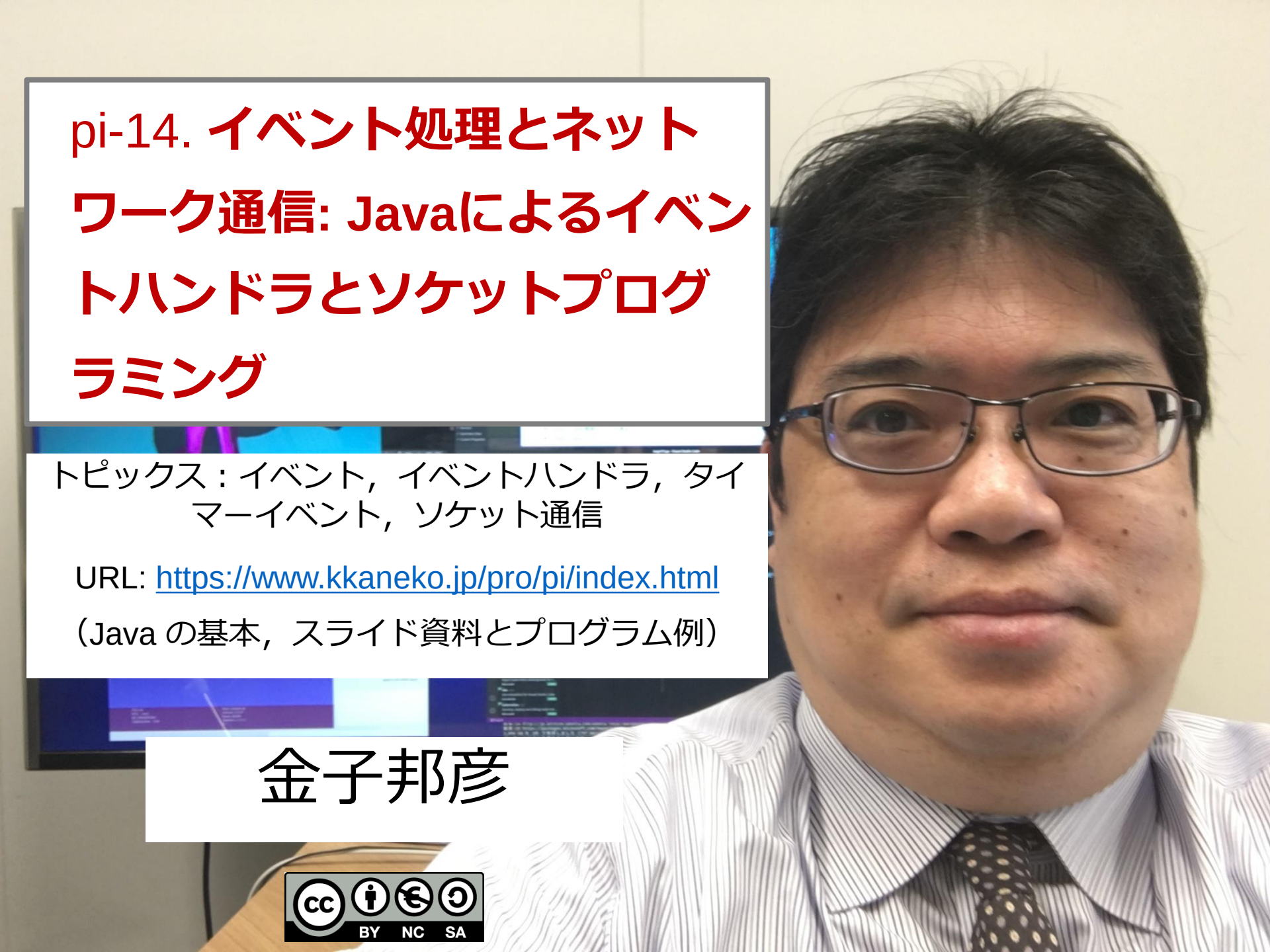
pi-14. イベント処理とネットワーク通信: Javaによるイベントハンドラとソケットプログラミング

