

pi-12. Java標準ライブラリによる時間制御と並行処理

トピック：Java の標準ライブラリ，時間，スリープ，疑似乱数，マルチスレッド，タイマー

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



- **Java** で**標準ライブラリ**には、次のような機能がある
 - スレッド
 - 時間, タイマー
 - 疑似乱数
- **Java** で**標準ライブラリ**の機能を使うとき, **標準ライブラリ**の**クラス**を**継承**して使う場合がある

番号	項目
	復習
12-1	Java の標準ライブラリ
12-2	時間、スリープ
12-3	スリープに関する演習
12-4	疑似乱数
12-5	マルチスレッド

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



```
Main.java
1 public class Main
2 {
3     public static void main(String[] args) {
4         int x = 100;
5         int y = 200;
6         System.out.printf("%d\n", x + y);
7     }
8 }
```

input

```
300
...Program finished with exit code 0
Press ENTER to exit console.
```

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>

A screenshot of a search bar with a magnifying glass icon on the left. The text "https://www.onlinegdb.com" is entered into the search field. The search bar is set against a light gray background.

🔍 <https://www.onlinegdb.com>

③ 「Language」 のところで, 「Java」 を選ぶ

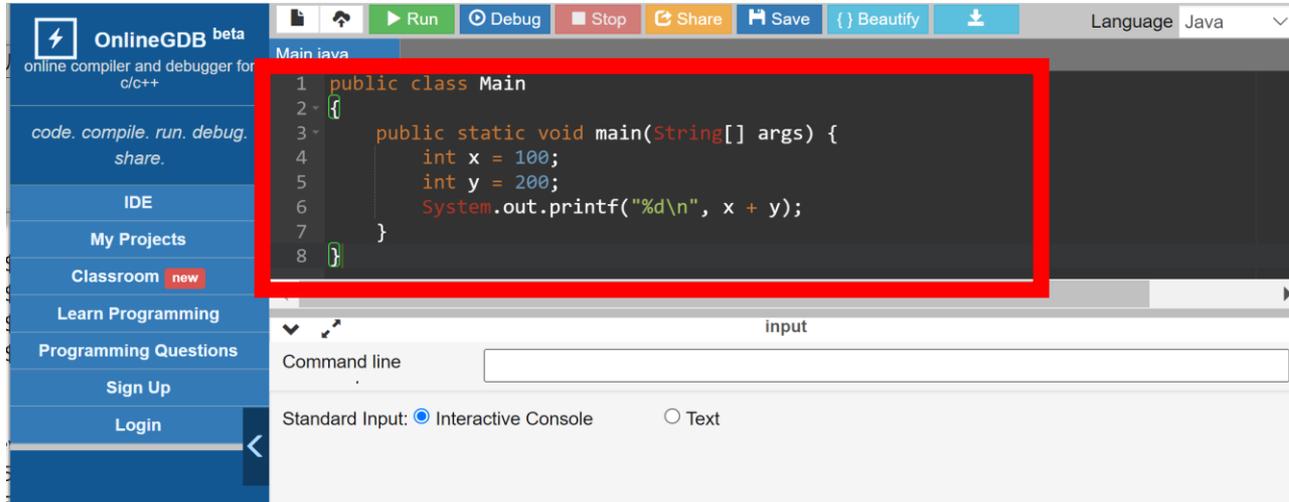
SPONSOR Slack — Bring your team together with Slack, the collaboration hub for work.

Run Debug Stop Share Save { } Beautify Language -- select --

```
1 /*****  
2  
3 Welcome to GDB Online.  
4 GDB online is an online compiler and debugger tool for C, C++, Python  
5 C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS  
6 Code, Compile, Run and Debug online from anywhere in world.  
7  
8 *****/  
9 #include <stdio.h>  
10  
11 int main()  
12 {  
13     printf("Hello World");  
14  
15     return 0;  
16 }  
17
```

Language dropdown menu options: -- select --, C, C++, C++ 14, C++ 17, **Java**, Python 3, PHP, C#, VB, HTML,JS,CSS, Ruby, Perl, Pascal, R, Fortran, Haskell, Assembly(GCC), Objective C, SQLite

