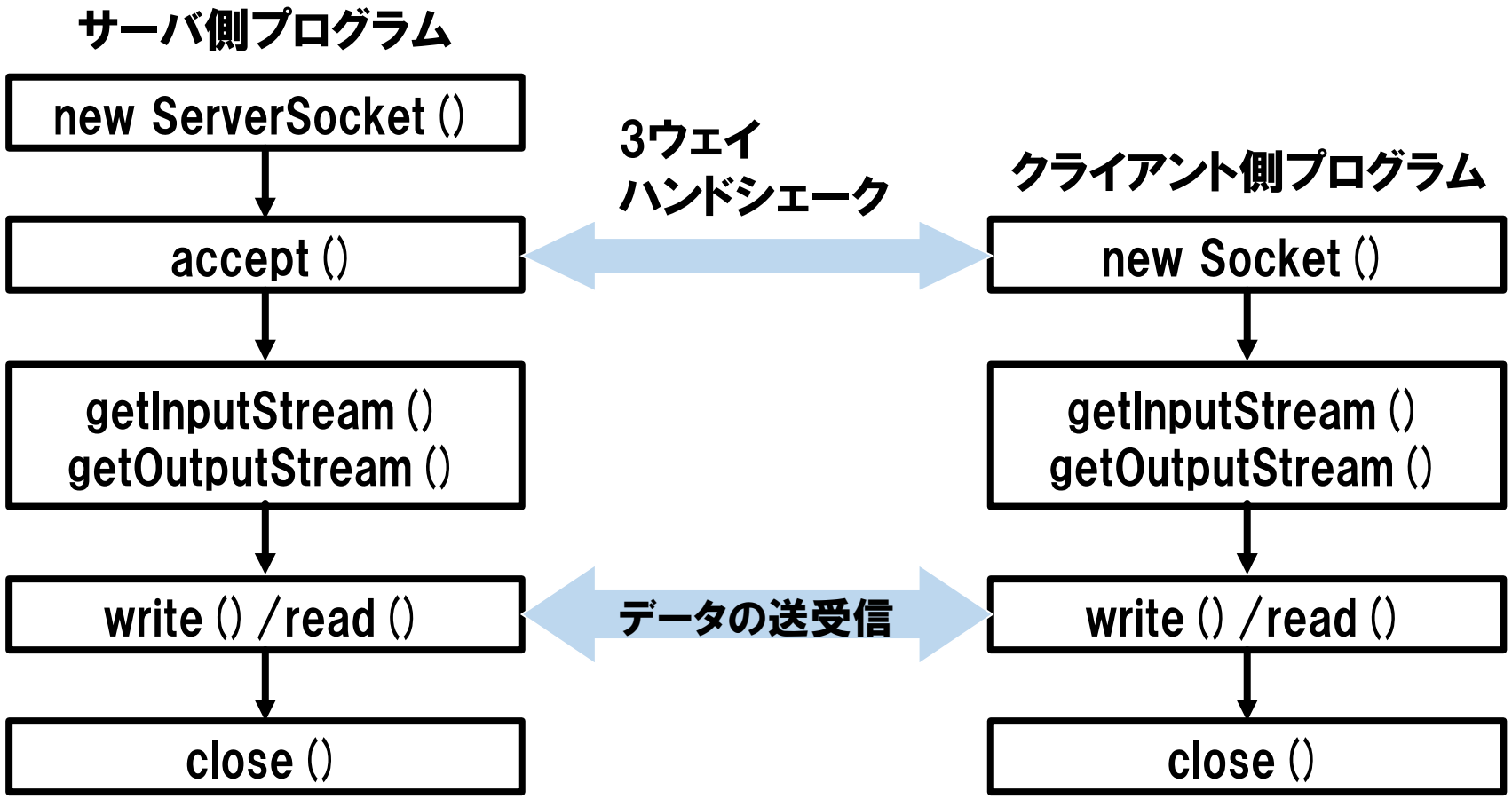
スケジュール

9月15日　第1回「TCP/IPプロトコルスイート」
9月29日　第2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6日　第3回「アプリケーションプロトコルの設計（1）」
10月13日　第4回「アプリケーションプロトコルの設計（2）」
10月20日　第5回「アプリケーションプロトコルの設計（3）」　演習（第3演習室）
10月27日　第6回「アプリケーションプロトコルの設計（4）」　演習（第3演習室）
11月10日　第7回「サーバサイドウェブプログラミング（1）」
11月17日　第8回「サーバサイドウェブプログラミング（2）」