トピックス：イベント，イベントハンドラ，タイマーイベント，ソケット通信

URL: https://www.kkaneko.jp/pro/pi/index.html

（Java の基本，スライド資料とプログラム例）



金子邦彦

今回の内容

- イベントの処理

  イベントハンドラ、イベントの登録のプログラム、

  マルチスレッドの仕組み
- 通信の基礎

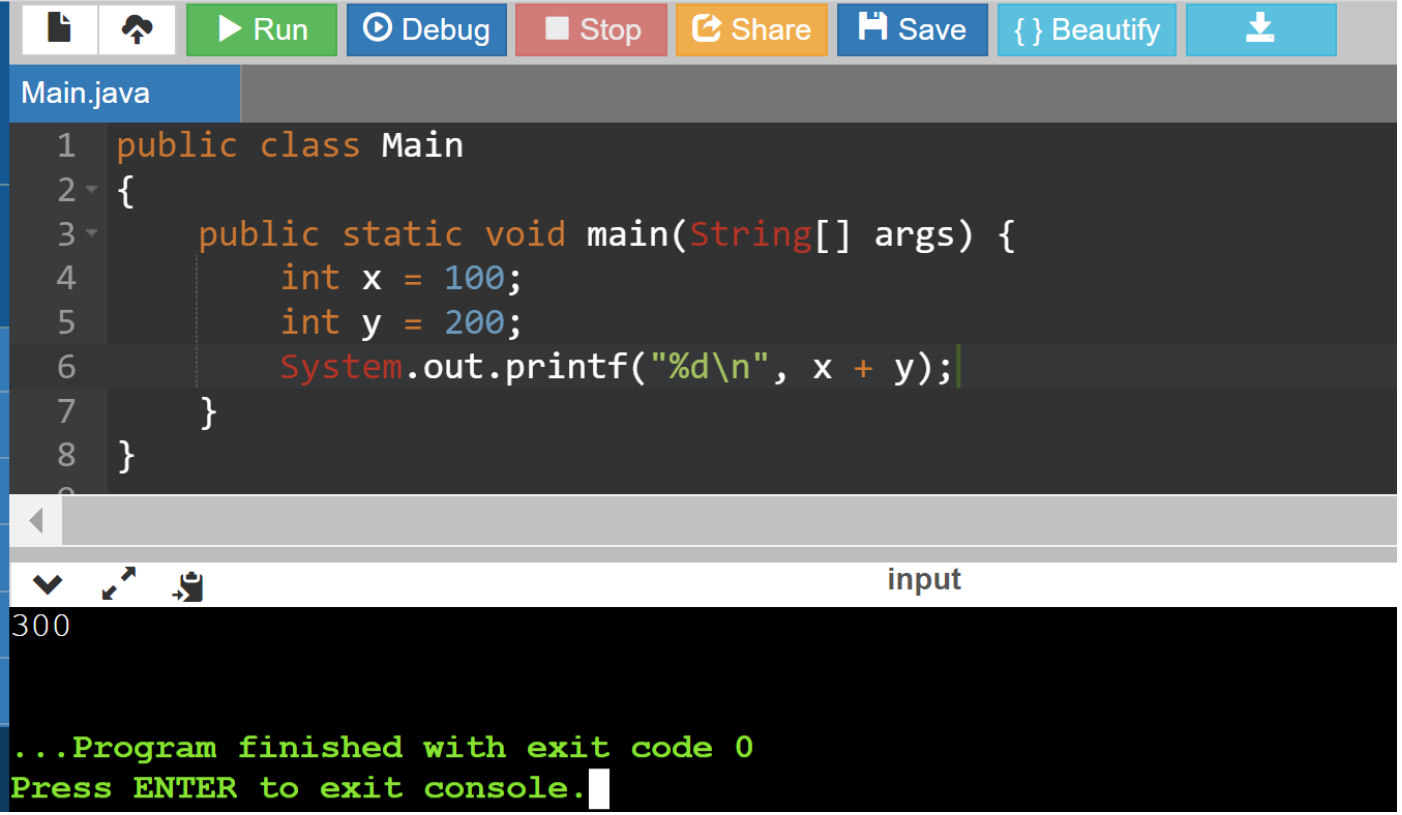
  ソケットのバインド，ソケットの接続要求

アウトライン

番号	項目
	復習
14-1	イベントとイベントハンドラ
14-2	Java でのタイマーイベント
14-3	タイマーイベントのプログラム例
14-4	ソケット通信
14-5	Java でのソケット通信