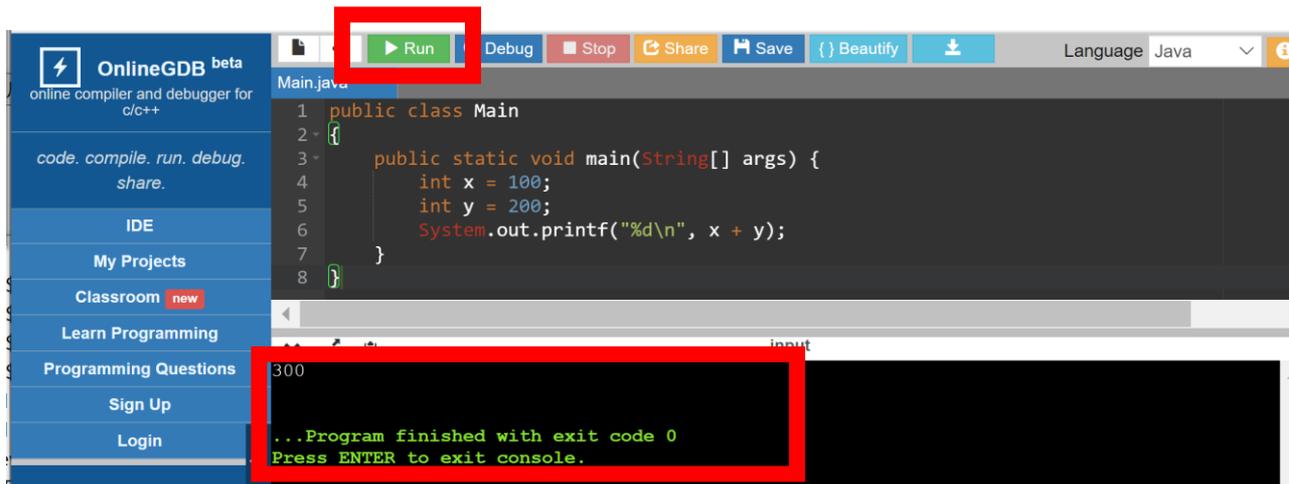
④ ソースコードを入れる



```
1 public class Main
2 {
3     public static void main(String[] args) {
4         int x = 100;
5         int y = 200;
6         System.out.printf("%d\n", x + y);
7     }
8 }
```

⑤ 実行. 実行結果を確認

「Run」をクリック.



```
300
...Program finished with exit code 0
Press ENTER to exit console.
```

12-1. Java の標準ライブラリ

ライブラリ

- 複数のプログラムが共有して使えるような機能を持ったプログラムのこと
- プログラミング言語処理系に元から備わっている**ライブラリ**のことを、**標準ライブラリ**という

Java の標準ライブラリの機能



- コレクション (ArrayList, HashMap など)
- ラップクラス (Integer, Double など)
- 文字列 (String)
- その他
 - 時間, スリープ
 - 疑似乱数
 - スレッド
 - 標準入出力
 - 数値処理
 - ファイル読み書き など

12-2. 時間, スリープ

時間, スリープ



- いまの日時（日付、時刻）を知る
- 日時に関する計算
- 処理を一定時間止める（タイミング、一定時間ごとの監視など）

Java での時間, スリープ



- **いまの日時（日付、時刻）を知る**

`java.time.LocalDateTime.now()`

- **日時に関する計算**

`plusHours(1)` 1時間後

`plusMinutes(1)` 1分後

`plusSeconds(1)` 1秒後

- **処理を一定時間止める**（タイミング, 一定時間ごとの監視など）
- `Thread.sleep(2000)` 2秒止まる.

