

# pi-16. プログラムのテ スト, アサーション, 例外処理

トピックス: プログラムの設計レシピ, 種々の工  
ラー, プログラムのテスト, アサーション, 例外  
処理

URL: <https://www.kkaneko.jp/cc/pi/index.html>

(Java の基本)

金子邦彦



# 今回の内容



- ・プログラム設計レシピでは、プログラムの作成を、①分析、②仕様、③例、④作成、⑤テストの5段階に分ける
- ・「必ず成立しているはずのこと」を書いたものがアサーションである。Javaでは次のように書く。アサーションは論理的エラーの発見に役立つ

```
assert x >= 0 : "error, x < 0 in area();"
```

- ・「`x >= 0`」がアサーション、
- ・`error, x < 0 in area()` がメッセージ
- ・例外処理とは、不測の事態における処理

# アウトライン



番号	項目
	復習
16-1	プログラムの設計レシピ
16-2	種々のエラーとプログラムのテスト
16-3	アサーション
16-4	例外処理

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



Run

Debug

Stop

Share

Save

{ } Beautify



Database Lab.

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

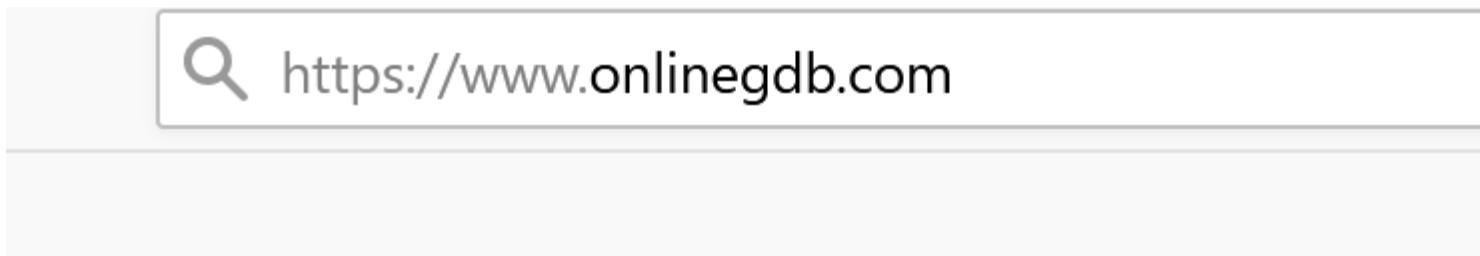
# GDB online で Java を動かす手順



① ウェブブラウザを起動する

② 次の URL を開く

**https://www.onlinegdb.com**





### ③ 「Language」 のところで、「Java」 を選ぶ

The screenshot shows the GDB Online interface. At the top, there's a 'SPONSOR' banner for Slack. Below it is a toolbar with buttons for Run, Debug, Stop, Share, Save, Beautify, and download. To the right of the toolbar is a 'Language' dropdown menu with a red border around it. A cursor is hovering over the 'Java' option in the dropdown menu, which is also highlighted with a red border. The main area shows a 'source code' tab with a 'Hello World' program in C:

```
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
```

The dropdown menu lists the following languages:

- 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



The screenshot shows the OnlineGDB beta interface. On the left, there's a sidebar with links like 'IDE', 'My Projects', 'Classroom new', 'Learn Programming', 'Programming Questions', 'Sign Up', and 'Login'. The main area has tabs for 'Main.java' and 'Main.c'. Below the tabs are buttons for 'Run', 'Debug', 'Stop', 'Share', 'Save', 'Beautify', and a download icon. The language is set to 'Java'. The code editor contains the following Java code:

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

Below the code editor, there's an 'input' field and a 'Command line' field. A radio button labeled 'Interactive Console' is selected under 'Standard Input'. The output window is currently empty.

# ⑤ 実行. 実行結果を確認

「Run」をクリック.