各自、資料を読み返したり、課題に取り組んだりも行う

この授業では、**Java** を用いて基礎を学び、マスターする



Java などのプログラミング言語の体験，演習ができる**オンラインサービス**

GDB online

http://www.pythontutor.com/

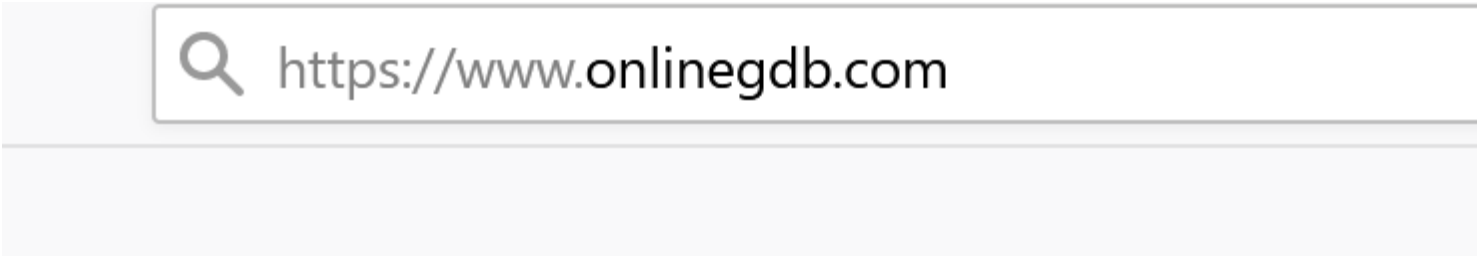
オンラインなので、「秘密にしたいプログラム」を扱うには十分な注意が必要

GDB online で Java を動かす手順

① ウェブブラウザを起動する

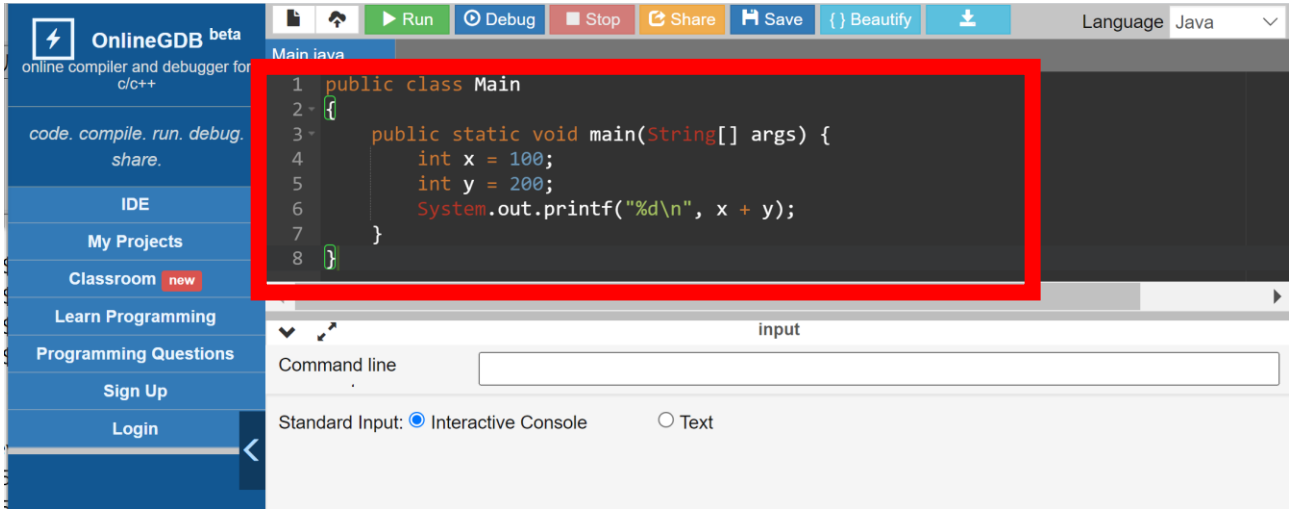
② 次の URL を開く

https://www.onlinegdb.com



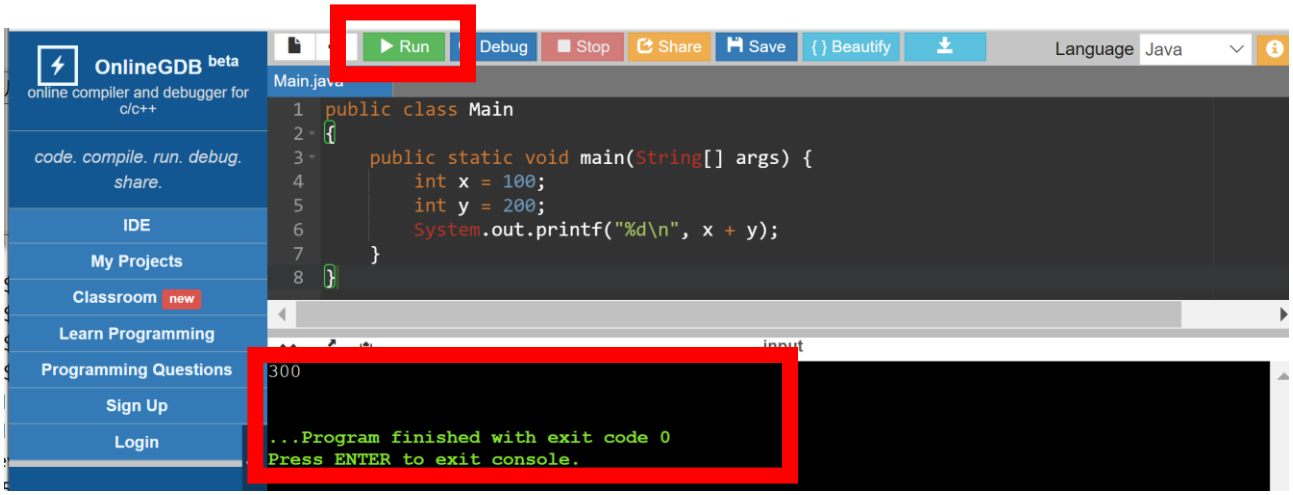
③「Language」のところで，「Java」を選ぶ

④ ソースコードを入れる



⑤ 実行．実行結果を確認

「**Run**」をクリック．



14-1. イベントとイベントハンドラ

イベントとイベントハンドラ

イベントの例

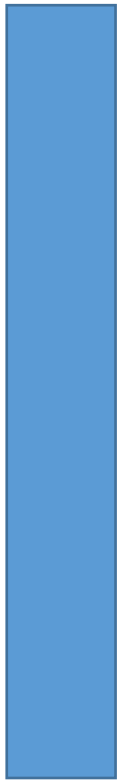
- タイマー
- 外部操作（キーボード、マウスの操作）

イベントハンドラ

イベントが発生したときの処理のこと．特定の**イベント**ごとに、事前に、行うべき処理を登録しておく．

通常のプログラム実行

イベント発生の**たび**に
マルチスレッド実行



14-2. Java でのタイマーイベント

演習

資料：**13 ～ 18**

【トピックス】

・タイマーイベント

処理を止める

① 次のソースコードを入れる

```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) {
5          System.out.println("start");
6          try {
7              Thread.sleep(100000);
8          } catch(InterruptedException e) {}
9      }
10 }
```