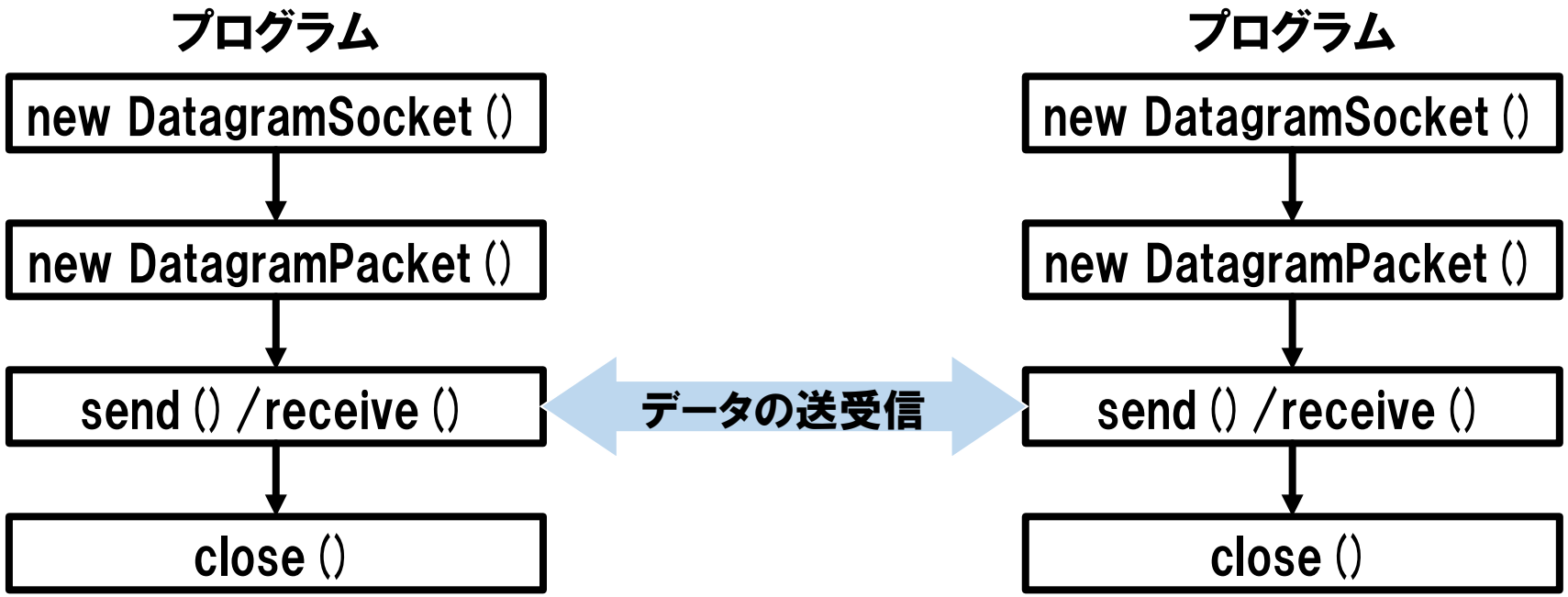
スケジュール

11月24日	第9回「サーバサイドウェブプログラミング（3）」	演習（第3演習室）
12月1日	第10回「サーバサイドウェブプログラミング（4）」	演習（第3演習室）
12月8日	第11回「クライアントサイドウェブプログラミング（1）」	
12月15日	第12回「クライアントサイドウェブプログラミング（2）」	
12月22日	第13回「クライアントサイドウェブプログラミング（3）」	演習（第3演習室）
1月12日	第14回「クライアントサイドウェブプログラミング（4）」	演習（第3演習室）
1月19日	第15回「まとめと演習」	演習（第3演習室）