The screenshot shows the OnlineGDB beta interface after clicking the 'Run' button. The 'Run' button is highlighted with a red box. The output window at the bottom shows the execution results:

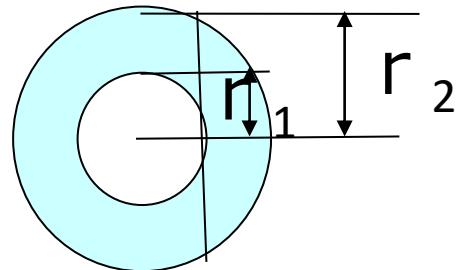
```
300
...Program finished with exit code 0
Press ENTER to exit console.
```

# 16-1. プログラムの設計レシピ

# リングの面積



- ・ リング **Ring** クラスのオブジェクト  
属性は、外径 **r2**, 内径 **r1**
- ・ 面積 **area** メソッド  
$$(\text{this.r2} * \text{this.r2} - \text{this.r1} * \text{this.r1}) * 3.14$$



# 演習

資料：11～12

## 【トピックス】

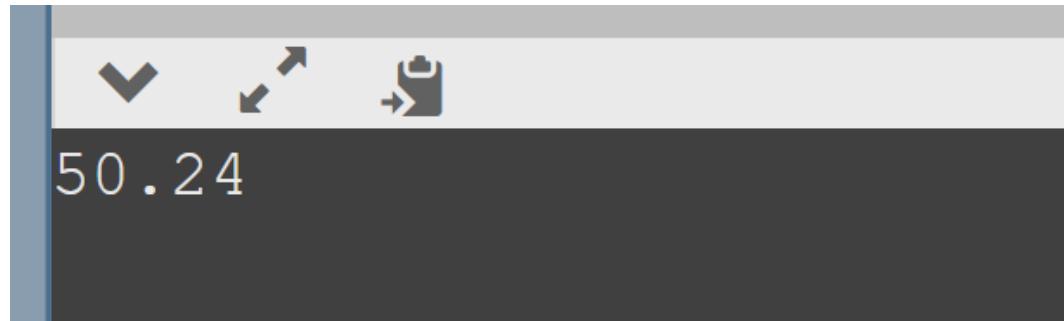
- ・プログラムの設計レシピ

# ① プログラム



```
1 class Ring {
2     double r1;
3     double r2;
4     public Ring(double r1, double r2) {
5         this.r1 = r1;
6         this.r2 = r2;
7     }
8     public double area() {
9         double x;
10        x = (this.r2 * this.r2 - this.r1 * this.r1) * 3.14;
11        return x;
12    }
13 }
14 public class Main
15 {
16     public static void main(String[] args) {
17         Ring a = new Ring(3, 5);
18         System.out.println(a.area());
19     }
20 }
```

## ② 実行し結果を確認



# プログラム設計レシピ



**プログラムを設計する**ときに、考えておいた方がよいこと、実行した方がよいことをリストアップしたもの

# プログラム設計レシピ



① 分析

② 仕様

③ 例

④ テスト



プログラムの設計段階で考えた方がよいこと

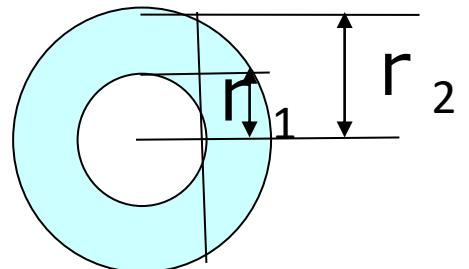


プログラムが設計通りに動くかを確かめる

# リングの面積



- ・ リング **Ring** クラスのオブジェクト  
属性は、外径  $r_2$ , 内径  $r_1$
- ・ 面積 **area** メソッド



# ① 分析



メソッド area

- ・その入力 なし                   ・・・オブジェクトの属性  
  だけで足りる
- ・その出力 数値 1つ            ・・・求まった面積