100秒止まる
「**100000**」
とあるのは
ミリ秒単位

② 実行結果の確認　（あとで使うので、ブラウザを閉じないこと）

「start」と表示される

Java でのタイマーイベントとイベントハンドラ

イベントハンドラ

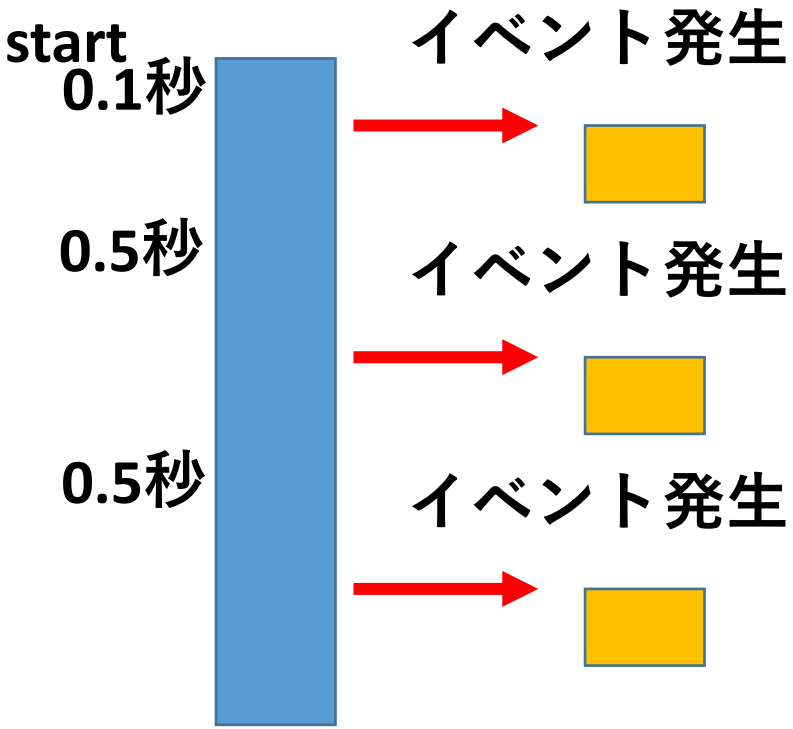
```
1  import java.util.*;
2
3  class MyTask extends TimerTask {
4      public void run() {
5          System.out.println("hello");
6      }
7  }
8
9  public class Main
10 {
11     public static void main(String[] args) {
12         Timer timer = new Timer();
13         timer.schedule(new MyTask(), 100, 500);
14         System.out.println("start");
15         try {
16             Thread.sleep(100000);
17         } catch(InterruptedException e) {}
18     }
19 }
```

- 「TimerTask」は Java の標準ライブラリのクラス

イベントハンドラの登録

1. 最初 0.1 秒経過したらイベント発生
2. その後，0.5 秒間隔でイベント発生

マルチスレッドでの実行



メインのスレッド（青）は動き続ける

タイマーイベント

③ 次のソースコードを入れる

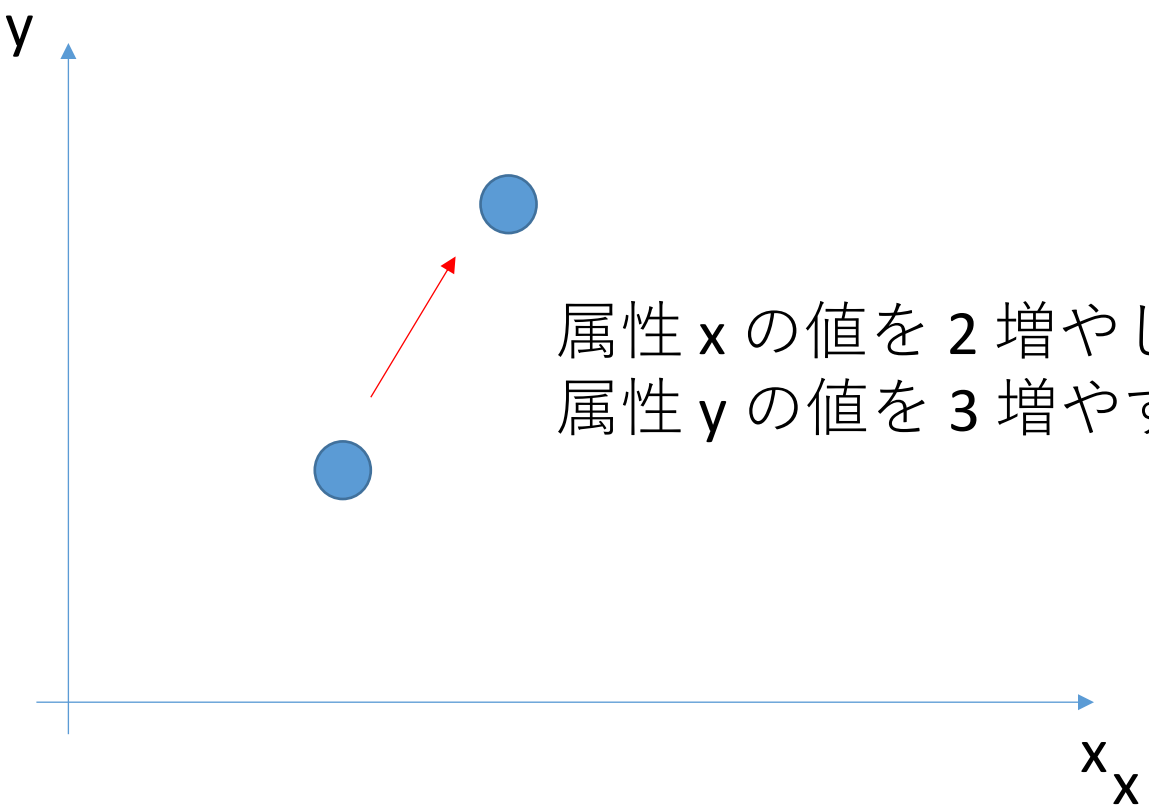
```
1  import java.util.*;
2
3  class MyTask extends TimerTask {
4      public void run() {
5          System.out.println("hello");
6      }
7  }
8
9  public class Main
10 {
11     public static void main(String[] args) {
12         Timer timer = new Timer();
13         timer.schedule(new MyTask(), 100, 500);
14         System.out.println("start");
15         try {
16             Thread.sleep(100000);
17         } catch(InterruptedException e) {}
18     }
19 }
```

④ 実行結果の確認　（あとで使うので、ブラウザを閉じないこと）

「start」と表示される

14-3. タイマーイベントのプログラム例

Ball クラスのオブジェクト



属性 x の値を 2 増やし,
属性 y の値を 3 増やすメソッド **move**

演習

資料：**22 ～ 27**

【トピックス】

・タイマーイベント

① 次のソースコードを入れる

```
class Ball {
    double x;
    double y;
    public Ball(double x, double y) {
        this.x = x;
        this.y = y;
    }
    public void move() {
        this.x = this.x + 2;
        this.y = this.y + 3;
    }
    public void printout() {
        System.out.printf("%f %f\n", this.x, this.y);
    }
}
```