（参考）TCPによるデータの送受信（Java編）



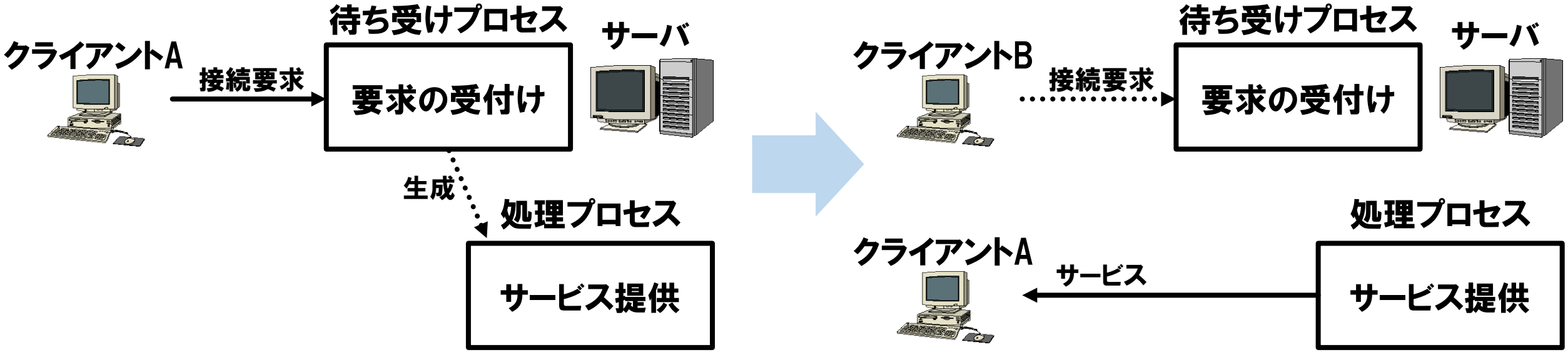
（参考）UDPによるデータの送受信（Java編）



マルチプロセス化・マルチスレッド化

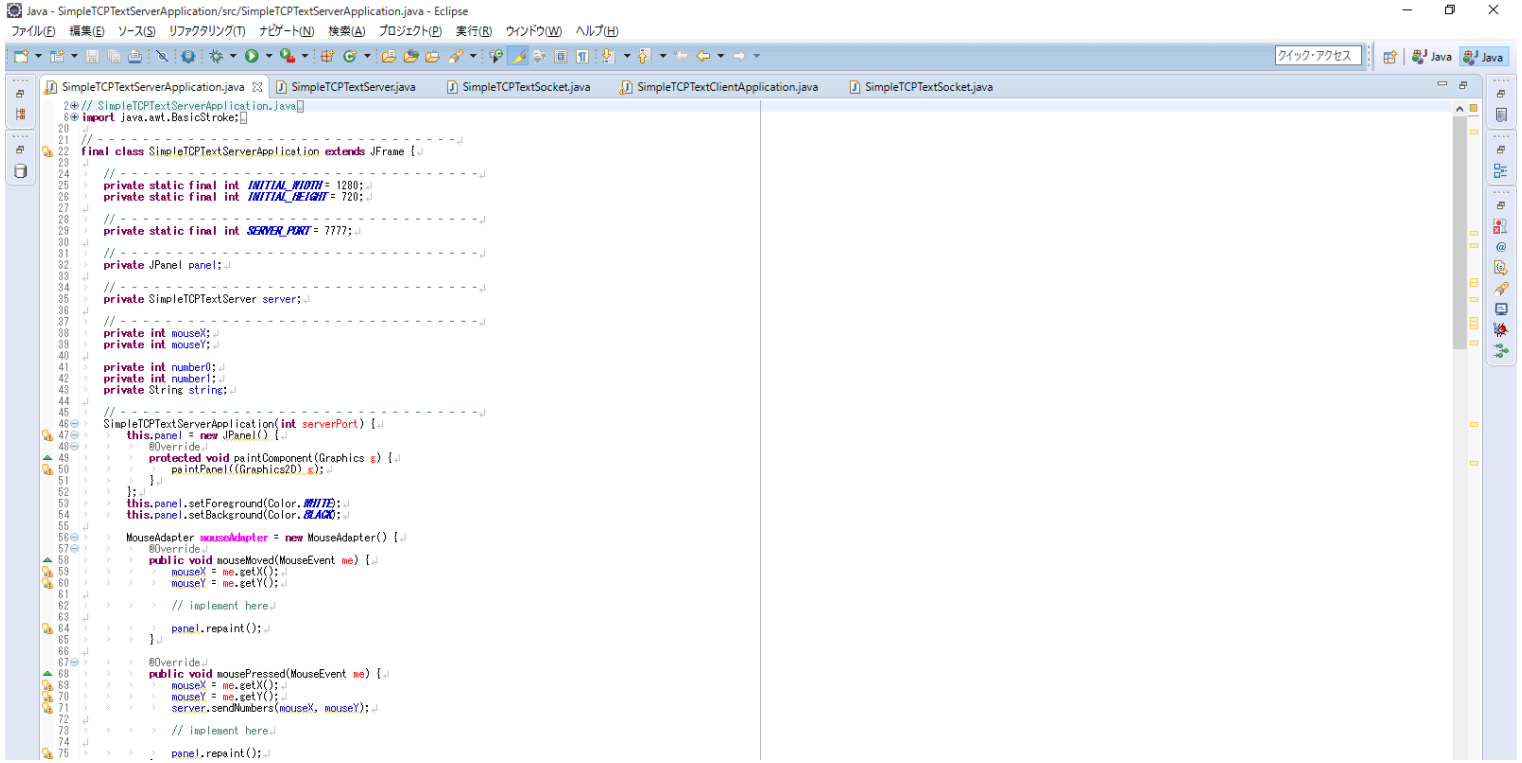
- accept () , recv () , recvfrom () はブロッキングコール
  - あるプログラムの接続・データを待っているとほかの処理を一切行えない

- マルチプロセス化 (fork) ・マルチスレッド化による並行処理が必要



プログラム

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


Java - SimpleTCPTextServerApplication/src/SimpleTCPTextServerApplication.java - Eclipse
ファイル(F) 編集(E) ソース(S) リファクタリング(T) ナビゲート(N) 検索(A) プロジェクト(P) 実行(R) ウィンドウ(W) ヘルプ(H)
SimpleTCPTextServerApplication.java
SimpleTCPTextServer.java
SimpleTCPTextSocket.java
SimpleTCPTextClientApplication.java
SimpleTCPTextSocket.java
// SimpleTCPTextServerApplication.java
import java.awt.BasicStroke;
final class SimpleTCPTextServerApplication extends JFrame {
private static final int INITIAL_WIDTH = 1280;
private static final int INITIAL_HEIGHT = 720;
private static final int SERVER_PORT = 7777;
private JPanel panel;
private SimpleTCPTextServer server;
private int mouseX;
private int mouseY;
private int number0;
private int number1;
private String string;
SimpleTCPTextServerApplication(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();
server.sendNumbers(mouseX, mouseY);
// implement here
panel.repaint();


this.panel.repaint();
}
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
}
public static void main(String args[]) {
EventQueue.invokeLater(new Runnable() {
@Override
public void run() {
try {
SimpleTCPTextServerApplication frame = new SimpleTCPTextServerApplication(SERVER_PORT);
frame.setVisible(true);
} catch (Exception e) {
e.printStackTrace();
}
}
});
}
}
一致なし
書き込み可能
スマート挿入
110 : 1
112M / 448M

はじめに

- クラスライブラリをいろいろ参照しながら理解する必要がある
  - http://docs.oracle.com/javase/jp/8/api/

SimpleTCPTextServerApplication.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

