



Java

練習問題

1	概要	3
2	標準出力	5
3	基本構文	8
4	リテラル	11
5	変数	13
6	入出力	17
7	式と演算子	21
8	配列	29
9	分岐処理	34
10	繰り返し	44
11	デバッグ①	55
12	クラスの基本	56
13	引数	58
14	戻り値	60
15	アクセス制限	63
16	オーバーロード	65
17	コンストラクタ	66
18	Static メンバ	67
19	クラスライブラリ	68
20	クラス型の変数	76
21	継承	78
22	オーバーライド	81
23	Object クラス	82
24	抽象クラス	86
25	インターフェイス	88
26	パッケージ	93

27	インポート	94
28	例外	96
29	コレクションフレームワーク	100
30	デバッグ②	105
31	コーディング規約	109

1 概要

Question1_1

Cドライブ配下に、新規フォルダを作成し、Java ファイルを作成し、
コマンドプロンプトでコンパイル及びプログラムの実行をしましょう。
また実行時にコンソール上に実行結果を表示するコードを記述すること。
記述にあたっては以下の仕様に従うものとします。

フォルダ名...java_exercises

クラス名...Question01_01

実行結果

こんにちは

Question1_2

Eclipse で Java ファイルを作成し、実行結果を表示するコードを記述しましょう。

記述にあたっては以下の仕様に従うものとします。

プロジェクト名...java_training

パッケージ名...question01

クラス名...Question01_02

実行結果

こんばんは

2 標準出力

Question2_1

System.out.println()を使用して、実行結果を表示するコードを記述しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question02

クラス名...Question02_01

実行結果

出力の練習

1 回目

Question2_2

System.out.print()を使用して、「出力の練習」と「2回目」という文言を実行結果の通りに表示するコードを記述しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question02

クラス名...Question02_02

実行結果

出力の練習 2 回目

Question2_3

System.out.print()と System.out.println()を使用して、「出力の練習」「3回目」「出力を終了します」という文言を実行結果の通りに表示するコードを記述しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question02

クラス名...Question02_03

実行結果

出力の練習 3 回目

出力を終了します

3 基本構文

Question3_1

下記プログラムはコンパイルエラーが発生しています。

原因を特定し、正しく動作するようにプログラムを修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question03

クラス名...Question03_01

```
package question03;

public class Question03_01
    public static void main(String[] args) {
        System.out.println("ようこそ Java へ");
    }
```

Question3_2

下記プログラムは正常に動作しているものの、インデントや改行がされておられません。
正しくインデント、スペース、改行を挿入し、ソースコードを読みやすく修正しましょう。
記述にあたっては以下の仕様に従うものとします。

パッケージ名...question03

クラス名...Question03_02

```
package question03;

public class Question03_02 {public static void main(String[]
args) {
System.out.println("赤パジャマ");System.out.println("青パジャマ
");System.out.println("黄パジャマ");}}
```

Question3_3

下記プログラムは正常に動作しているものの、コメントが記述されておりません。

「System.out.print("表示を")」の直前に 1 行コメントを記述しましょう。

コメントの内容は任意とします。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question03

クラス名...Question03_03

```
package question03;

public class Question03_03 {
    public static void main(String[] args) {

        System.out.print("表示を");
        System.out.println("行います");
        System.out.print("終了します");
    }
}
```

4 リテラル

Question4_1

下記プログラムの「(タブ)」と「(改行)」の部分をエスケープシーケンスに変更し、実行結果の通りに表示するコードを記述しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question04

クラス名...Question04_01

```
package question04;

public class Question04_01 {
    public static void main(String[] args) {
        System.out.print("少年老い易く(タブ)学成り難し(改行)一寸
の光陰軽んずべからず");
    }
}
```

実行結果

```
少年老い易く 学成り難し
一寸の光陰軽んずべからず
```

Question4_2

「10」「桁」「30.8」「回」という文言を、`System.out.print()`を使用して、実行結果の通りに表示しましょう。
その際、「桁」「回」は文字リテラル、「10」は整数リテラル、「30.8」は浮動小数点リテラルとして記述してください。

また、改行の際はエスケープシーケンスを使用してください。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question04