次のページに続く

```
16
17  public class Main
18  {
19    public static void main(String[] args) {
20      Ball b = new Ball(0, 0);
21      b.printout();
22      b.move();
23      b.printout();
24    }
25  }
```

② 実行結果の確認

```
0.000000 0.000000
2.000000 3.000000
```

- **0.5 秒ごとに動かす**

  0.5 秒ごとにイベント発生

③ 次のように書き換える

```java
import java.util.*;

class Ball {
  double x;
  double y;
  public Ball(double x, double y) {
    this.x = x;
    this.y = y;
  }
  public void move() {
    this.x = this.x + 2;
    this.y = this.y + 3;
  }
  public void printout() {
    System.out.printf("%f %f\n", this.x, this.y);
  }
}
```

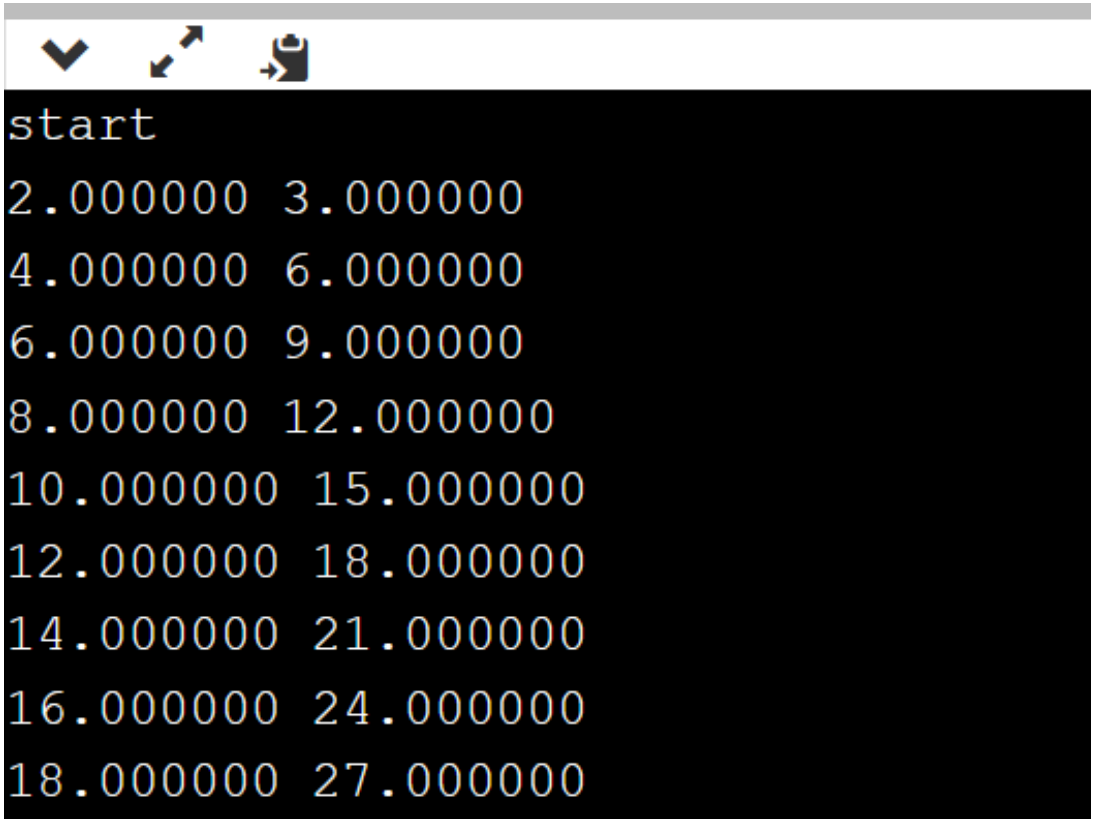
次のページに続く

```
19  class MyTask extends TimerTask {
20      Ball b;
21      public MyTask() {
22          super();
23          this.b = new Ball(0, 0);
24      }
25      public void run() {
26          b.move();
27          b.printout();
28      }
29  }
30
31  public class Main
32  {
33    public static void main(String[] args) {
34      Timer timer = new Timer();
35      timer.schedule(new MyTask(), 100, 500);
36      System.out.println("start");
37      try {
38        Thread.sleep(100000);
39      } catch(InterruptedException e) {}
40    }
41  }
```