「**2000**」とあるのはミリ秒単位

演習

資料 : 15 ~ 17

【トピックス】

- 日時に関する計算
- 処理を一定時間止める

日時に関する計算



① 次のソースコードを入れる

```
1 public class Main
2 {
3     public static void main(String[] args) {
4         java.time.LocalDateTime d;
5         d = java.time.LocalDateTime.now();
6         System.out.println(d);
7     }
8 }
```

② 実行結果の確認

A screenshot of a terminal window with a black background and green text. At the top, there are three small icons: a downward arrow, a cursor, and a trash can. The main text in the terminal reads: "2021-12-02T01:44:33.506569", followed by "...Program finished with exit code 0" and "Press ENTER to exit console." with a small white cursor box at the end.

```
2021-12-02T01:44:33.506569

...Program finished with exit code 0
Press ENTER to exit console.█
```

世界標準時が表示される。9時間ずれている。

日時に関する計算



① 次のソースコードを入れる

```
1 public class Main
2 {
3     public static void main(String[] args) {
4         java.time.LocalDateTime d;
5         d = java.time.LocalDateTime.now();
6         System.out.println(d);
7         System.out.println(d.plusHours(1));
8         System.out.println(d.plusMinutes(1));
9         System.out.println(d.plusSeconds(1));
10    }
11 }
```

② 実行結果の確認

```
2021-12-02T01:45:42.360611
2021-12-02T02:45:42.360611
2021-12-02T01:46:42.360611
2021-12-02T01:45:43.360611

...Program finished with exit code 0
Press ENTER to exit console.
```

処理を一定時間止める



① 次のソースコードを入れる

```
1 public class Main
2 {
3     public static void main(String[] args) {
4         java.time.LocalDateTime d;
5         d = java.time.LocalDateTime.now();
6         System.out.println(d);
7         try {
8             Thread.sleep(2000);
9         } catch (InterruptedException e) {}
10        d = java.time.LocalDateTime.now();
11        System.out.println(d);
12    }
13 }
```

② 実行結果の確認

```
2021-12-02T01:46:40.882097
2021-12-02T01:46:42.882626
...Program finished with exit code 0
Press ENTER to exit console.
```