クラス名...Question04_02

実行結果

10 桁

30.8 回

5 変数

Question5_1

異なるデータ型の変数を 5 種類以上宣言（作成）しましょう。

変数名は自由とします。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question05

クラス名...Question05_01

Question5_2

コンソール上に実行結果を表示するコードを記述しましょう。

変数を宣言し(変数名は任意)、値を代入し出力すること。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question05

クラス名...Question05_02

実行結果

変数の値は 50 です

変数の値は 10.5 です

変数の値は true です

Question5_3

「12」「1.6」「こんにちは」「true」の4つの値を、それぞれ個別の変数に代入してコンソール上に出力してみましよう(変数名は任意)。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question05

クラス名...Question05_03

実行結果

```
12
1.6
こんにちは
true
```


Question5_4

以下の要件を全て満たすプログラムを作成して、コンソール上に実行結果を表示させましょう。

1. 整数値型の変数 `i1` を宣言し、「20」を代入する。
2. 整数値型の変数 `i2` を宣言し、「30」を代入する。
3. 文字型の変数 `c` を宣言し、「A」を代入する。
4. 浮動小数点型の変数 `d` を宣言し、「3.14」を代入する。
5. 文字列を格納する変数 `str` を宣言し、「明日から3連休!!!」を代入する。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question05

クラス名...Question05_04

実行結果

```
明日から3連休!!!  
円周率はいつでも3.14です。  
A2030
```

6 入出力

Question6_1

キーボードから値を入力し、実行結果の通りに表示するコードを記述しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question06

クラス名...Question06_01

入力する値...こんばんは

実行結果

こんばんは

こんばんは夜の 20 時です

Question6_2

コンソール上に実行結果を表示するコードを記述しましょう。

その際、キーボードから入力された値を整数に変換し、使用してください。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question06

クラス名...Question06_02

入力する値...30

実行結果

30

今年で 30 歳になります

Question6_3

コンソール上に実行結果を表示するコードを記述しましょう。

その際、キーボードから入力された値を少数に変換し、使用してください。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question06

クラス名...Question06_03

入力する値...25.5

実行結果

25.5

サイズが 25.5 の靴を購入します

Question6_4

キーボードから値を 3 回入力し、実行結果の通りに表示するコードを記述しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question06

クラス名...Question06_04

入力する値（1 回目） ...知恵は

入力する値（2 回目） ...万代の

入力する値（3 回目） ...宝

実行結果

知恵は

万代の

宝

知恵は万代の宝

7 式と演算子

Question7_1

コンソール上に実行結果を表示するコードを記述しましょう。

算術演算子を使用し、実際の計算結果を出力すること。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question07

クラス名...Question07_01

実行結果

12 + 3 は 15 です

12 - 3 は 9 です

12 × 3 は 36 です

12 ÷ 3 は 4 です

4 ÷ 3 の余りは 1 です

Question7_2

変数を宣言してください。

その後、System.out.println()文の中でインクリメント・デクリメント演算子を使用し、実行結果の通りに表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question07

クラス名...Question07_02

使用する変数...下記参照

```
int sum = 10;
```

実行結果

sum に 1 を足すと 11 です

sum から 1 を引くと 10 です

Question7_3

下記ソースコードは計算結果をコンソール上に表示するプログラムです。

表示結果を変えずに、加算代入演算子を使用する形に修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question07

クラス名...Question07_03

```
package question07;

public class Question07_03 {
    public static void main(String[] args) {
        int num = 10;
        num = num + 5;

        System.out.println("合計数は" + num + "です");
    }
}
```


Question7_4

変数 dnum をキャストし、変数 inum へ代入しましょう。

その後、実行結果の通りにコンソール上に表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question07

クラス名...Question07_04

使用する変数...下記参照

```
double dnum = 10.5;  
int inum;
```

実行結果

```
dnum を inum に代入すると 10 になります
```

Question7_5

変数 inum を変数 dnum へ代入しましょう。

その後、実行結果の通りにコンソール上に表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question07

クラス名...Question07_05

使用する変数...下記参照

```
int inum = 10;  
double dnum;
```

実行結果

```
inum を dnum に代入すると 10.0 になります
```

Question7_6

コンソール上に実行結果を表示するコードを記述しましょう。

指定された変数の除算を行い、計算結果を表示すること。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question07