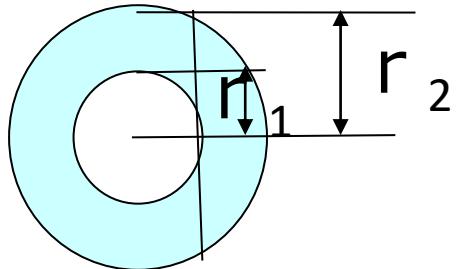
※ 必要なメソッドそれぞれについて、入力と出力を  
考える

## ② 仕様



メソッド area

r1 を内径, r2 を外径とするリングの面積



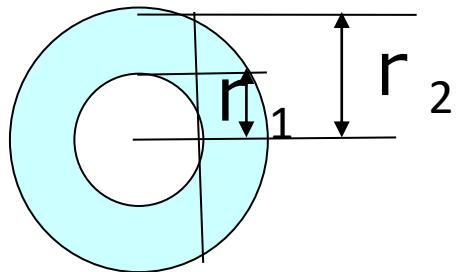
メソッドの性能, 機能の条件を考える

### ③ 例



メソッド area

$r1 = 3, r2 = 5$  のとき, 50.24 が求まる



どのようなときに、どのような出力が出てほしいか

# 「③ 例」があることの効果



## ③ 例

r1 = 3, r2 = 5 のとき, 50.24 が求まる

- ・例があれば、プログラムが書きやすくなる
- ・例があれば、他の人がプログラムを見たとき、理解しやすくなる
- ・例は、テストの実施に利用

# ④ テスト



## ③ 例

r1 = 3, r2 = 5 のとき, 50.24 が求まる

```
1 class Ring {  
2     double r1;  
3     double r2;  
4     public Ring(double r1, double r2) {  
5         this.r1 = r1;  
6         this.r2 = r2;  
7     }  
8     public double area() {  
9         double x;  
10        x = (this.r2 * this.r2 - this.r1 * this.r1) * 3.14;  
11        return x;  
12    }  
13 }  
14 public class Main  
15 {  
16     public static void main(String[] args) {  
17         Ring a = new Ring(3, 5);  
18         System.out.println(a.area());  
19     }  
20 }  
21
```

この通りの結果が得られることを  
確かめる

Ring a = new Ring(3, 5);

50.24

input

# プログラム設計レシピ



- ① **分析** 名前付け，入力と出力を明らかに
- ② **仕様** 性能機能の条件
- ③ **例** どのようなときに，どのような出力か
- ④ **テスト** プログラムが設計通りに動くか確認

# プログラム設計レシピ



① 分析 名前付け，入力と出力を明らかに

名前：area，入力：なし，出力：数値1つ

② 仕様 性能機能の条件

r1 を内径，r2 を外径とするリングの面積

③ 例 どのようなときに，どのような出力か

r1 = 3, r2 = 5 のとき，50.24 が求まる

④ テスト プログラムが設計通りに動くか

```
Ring a = new Ring(3, 5);
System.out.println(a.area());
```

テスト用プログラム

# 16-2. 種々のエラーとプログラム のテスト

## ① 構文エラー(Syntax Errors)

- ・言語の書式に合っていない場合

## ② 実行エラー(Run-time Errors)

- ・プログラムが出力を出さない場合

## ③ 論理的エラー(Logical Errors)

- ・出てきた出力が、仕様（性能機能の条件）に一致しない場合

# ① 構文エラー



Compile error

```
1 import java.util.*;
2
3 interface Zukei {
4     public double area() throws Exception;
5 }
6 class Ring implements Zukei {
7     double r1;
8     double r2;
9     public Ring(double r1, double r2) {
10         this.r1 = r1;
11         this.r2 = r2;
12     }
13     public double area() throws Exception {
14         double x = this.r2 * this.r2 * 3.14 - this.r1 * this.r1 * 3.14;
15         return x;
16     }
17 }
18 public class Main {
19     public static void main(String[] args) throws Exception {
20         Ring a = new Ring(3, 5);
21         System.out.println(a.area());
22     }
23 }
```

スペルミスで、  
プログラムが  
実行できない

➡ 実行 (Ctrl-Enter)

コンパイルエラー 入力 コメント 0

```
Main.java:9: error: invalid method declaration; return type required
    public Ring(double r1, double r2) {
        ^
1 error
```