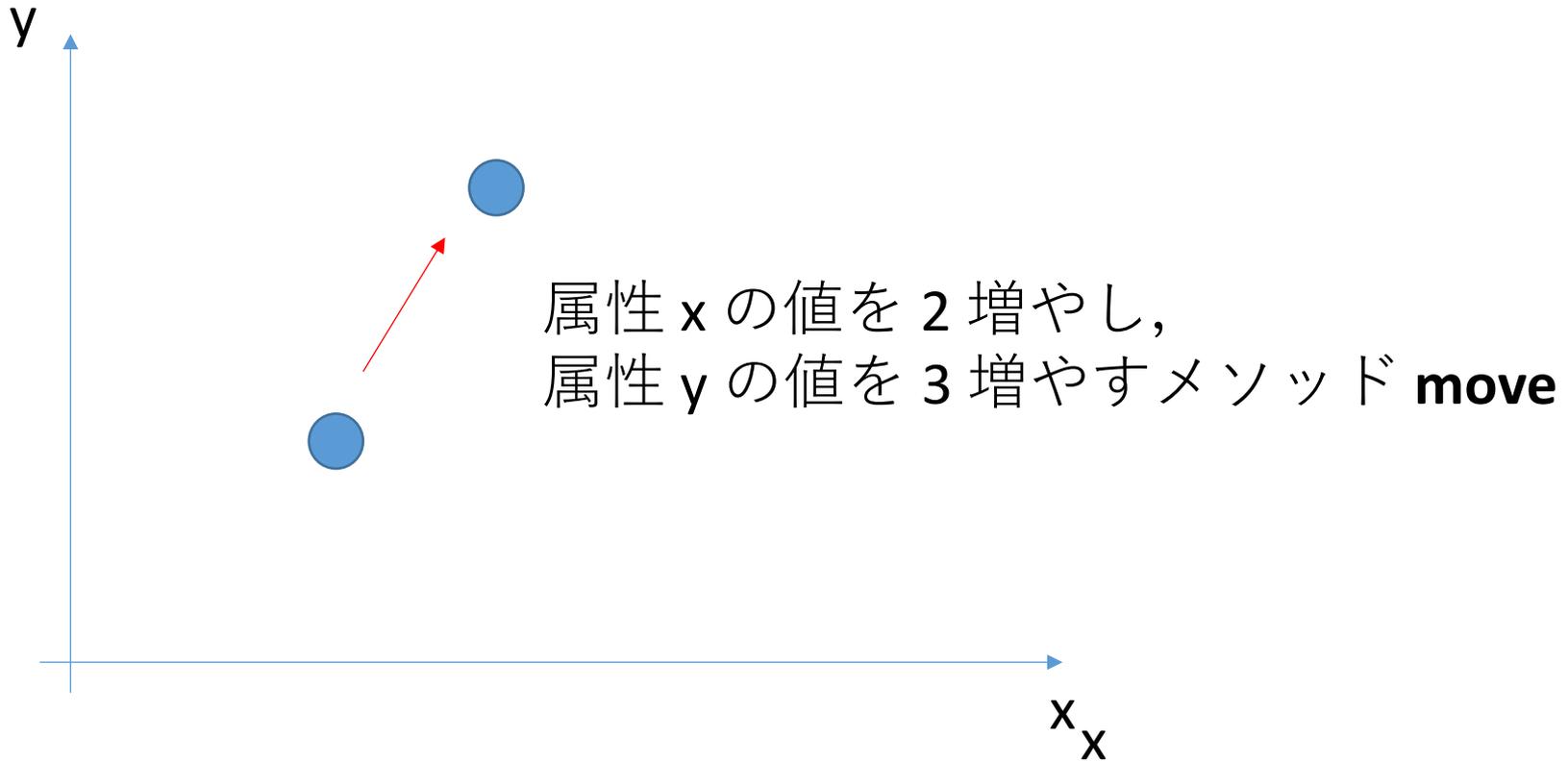
2秒止まる

「2000」

とあるのは
ミリ秒単位

12-3. スリープに関する演習

Ball クラスのオブジェクト



演習

資料 : 21 ~ 25

【トピックス】
・スリープ

① 次のソースコードを入れる



```
1 class Ball {
2     double x;
3     double y;
4     public Ball(double x, double y) {
5         this.x = x;
6         this.y = y;
7     }
8     public void move() {
9         this.x = this.x + 2;
10        this.y = this.y + 3;
11    }
12    public void printout() {
13        System.out.printf("%f %f\n", this.x, this.y);
14    }
15 };
```

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
```

動く前は0と0. 動いた後は2と3.

- 10回動かす
for 文, メソッド move
- 動かす前に1秒止まる
Thread.sleep(1000)

③ public class Main の中を次のように書き換える



```
17 public class Main
18 {
19     public static void main(String[] args) {
20         Ball b = new Ball(0, 0);
21         b.printout();
22         for(int i = 1; i <= 10; i++) {
23             try {
24                 Thread.sleep(1000);
25             } catch(InterruptedException e) {}
26             b.move();
27             b.printout();
28         }
29     }
30 }
```

④ 実行結果の確認

```
0.000000 0.000000
2.000000 3.000000
4.000000 6.000000
6.000000 9.000000
8.000000 12.000000
10.000000 15.000000
12.000000 18.000000
14.000000 21.000000
16.000000 24.000000
18.000000 27.000000
20.000000 30.000000
```

1つ表示のたびに、
1秒止まる

12-4. 疑似乱数

疑似乱数とは



- **疑似乱数は、コンピュータによって生成されたランダムな数や数の列**
- 「乱数」ということもある

疑似乱数

アルゴリズムにより生成される数である。再現が可能。その意味で「疑似」

演習

資料： 29

【トピックス】
・ 疑似乱数

① 次のソースコードを入れる