クラス名...Question07_06

使用する変数...下記参照

```
int num1 = 5;  
int num2 = 2;  
double num3 = 2.0;
```

実行結果

```
num1÷num2 は 2 になります  
num1÷num3 は 2.5 になります
```

Question7_7

コンソール上から任意の数字を入力し、0.7 をかけてから「3 割引きで〇〇円です」と出力するプログラムを記述しましょう。

その際、計算結果の小数点以下の値は切り捨ててください。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question07

クラス名...Question07_07

実行結果

123

3 割引きで 86 円です

Question7_8

3つの任意の数字（商品の値段）を入力すると、以下の値を計算して出力するプログラムを記述しましょう。

- ・ 各商品の3割引き価格の合計値
- ・ 上記の合計値の平均値

その際、計算結果の小数点以下の値は切り捨ててください。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question07

クラス名...Question07_08

実行結果

```
100
200
300
合計420円
平均 140 円
```

8 配列

Question8_1

int 型で要素数 5 の配列を 2 つ宣言（作成）しましょう。

1 つは「配列の宣言」と「要素の確保」を 2 文に分けて、もう 1 つは「配列の宣言」と、「要素の確保」をまとめて 1 つの文で宣言すること。

配列変数名は自由とします。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question08

クラス名...Question08_01

Question8_2

下記プログラムは正常に動作しているが、変数の宣言を多用しており、冗長的です。

配列を使用する形に書き換え、ソースコードを読みやすく修正しましょう。

その後、実行結果の通りにコンソール上に表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question08

クラス名...Question08_02

```
package question08;

public class Question08_02 {
    public static void main(String[] args) {
        int ichiro = 88;
        int jiro = 62;
        int saburo = 54;
        int shiro = 76;
        int goro = 45;

        int sum = ichiro + jiro + saburo +
            shiro + goro;

        System.out.println("全員のテストの合計は" + sum + "点で
す");
    }
}
```

実行結果

全員のテストの合計は 325 点です

Question8_3

下記プログラムに新たな要素の追加を行いたい。

int 型の配列 sum に 40 と 50 を追加し、要素数を合計 5 にしましょう。

その後、実行結果の通りにコンソール上に表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question08

クラス名...Question08_03

```
package question08;

public class Question08_03 {
    public static void main(String[] args) {
        int[] sum = {10, 20, 30};

        System.out.println("1 番目の中身は" + sum[0]);
        System.out.println("2 番目の中身は" + sum[1]);
        System.out.println("3 番目の中身は" + sum[2]);

        System.out.println("処理を終了します");
    }
}
```

実行結果

```
1 番目の中身は 10
2 番目の中身は 20
3 番目の中身は 30
4 番目の中身は 40
5 番目の中身は 50
処理を終了します
```


Question8_4

下記プログラムは int 型の配列を 2 種類宣言しています。

二次元配列を使用する形に修正し、配列宣言を 1 つにしましょう。

その後、実行結果の通りにコンソール上に表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question08

クラス名...Question08_04

```
package question08;

public class Question08_04 {
    public static void main(String[] args) {
        int[] num1 = new int[3];
        int[] num2 = new int[3];

        num1[0] = 10;
        num1[1] = 20;
        num1[2] = 30;
        num2[0] = 40;
        num2[1] = 50;
        num2[2] = 60;

        System.out.println("1 段目の 1 つ目の値は" + num1[0]);
        System.out.println("1 段目の 2 つ目の値は" + num1[1]);
        System.out.println("1 段目の 3 つ目の値は" + num1[2]);
        System.out.println("2 段目の 1 つ目の値は" + num2[0]);
        System.out.println("2 段目の 2 つ目の値は" + num2[1]);
        System.out.println("2 段目の 3 つ目の値は" + num2[2]);
    }
}
```

実行結果

1 段目の 1 つ目の値は 10
1 段目の 2 つ目の値は 20
1 段目の 3 つ目の値は 30
2 段目の 1 つ目の値は 40
2 段目の 2 つ目の値は 50
2 段目の 3 つ目の値は 60

9 分岐処理

Question9_1

コンソール上に実行結果を表示するためにはどのような処理の流れが考えられるでしょうか。

- 1 変数 number に 10 以上の数値が代入されている場合は実行結果 1 を表示し、それ以外の数値の場合は実行結果 2 を表示する処理の流れを示すフローチャートをノートに描きましょう。
- 2 作成したフローチャートに基づいてコードを記述しましょう。
コードの記述にあたっては以下の仕様に従うものとします。