# 実行エラーを含むプログラム



```
class Ring {  
    double r1;  
    double r2;  
    public Ring(double r1, double r2) {  
        this.r1 = r1;  
        this.r2 = r2;  
    }  
    public double area() {  
        double x;  
        x = (this.r2 * this.r2 - this.r1 * this.r1) / 0;  
        return x;  
    }  
}  
public class Main  
{  
    public static void main(String[] args) {  
        Ring a = new Ring(3, 5);  
        System.out.println(a.area());  
    }  
}
```

### ③ 論理的エラー



```
1 import java.util.*;
2
3 interface Zukei {
4     public double area() throws Exception;
5 }
6 class Ring implements Zukei {
7     double r1;
8     double r2;
9     public Ring(double r1, double r2) {
10         this.r1 = r1;
11         this.r2 = r2;
12     }
13     public double area() throws Exception {
14         double x = this.r2 * this.r2 * 3.14 - this.r1 * this.r1
15         return x;
16     }
17 }
18 public class Main {
19     public static void main(String[] args) throws Exception {
20         Ring a = new Ring(3, 5);
21         System.out.println(a.area());
22     }
23 }
```

2.14;

スペルミスで、  
おかしな答え  
が出ている

実行 (Ctrl-Enter)

出力 入力 コメント 0

59.239999999999995

# エラーの種類ごとに発見方法が異なる



## ① 構文エラー(Syntax Errors)

- ・言語の書式に合っていない場合

プログラムを実行させようとするとき、実行できず、エラーメッセージが出る

## ② 実行エラー(Run-time Errors)

- ・プログラムが出力を出さない場合

プログラムを実際に実行させる。そしてエラーメッセージが出たり、プログラムが「止まっている」ように見えたりする

## ③ 論理的エラー(Logical Errors)

- ・出てきた出力が、仕様に合致しない場合

プログラムを実際に実行させる。そしてテストする

# 分野の知識(Domain Knowledge)



- ・プログラムを定義するには、プログラムをしようとする領域の知識（分野の知識）が必要。
  - ・音楽，交通工学，生物学，芸術・・・など  
　　プログラムしたい領域の用語，知識の理解が必要
- ・「**対象分野に応じてプログラム言語 자체を発明する，設計する**」という考え方（興味のある人は「Domain Specific Language」という言葉で調べてみること）
- ・コンピュータ言語のみにとらわれず，**対象分野のしっかりとした基礎の理解**が不可欠。

## 16-3. アサーション

# アサーション



- 「必ず成立しているはずのこと」を書いたものがアサーションである。Javaでは次のように書く。  
アサーションは論理的エラーの発見に役立つ

```
assert x >= 0 : "error, x < 0 in area();"
```

- 「 $x \geq 0$ 」がアサーション、
- $\text{error, } x < 0 \text{ in area()}$  がメッセージ

# アサーションをエラーの発見に役立てる



```
import java.util.*;  
  
interface Zukei {  
    public double area();  
}  
  
class Ring implements Zukei {  
    double r1;  
    double r2;  
    public Ring(double r1, double r2) {  
        this.r1 = r1;  
        this.r2 = r2;  
    }  
    public double area() {  
        double x = this.r1 * this.r1 * 3.14 - this.r2 * this.r2 * 3.14;  
        return x;  
    }  
}  
public class Main {  
    public static void main(String[] args) throws Exception {  
        Ring a = new Ring(3, 5);  
        System.out.println(a.area());  
    }  
}
```

```
D:\$>java Main  
-50. 2399999999999995
```

間違い（エラー）を含む式

実行結果の例

アサーションがない場合、  
論理的エラーを見逃しそう

# アサーションをエラーの発見に役立てる



```
import java.util.*;  
  
interface Zukei {  
    public double area();  
}  
  
class Ring implements Zukei {  
    double r1;  
    double r2;  
    public Ring(double r1, double r2) {  
        this.r1 = r1;  
        this.r2 = r2;  
    }  
    public double area() {  
        double x = this.r1 * this.r1 * 3.14 - this.r2 * this.r2 * 3.14;  
        assert x >= 0 : "error, x < 0 in area()";  
        return x;  
    }  
}  
public class Main {  
    public static void main(String[] args) throws Exception {  
        Ring a = new Ring(3, 5);  
        System.out.println(a.area());  
    }  
}
```