SimpleTCPTextServerApplication.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

SimpleTCPTextServerApplication.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

}
```

SimpleTCPTextServer.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();
        }
    }
}
```

SimpleTCPTextServer.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();
}
```

SimpleTCPTextSocket.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) {
```

SimpleTCPTextSocket.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

SimpleTCPTextSocket.java

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

```java
// ----------------------------------------
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

SimpleTCPTextClientApplication.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

SimpleTCPTextClientApplication.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

SimpleTCPTextClientApplication.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

SimpleTCPTextClientApplication.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

}
```

SimpleTCPTextSocket.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

SimpleTCPTextSocket.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

SimpleTCPTextSocket.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

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

SimpleBinary {Server,Client} について

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

SimpleTCPBinarySocket.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

SimpleTCPBinarySocket.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

SimpleTCPBinarySocket.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

提出課題

- SimpleBinaryServer/SimpleBinaryClient は
  バイナリデータを送受信することで，
  マウスの座標・メッセージを交換するプログラム（未完）である
  - サーバ側・クライアント側ともに，
    送信部分は実装されているが，受信部分は実装されていない
- 受信部分を実装し，SimpleTextServer/SimpleTextClient と
  同様に動作するように完成させる

- 提出期限: 12月22日（火）深夜まで
- 提出方法
  - サーバ・クライアントともに，1つのフォルダにいれて，
    <学籍番号>.zip というファイルに圧縮して提出
  - 課題提出用ドライブ（Xドライブ）SimpleTCPBinaryApplication フォルダに置く

提出課題

```
//- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
@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();
    }
}
```

- ヒント
  - SimpleBinarySocket.java の run () で受信部分を記述している
  - this.input.readChar () で1文字受信できるが，int型のデータや文字データを受信するにはどうすればよいかをクラスライブラリのドキュメントを読んで考える

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

最終課題について（一般的な話）

- 選択制
  - 2つのテーマの中から好きなものを選んで提出する
    - 独自のアプリケーションプロトコル
    - クライアントサイドウェブプログラミング
  - 複数のテーマの課題が提出された場合，最も高い点数のテーマのみ採用する

- 先願優先主義
  - 内容に重複があった場合，（特許と同じ）先願優先主義にて採点する
  - つまり，同じような内容の課題が提出された場合，
    あとに提出された課題の評価を減点する

最終課題について（一般的な話）

- プログラム実行に基づく評価
  - プログラムのソースコードを提出
    → こちらでコンパイル・設置のうえ実行した結果により点数をつける
    - どんなに設計がよくできていても機能が充実していても，
      実行できない場合点数がつきません．
      ただし，実行できない場合はメールにて連絡するようにします．
  - プログラムの起動方法・必要な外部ライブラリ・設定ファイルの配置方法など，
    プログラムを実行するために必要な説明を記した簡単なドキュメントも提出
  - その他任意で，ドキュメント以外の参考資料も提出可
    - 例えば，正しく動作しているときの動画などもOK

- 最終提出期限
  - 試験期間の当日（その日の夜まで）にする予定（スケジュールが固まってきたら決定）

最終課題について（独自アプリケーション）

- ソケット通信を行うアプリケーションを考え，
  通信プロトコルを各自設計の上，
  それを実装したプログラムを提出せよ
  - Javaのサンプルプログラムを拡張して画面に何かを表示するプログラムを作成することを推奨する
    - C言語のプログラムでも可とするが，専門演習と全く同じプログラムは評価しない
  - 2人1組でサーバクライアントプログラムを書くことを可とする
    - ただし，プログラム同士連携のために繰り返し修正が必要なので，おすすめしない

- 提出方法
  - ソースコード・実行するための説明を記したドキュメント・必要なライブラリ・参考資料を，すべて1つのフォルダにいれて，＜学籍番号＞.zip というファイルに圧縮して提出
  - 課題提出用ドライブ（Xドライブ）最終課題（独自アプリケーション）フォルダに置く

- 来週10月27日（火）も**第3演習室**に集まってください