パッケージ名...question09

クラス名...Question09_01

実行結果 1

number の値は 10 以上です
処理を終了します

実行結果 2

処理を終了します

Question9_2

コンソール上に実行結果を表示するためにはどのような処理の流れが考えられるでしょうか。

- 1 変数 number に 30 以上の数値が代入されている場合は実行結果 1 を表示し、
それ以外の数値の場合は実行結果 2 を表示する処理の流れを示すフローチャートを
ノートに描きましょう。
- 2 作成したフローチャートに基づいてコードを記述しましょう。
コードの記述にあたっては以下の仕様に従うものとします。

パッケージ名...question09

クラス名...Question09_02

実行結果 1

number の値は 30 以上です
処理を終了します

実行結果 2

number の値は 30 未満です
処理を終了します

Question9_3

- 1 変数 point に 80 以上の数値が代入されている場合は実行結果 1 を表示し、
80 未満 50 以上の数値の場合は実行結果 2 を表示し、
50 未満 30 以上の数値の場合は実行結果 3 を表示し、
それ以外の数値の場合は実行結果 4 を表示する処理の流れを示すフローチャートをノートに描きましょう。
- 2 作成したフローチャートに基づいてコードを記述しましょう。
コードの記述にあたっては以下の仕様に従うものとします。

パッケージ名...question09

クラス名...Question09_03

実行結果 1

テストの点数は優秀です
お疲れ様でした

実行結果 2

テストの点数は平均的です
お疲れ様でした

実行結果 3

テストの点数が及第です
お疲れ様でした

実行結果 4

赤点のため追試が必要です
お疲れ様でした

Question9_4

下記プログラムは正常に動作しているものの、if文を多用しており、冗長的です。

if文をswitch文に書き換え、ソースコードを読みやすく修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question09

クラス名...Question09_04

※new Random().nextInt(①)という記述をすると乱数（ランダムな数値）を発生させることができます。

「0 ～ ①に記入した値-1」の範囲の乱数が発生するため、4を記入すると0～3が発生します。

```
package question09;

import java.util.Random;

public class Question09_04 {
    public static void main(String[] args) {
        int result = new Random().nextInt(4) + 1;
        System.out.println("福引きを購入します");

        if (result == 1) {
            System.out.println("大当たり");
        } else if (result == 2) {
            System.out.println("当たり");
        } else if (result == 3) {
            System.out.println("外れ");
        } else {
            System.out.println("大外れ");
        }
    }
}
```

Question9_5

下記プログラムは入力された値によって条件分岐するプログラムです。

if 文のネストになっている条件分岐の記述を「&&」演算子を使用する形に修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question09

クラス名...Question09_05

```
package question09;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Question09_05 {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new
            InputStreamReader(System.in));

        System.out.println("1 か 2 を入力してください");
        String str = br.readLine();
        int num = Integer.parseInt(str);

        System.out.println("もう一度 1 か 2 を入力してください");
        String str2 = br.readLine();
        int num2 = Integer.parseInt(str2);

        if (num == 1) {
            if (num2 == 1) {
                System.out.println("1 が 2 回入力されました");
            }
        }
    }
}
```

Question9_6

下記プログラムは入力された値によって条件分岐するプログラムです。

if～else 文の記述を「||」演算子を使用して 1 行に修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question09

クラス名...Question09_06

```
package question09;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Question09_06 {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new
            InputStreamReader(System.in));

        System.out.println("1 か 2 を入力してください");
        String str = br.readLine();
        int num = Integer.parseInt(str);

        if (num == 1) {
            System.out.println("1 か 2 が入力されました");
        } else if (num == 2) {
            System.out.println("1 か 2 が入力されました");
        }
    }
}
```


Question9_7

下記プログラムは入力された値によって条件分岐するプログラムです。

2 つ目の if 文の条件の記述を「!」演算子を使用する形に修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question09

クラス名...Question09_07

```
package question09;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Question09_07 {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new
            InputStreamReader(System.in));

        System.out.println("1 以上の数値を入力してください");
        String str = br.readLine();
        int num = Integer.parseInt(str);

        boolean errFlag = false;
        if (num < 1) {
            errFlag = true;
        }
        if (errFlag == false) {
            System.out.println("正常な入力です");
        }
    }
}
```