**assert x >= 0 : "error, x < 0 in area()";**

```
D:¥>java -ea Main  
Exception in thread "main" java.lang.AssertionError: error, x < 0 in area()  
at Ring.area(Main.java:16)  
at Main.main(Main.java:23)  
D:¥>
```

アサーションの機能により  
エラーメッセージが出る

実行結果の例

- ・アサーションの機能は Java Tutor, GDB online, Paiza.io で動かすのが難しいので、**前ページの実演のみ**（演習なし）



## 16-4. 例外處理

# 例外処理



- **例外処理**とは、不測の事態における処理

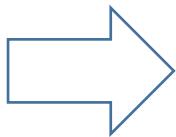
不測の事態（例外）の例

- まさか「ファイルが無い」とは
- まさか「プログラム中にバグがある」とは
- まさか「半径が負の数に設定されている」とは
- まさか「アサーションが成立しない」とは

例外処理の例

- 「ファイルを準備するように」とメッセージを出して処理を**再開**
- 「正しい値を設定するように」とメッセージを出して処理を**終了**

例外の  
発生



例外処理

# Java での例外処理



- throws Exception

メソッド内で例外が発生する可能性があること

- throw new Exception

## 例外の発生

- try { ... } catch( ... ) { ... }

## 例外処理



```
class Ring implements Zukei {  
    double r1;  
    double r2;  
    public Ring(double r1, double r2) {  
        this.r1 = r1;  
        this.r2 = r2;  
    }  
    public double area() throws Exception {  
        if (r2 < r1) {  
            throw new Exception("r2 < r1");  
        }  
        double x = this.r2 * this.r2 * 3.14 - this.r1 * this.r1 * 3.14;  
        assert x >= 0 : "error, x < 0 in area()";  
        return x;  
    }  
}  
public class Main {  
    public static void main(String[] args) throws Exception {  
        Ring a = new Ring(3, 5);  
        try {  
            System.out.println(a.area());  
        } catch(Exception e) {  
            System.out.println("a.area() failed");  
            return;  
        }  
    }  
}
```

### throws Exception

「area メソッド内で、例外の発生がある」と宣言（コンピュータに教える）

r2 < r1 は想定外なので、例外を発生させる

例外処理

# Java での例外処理の書き方



```
try {  
    处理  
} catch(Exception e) {  
    例外処理  
}
```

<処理>で例外が発生したときの  
例外処理ができるようになる  
※ 「Exception」のところは,  
例外の種類を細かく指定することも可能

# 16-1



```
class Ring {  
    double r1;  
    double r2;  
    public Ring(double r1, double r2) {  
        this.r1 = r1;  
        this.r2 = r2;  
    }  
    public double area() {  
        double x;  
        x = (this.r2 * this.r2 - this.r1 * this.r1) * 3.14;  
        return x;  
    }  
}  
public class Main  
{  
    public static void main(String[] args) {  
        Ring a = new Ring(3, 5);  
        System.out.println(a.area());  
    }  
}
```

# 16-2 実行エラーを含むプログラム



```
class Ring {  
    double r1;  
    double r2;  
    public Ring(double r1, double r2) {  
        this.r1 = r1;  
        this.r2 = r2;  
    }  
    public double area() {  
        double x;  
        x = (this.r2 * this.r2 - this.r1 * this.r1) / 0;  
        return x;  
    }  
}  
public class Main  
{  
    public static void main(String[] args) {  
        Ring a = new Ring(3, 5);  
        System.out.println(a.area());  
    }  
}
```

# 関連ページ

- Java プログラミング入門

GDB online を使用

<https://www.kkaneko.jp/cc/ji/index.html>

- Java の基本

Java Tutor, GDB online を使用

<https://www.kkaneko.jp/cc/pi/index.html>

- Java プログラム例

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