```
1 public class Main
2 {
3     public static void main(String[] args) {
4         int a;
5         java.util.Random r = new java.util.Random();
6         a = r.nextInt(10);
7         System.out.println(a);
8     }
9 }
```

nextInt(10) は、0から9までの数（10通り）の中から1つをランダムに得る

2

7

実行のたびに
結果は変わる

12-5. マルチスレッド

スレッド

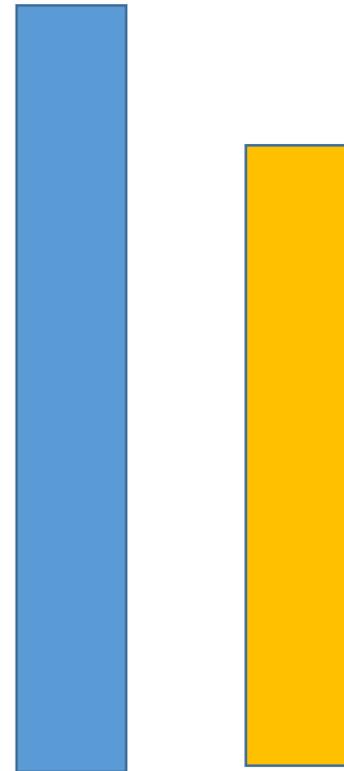


- **スレッド**とは、処理の流れのこと。
- 複数の処理を並行させたいときには、**マルチスレッド**（複数のスレッド）で処理を行う

通常のプログラム実行
(シングルスレッド)



マルチスレッド



ここではスレッド数は2.
(スレッド数は3以上にもできる) 32

演習

資料 : 34 ~ 41

【トピックス】

- ・ マルチスレッド

シングルスレッドの例

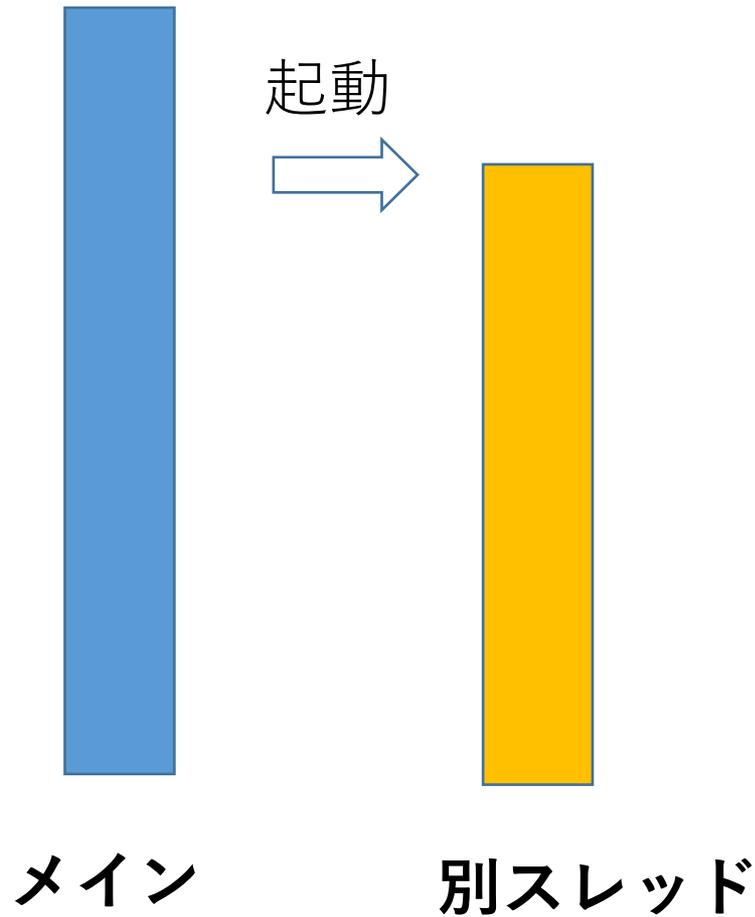


① 次のソースコードを入れる

```
1 public class Main
2 {
3     public static void main(String[] args) {
4         for(int i = 0; i < 100; i++) {
5             try {
6                 Thread.sleep(1000);
7             } catch(InterruptedException e) {}
8             System.out.println("Hello");
9         }
10    }
11 }
```

Hello を 100回表示. 表示のたびに 1秒止まる.

マルチスレッドの例



マルチスレッドの例



③ 前のソースコードは消して、次のソースコードを入れる

Java では標準ライブラリのクラス Thread のサブクラスを定義