イベントハンドラ

イベントハンドラの登録と、メインのスレッド

④ 実行結果の確認



0.5 秒ごとに動く

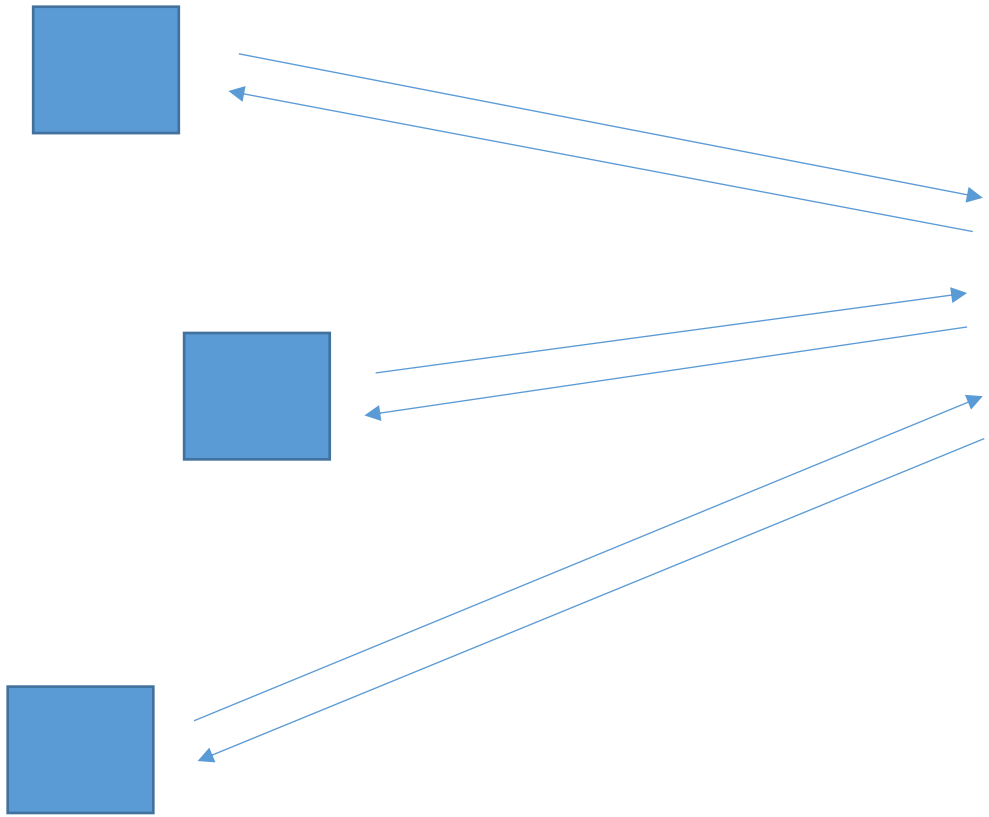
14-4. ソケット通信

ソケットとは

ソケットは，機器の間の通信や，プロセス間通信に使用されるインターフェース.