Question9_8

下記プログラムは入力された値によって条件分岐するプログラムです。

if～else 文の部分を、条件演算子を使用する形に修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question09

クラス名...Question09_08

```
package question09;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Question09_08 {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new
            InputStreamReader(System.in));

        System.out.println("1 を入力してください");
        String str1 = br.readLine();
        int num = Integer.parseInt(str1);

        if (num == 1) {
            System.out.println("1 が入力されました");
        } else {
            System.out.println("1 以外が入力されました");
        }
    }
}
```

Question9_9

BufferedReader を使い、入力された整数が「偶数」か「奇数」かを求めるプログラムを作成しましょう。
記述にあたっては以下の仕様に従うものとします。

パッケージ名...question09

クラス名...Question09_09

実行結果（奇数を入力した場合）

整数を入力してください。

15

15 は奇数です。

実行結果（偶数を入力した場合）

整数を入力してください。

12

12 は偶数です。

Question9_10

BufferedReader を使い、入力された整数が「10 の倍数」かを求めるプログラムを作成しましょう。
記述にあたっては以下の仕様に従うものとします。

パッケージ名...question09

クラス名...Question09_10

実行結果（10 の倍数を入力した場合）

整数を入力してください。

20

20 は 10 の倍数です。

実行結果（10 の倍数ではない数値を入力した場合）

整数を入力してください。

21

21 は 10 の倍数ではありません。

10 繰り返し

Question10_1

コンソール上に実行結果を表示するためにはどのような処理の流れが考えられるでしょうか。

- 1 for 文を使用し、ループカウンタの値を出力する処理の流れを示すフローチャートをノートに描きましょう。カウントするための変数は 0 で初期化することとします。
- 2 作成したフローチャートに基づいてコードを記述しましょう。
コードの記述にあたっては以下の仕様に従うものとします。

パッケージ名...question10

クラス名...Question10_01

実行結果

1 回目の処理
2 回目の処理
3 回目の処理
処理を終了します

Question10_2

下記プログラムの for 文を while 文に修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question10

クラス名...Question10_02

```
package question10;

public class Question10_02 {
    public static void main(String[] args) {
        System.out.println("1 回目の繰り返し処理です");
        for (int i = 0; i < 5; i++) {
            System.out.println((i + 1) + "回目");
        }

        System.out.println("2 回目の繰り返し処理です");
        for (int i = 5; i > 0; i--) {
            System.out.println(i + "回目");
        }

        System.out.println("処理を終了します");
    }
}
```

Question10_3

下記プログラムの while 文を do-while 文に修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question10

クラス名...Question10_03

```
package question10;

public class Question10_03 {
    public static void main(String[] args) {
        int i = 1;

        System.out.println("1 回目の繰り返し処理です");
        while (i <= 5) {
            System.out.println(i + "回目");
            i++;
        }

        int n = 1;

        System.out.println("2 回目の繰り返し処理です");
        while (n <= 10) {
            System.out.println(n + "回目");
            n++;
        }

        System.out.println("処理を終了します");
    }
}
```

Question10_4

コンソール上に実行結果を表示するためにはどのような処理の流れが考えられるでしょうか。

- 1 for 文のネストを利用して出力する流れを示すフローチャートをノートに描きましょう。
- 2 作成したフローチャートに基づいてコードを記述しましょう。
コードの記述にあたっては以下の仕様に従うものとします。

パッケージ名...question10

クラス名...Question10_04

実行結果

```
1 2 3 4 5 6 7 8 9
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```


Question10_5

下記プログラムは 10 回分処理を繰り返し、コンソール上に実行結果を表示します。

繰り返し処理を 5 回目で強制的に終了するように break 文を使用し修正しましょう。

また、for 文の繰り返し条件は修正しないこと。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question10

クラス名...Question10_05

```
package question10;

public class Question10_05 {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            System.out.println(i + "回目の処理です");
        }

        System.out.println("処理を終了します");
    }
}
```

実行結果

```
1 回目の処理です
2 回目の処理です
3 回目の処理です
4 回目の処理です
5 回目の処理です
処理を終了します
```

Question10_6

下記プログラムは 10 回分処理を繰り返し、コンソール上に実行結果を表示します。

繰り返し処理の 5 回目の処理だけを飛ばすように continue 文を使用し修正しましょう。

また、for 文の繰り返し条件は修正しないこと。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question02

クラス名...Question10_06