```
1 class Morning extends Thread {
2     public void run() {
3         for(int i = 0; i < 50; i++) {
4             try {
5                 Thread.sleep(2000);
6             } catch(InterruptedException e) {}
7             System.out.println("Morning");
8         }
9     }
10 }
11
12 public class Main
13 {
14     public static void main(String[] args) {
15         Morning m = new Morning();
16         m.start();
17     }
18 }
```

run メソッドの中に、別スレッドでの処理を書く

オブジェクトを生成

新しいスレッドを起動

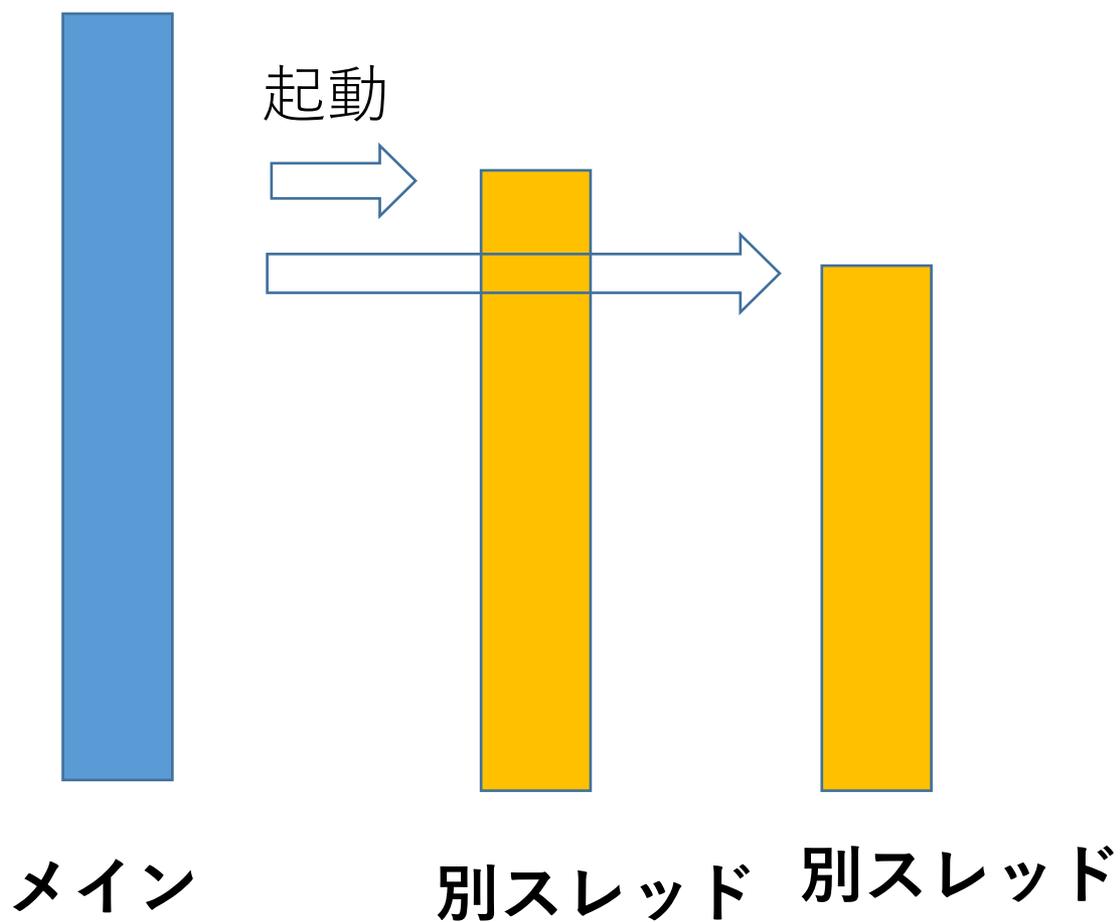
④ 実行結果の確認

A screenshot of a terminal window. The window has a dark background and a light-colored title bar. The title bar contains three icons: a downward-pointing chevron, a square with an upward-pointing chevron, and a document icon with an arrow. The main area of the terminal is black with white text. The text "Morning" is displayed on three separate lines. A small white cursor is visible at the end of the third line.