インターネットでの通信

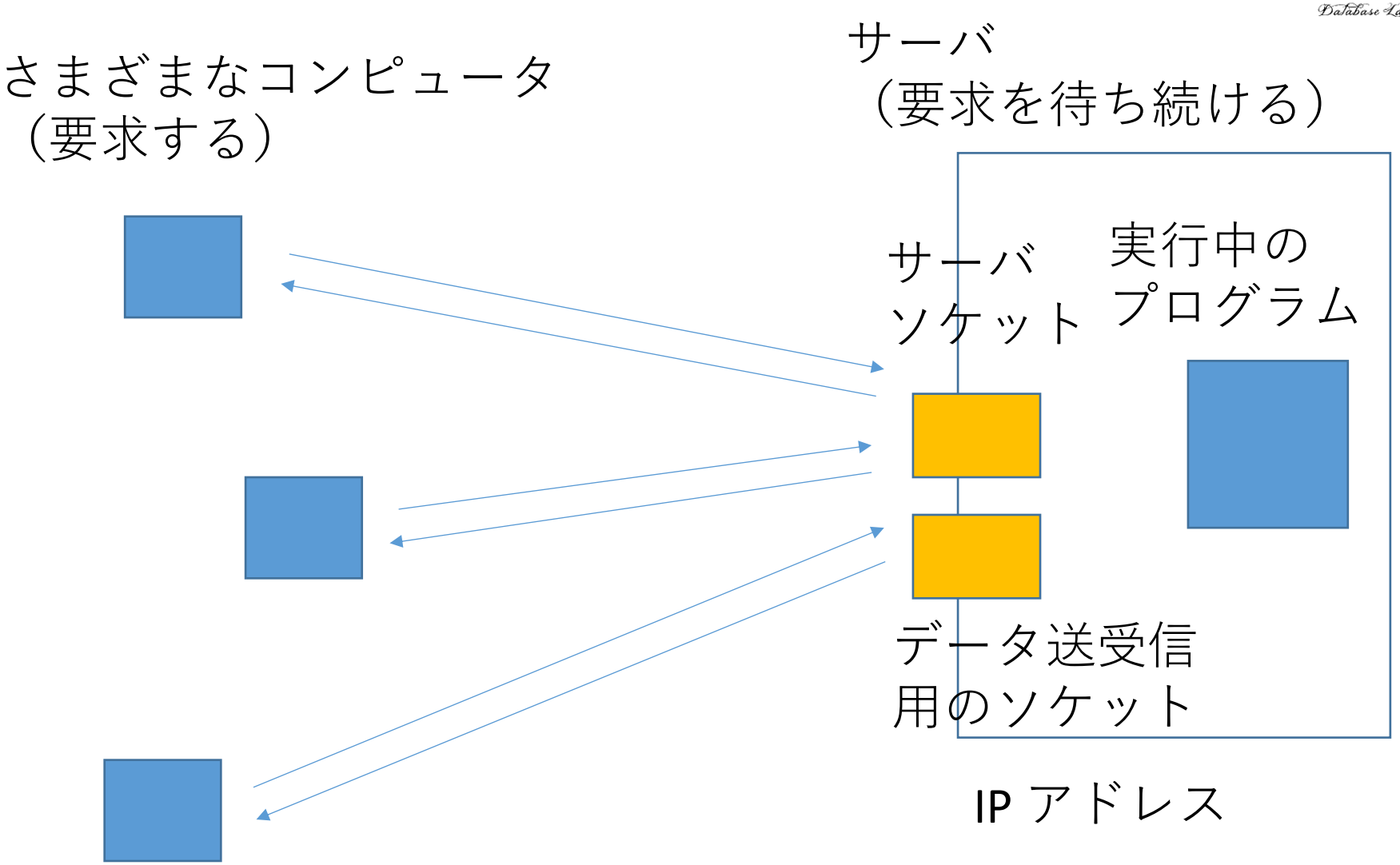
さまざまなコンピュータ
（要求する）



サーバ
（要求を待ち続ける）

IP アドレス

プログラムにバインドされたポート



インターネットでの通信

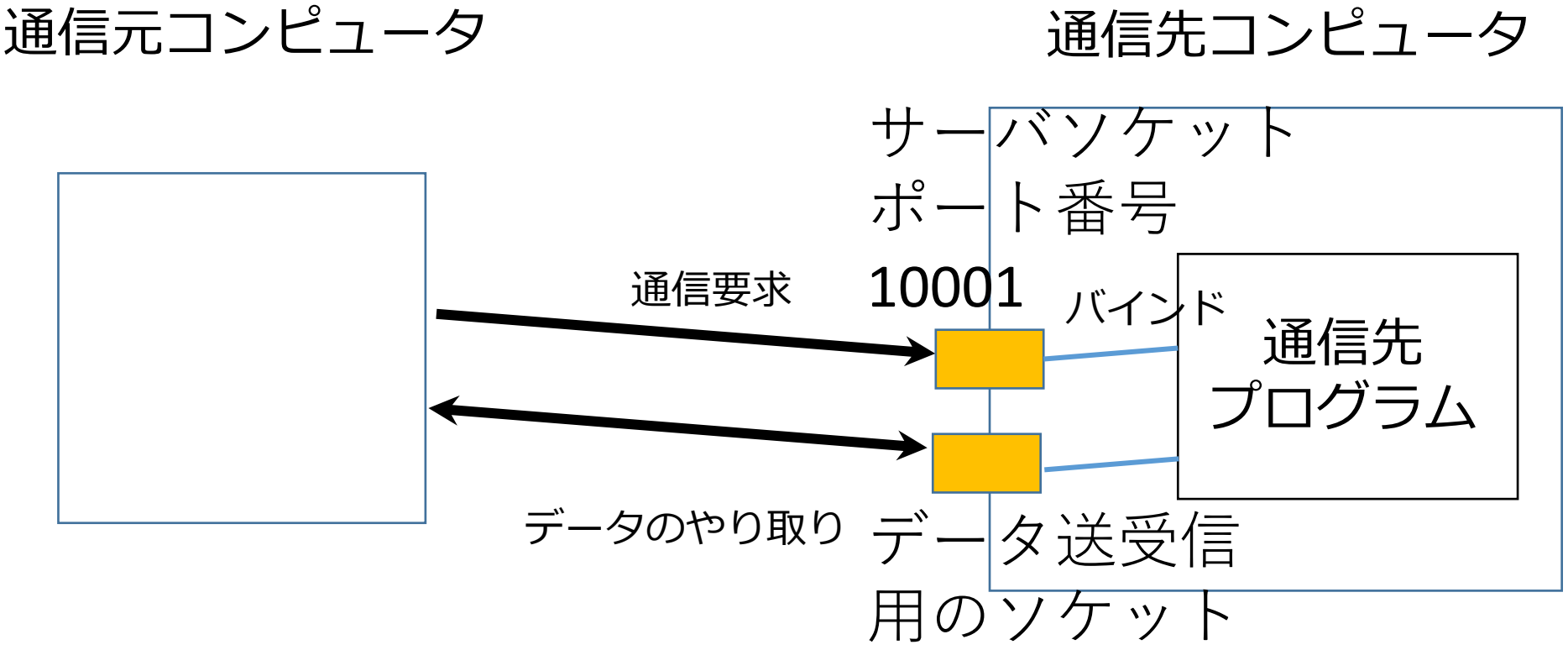
- サーバ（要求を待ち続ける）

実行中のプログラムは、**サーバソケット**に**バインド**される

- さまざまなコンピュータ（要求する）

サーバの **IP アドレス（あるいはサイト名）**と**サーバソケット**の**ポート番号**を指定して、通信を開始する

インターネットでの通信



14-5. Java でのソケット通信

さまざまなコンピュータ
（要求する）

サーバの **IPアドレス（またはサイト名）**と**サーバソケット**の**ポート番号**を指定して
通信を開始

データの送受信

サーバ
（要求を待ち続ける）

プログラムを**サーバソケット**に**バインド**

接続要求が来るのを待つ
（**accept**という）

データの送受信

サーバのプログラムの例

プログラムを、ポート番号 10001 にバインド．接続要求が来るのを待って、データの送受信を行う

```
ServerSocker sock;
Socker s;
BufferedReadrer reader;
sock = new ServerSocket();
sock.bind(new InetSocketAddress(“＜サイト名＞", 10001));

s = sock.accept();
reader = new BufferedReader(new InputStreamReader
                                    (s.getInputStream()));
String line = reader.readLine();
```

サーバソケットとデータ送受信用のソケット


```
ServerSocker sock;
Socker s;
BufferedReadrer reader;
sock = new ServerSocket();
sock.bind(new InetSocketAddress("<ホスト名>", 10001));

s = sock.accept();
reader = new BufferedReader(new InputStreamReader
                            (s.getInputStream()));
String line = reader.readLine();
```

サーバソケット

データ送受信用のソケット
※データ送受信に使うソケット．
「サーバソケット」とは別のもの

サーバとの通信のプログラムの例

サーバの IP アドレス（またはサイト名）と、ポート番号 10001 を指定して通信を開始。

```
Socket s = new Socket("＜サーバのサイト名＞", 10001);

BufferedWriter out = new BufferedWriter
                (new OutputStreamWriter(s.getOutputStream()));
out.write("GET /index.html HTTP/1.0¥r¥n");
out.flush();
```