```
Morning
Morning
Morning
```

2秒ごとに Morning を表示

マルチスレッドの例



マルチスレッドの例



⑤ 前のソースコードに書き加える

```
1 class Morning extends Thread {
2     public void run() {
3         for(int i = 0; i < 50; i++) {
4             try {
5                 Thread.sleep(2000);
6             } catch(InterruptedException e) {}
7             System.out.println("Morning");
8         }
9     }
10 }
11
12 class Hello extends Thread {
13     public void run() {
14         for(int i = 0; i < 50; i++) {
15             try {
16                 Thread.sleep(3000);
17             } catch(InterruptedException e) {}
18             System.out.println("Hello");
19         }
20     }
21 }
22
23 public class Main
24 {
25     public static void main(String[] args) {
26         Morning m = new Morning();
27         m.start();
28         Hello h = new Hello();
29         h.start();
30     }
31 }
```

⑥ 実行結果の確認

```
Morning  
Hello  
Morning  
Hello  
Morning  
Morning  
Hello  
Morning
```

2秒ごとに Morning を表示

3秒ごとに Hello を表示

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>

```
class Student {
    String id;
    String name;
    String address;
    public Student(String id, String name, String address) {
        this.id = id;
        this.name = name;
        this.address = address;
    }
    public void printout() {
        System.out.printf("%s %s %s", this.id, this.name, this.address);
    }
};

public class Main
{
    public static void main(String[] args) {
        Student k = new Student("t001", "kaneko", "matsunaga");
        k.printout();
    }
}
```

12-2



```
public class Main
{
    public static void main(String[] args) {
        java.time.LocalDateTime d;
        d = java.time.LocalDateTime.now();
        System.out.println(d);
    }
}
```

```
-----
public class Main
{
    public static void main(String[] args) {
        java.time.LocalDateTime d;
        d = java.time.LocalDateTime.now();
        System.out.println(d);
        System.out.println(d.plusHours(1));
        System.out.println(d.plusMinutes(1));
        System.out.println(d.plusSeconds(1));
    }
}
```

```
-----
public class Main
{
    public static void main(String[] args) {
        java.time.LocalDateTime d;
        d = java.time.LocalDateTime.now();
        System.out.println(d);
        try {
            Thread.sleep(2000);
        }
        d = java.time.LocalDateTime.now();
        System.out.println(d);
    }
}
```

12-3



```
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);
        k.printout();
    }
}
```

```
-----
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();
        for(int i = 1; i <= 10; i++) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {}
            b.move();
            b.printout();
        }
    }
}
```

12-4



```
public class Main
{
    public static void main(String[] args) {
        int a;
        java.util.Random r = new java.util.Random();
        a = r.nextInt(10);
        System.out.println(a);
    }
}
```

12-5



```
class Morning extends Thread {  
    public void run() {  
        for(int i = 0; i < 50; i++) {  
            try {  
                Thread.sleep(2000);  
            } catch(InterruptedException e) {}  
            System.out.println("Morning");  
        }  
    }  
}
```

```
public class Main  
{  
    public static void main(String[] args) {  
        Morning m = new Morning();  
        m.start();  
    }  
}
```

12-5



```
class Morning extends Thread {
    public void run() {
        for(int i = 0; i < 50; i++) {
            try {
                Thread.sleep(2000);
            } catch (InterruptedException e) {}
            System.out.println("Morning");
        }
    }
}

class Hello extends Thread {
    public void run() {
        for(int i = 0; i < 50; i++) {
            try {
                Thread.sleep(3000);
            } catch (InterruptedException e) {}
            System.out.println("Hello");
        }
    }
}

public class Main
{
    public static void main(String[] args) {
        Morning m = new Morning();
        m.start();
        Hello h = new Hello();
        h.start();
    }
}
```