演習

資料：**40 ～ 45**

【トピックス】

・ソケット通信

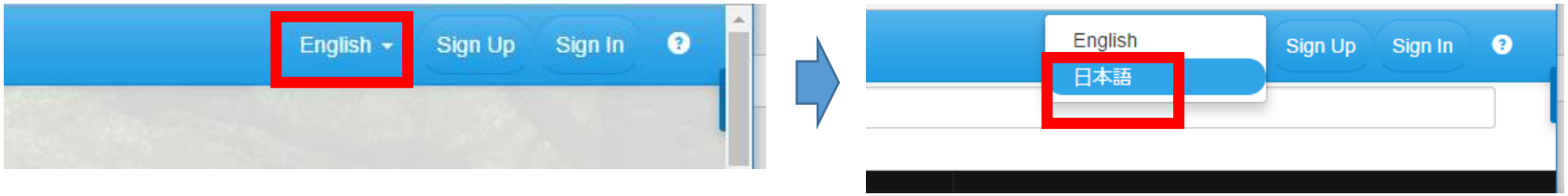
演習

① ウェブブラウザを起動する

② 次の URL を開く

https://paiza.io/

③ もし，表示が英語になっていたら，**日本語**に切り替える



④「コード作成を試してみる」をクリック

⑤「**Java**」を選ぶ（**左上のボタンをクリック**するとメニューが出る）

プログラムの編集画面

プログラムを**書き換えることができる**

実行ボタン

編集画面を確認する。
すでに、Java **プログラムが入っている.**
書き換えて使う.

```
1  import java.util.*;
2
3  public class Main {
4      public static void main(String[] args) throws Exception {
5          // Your code here!
6
7          System.out.println("XXXXXXXX");
8      }
9  }
10
```

① 次のソースコードを入れる

```java
import java.util.*;
import java.net.*;
import java.io.*;

public class Main {
  public static void main(String[] args) throws Exception {
    try {
      Socket s = new Socket("www.kkaneko.jp", 80);
      BufferedReader in = new BufferedReader
        (new InputStreamReader(s.getInputStream()));
      BufferedWriter out = new BufferedWriter
        (new OutputStreamWriter(s.getOutputStream()));
      out.write("GET /index.html HTTP/1.0\r\n");
      out.write("Host: www.kkaneko.jp:80\r\n");
      out.write("\r\n");
      out.flush();
      System.out.println(in.readLine());
      System.out.println(in.readLine());
      System.out.println(in.readLine());
      System.out.println(in.readLine());
    } catch(Exception e) {
      e.printStackTrace();
    }
  }
}
```

② 実行をクリック．実行結果を確認

出力 入力 コメント 0

```
HTTP/1.1 301 Moved Permanently
Date: Wed, 15 Dec 2021 00:39:44 GMT
Server: Apache/2.4.41 (Ubuntu)
Location: https://www.kkaneko.jp/index.html
```

まとめ

- **ソケット**の機能により，**インターネットでの通信**ができる

- **サーバ側**では，**プログラム**を，**サーバソケット**を**バインド**する.

- **サーバとの通信**では，**サーバ**の **IP アドレス**（またはサイト名）と、**ポート番号**を指定して通信を開始する.

関連ページ

- **Java プログラミング入門**

GDB online を使用

https://www.kkaneko.jp/pro/ji/index.html

- **Java の基本**

Java Tutor, GDB online を使用

https://www.kkaneko.jp/pro/pi/index.html

- **Java プログラム例**

https://www.kkaneko.jp/pro/java/index.html

14-2


```
import java.util.*;


public class Main {
    public static void main(String[] args) {
        System.out.println("start");
        try {
            Thread.sleep(100000);
        } catch(InterruptedException e) {}
    }
}
```

14-2


```
import java.util.*;

class MyTask extends TimerTask {
    public void run() {
        System.out.println("hello");
    }
}

public class Main
{
    public static void main(String[] args) {
        Timer timer = new Timer();
        timer.schedule(new MyTask(), 100, 500);
        System.out.println("start");
        try {
            Thread.sleep(100000);
        } catch(InterruptedException e) {}
    }
}
```

14-3


```
class Ball {
  double x;
  double y;
  public Ball(double x, double y) {
    this.x = x;
    this.y = y;
  }
  public void move() {
     this.x = this.x + 2;
     this.y = this.y + 3;
  }
  public void printout() {
     System.out.printf("%f %f¥n", this.x, this.y);
  }
}

public class Main
{
  public static void main(String[] args) {
    Ball b = new Ball(0, 0);
    b.printout();
    b.move();
    b.printout();
  }
}
```

14-3


```
import java.util.*;

class Ball {
  double x;
  double y;
  public Ball(double x, double y) {
    this.x = x;
    this.y = y;
  }
  public void move() {
     this.x = this.x + 2;
     this.y = this.y + 3;
  }
  public void printout() {
     System.out.printf("%f %f¥n", this.x, this.y);
  }
}

class MyTask extends TimerTask {
    Ball b;
    public MyTask() {
        super();
        this.b = new Ball(0, 0);
    }
    public void run() {
        b.move();
        b.printout();
    }
}

public class Main
{
  public static void main(String[] args) {
    Timer timer = new Timer();
    timer.schedule(new MyTask(), 100, 500);
    System.out.println("start");
    try {
      Thread.sleep(100000);
    } catch(InterruptedException e) {}
  }
}
```

14-5


```
import java.util.*;
import java.net.*;
import java.io.*;

public class Main {
  public static void main(String[] args) throws Exception {
    try {
      Socket s = new Socket("www.kkaneko.jp", 80);
      BufferedReader in = new BufferedReader
        (new InputStreamReader(s.getInputStream()));
      BufferedWriter out = new BufferedWriter
        (new OutputStreamWriter(s.getOutputStream()));
      out.write("GET /index.html HTTP/1.0¥r¥n");
      out.write("Host: www.kkaneko.jp:80¥r¥n");
      out.write("¥r¥n");
      out.flush();
      System.out.println(in.readLine());
      System.out.println(in.readLine());
      System.out.println(in.readLine());
      System.out.println(in.readLine());
    } catch(Exception e) {
      e.printStackTrace();
    }
  }
}
```