```
package question10;

public class Question10_06 {
    public static void main(String[] args) {
        for (int i = 1; i <= 10; i++) {
            System.out.println(i + "回目の処理です");
        }

        System.out.println("処理を終了します");
    }
}
```

実行結果

```
1 回目の処理です
2 回目の処理です
3 回目の処理です
4 回目の処理です
6 回目の処理です
7 回目の処理です
8 回目の処理です
9 回目の処理です
10 回目の処理です
処理を終了します
```

Question10_7

下記プログラムは配列の要素の中身を昇順にソートするプログラムです。

for 文のネストを使用している構文を sort()メソッドを使用するコードに修正しましょう。

また、配列を標準出力している for 文を拡張 for 文に修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question10

クラス名...Question10_07

```
package question10;

public class Question10_07 {
    public static void main(String[] args) {
        int[] num = {30, 53, 21, 70, 60};

        for (int i = 0; i < num.length - 1; i++) {
            for (int j = i + 1; j < num.length; j++) {
                if (num[i] > num[j]) {
                    int tmp = num[i];
                    num[i] = num[j];
                    num[j] = tmp;
                }
            }
        }
        System.out.println("ソートが完了しました");

        for (int i = 0; i < num.length; i++) {
            System.out.print(num[i] + " ");
        }
    }
}
```

実行結果

ソートが完了しました

21 30 53 60 70

Question10_8

for 文のネストを利用して実行結果の通りに表示するプログラムを作成しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question10

クラス名...Question10_08

実行結果

```
* * * * *  
* * * * *  
* * * * *
```

※処理が複雑なので、難しい場合は一度フローチャートを作成してみてください

Question10_9

for 文のネストを利用して実行結果の通りに表示するプログラムを作成しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question10

クラス名...Question10_09

実行結果

```
*  
* *  
* * *  
* * * *  
* * * * *
```

※処理が複雑なので、難しい場合は一度フローチャートを作成してみてください

Question10_10

for 文のネストを利用して実行結果の通りに表示するプログラムを作成しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question10

クラス名...Question10_10

実行結果

```
      ■
    □□□
  ■ ■ ■ ■ ■
□□□□□□□
■ ■ ■ ■ ■ ■ ■ ■ ■
□□□□□□□□□□
■ ■ ■ ■ ■ ■ ■ ■ ■ ■ ■
□□□□□□□□□□□□
```

※処理が複雑なので、難しい場合は一度フローチャートを作成してみてください

Question10_11

以下の要件を満たすプログラムを作成しましょう。

1～100 までの数字を 3 と 5 の数字で割り、3 で割れる場合は「Fizz」、5 で割り切れる場合は「Buzz」とコンソールに出力します。

また、3 と 5 両方の数字で割り切れる場合「Fizz Buzz」とコンソールに出力してください。

どちらの数字でも割り切れない場合は、数字をそのままコンソールに出力します。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question10

クラス名...Question10_11

実行結果

```
1
2
Fizz
4
Buzz
Fizz
7
～省略～
14
FizzBuzz
16
～省略～
98
Fizz
Buzz
```

※処理が複雑なので、難しい場合は一度フローチャートを作成してみてください

11 デバッグ①

Question11_1

下記プログラムは実行しても、想定通りの動作（“number は 2 です”の出力）が行われません。

デバッグを使用し、誤っているソースコードを修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question11

クラス名...Question11_01

```
package question11;

public class Question11_01 {
    public static void main(String[] args) {
        int number = 2;

        if (number >= 1) {
            System.out.println("number は 1 以上です");
        } else if (number == 2) {
            System.out.println("number は 2 です");
        } else {
            System.out.println("number はそれ以外の数値です");
        }

        System.out.println("処理を終了します");
    }
}
```


12 クラスの基本

Question12_1

コンソール上に実行結果を表示する為、下記 Cat クラスを使用して、以下の条件を満たすプログラムを作成しましょう。

- ・ Cat クラスのオブジェクトを作成する。
- ・ Cat クラスのフィールド name と age に値を代入する。
- ・ Cat クラスのフィールド name と age の値を出力する。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question12

クラス名...Cat、Question12_01

```
package question12;

class Cat {
    String name;
    int age;
}
```

実行結果

```
名前はコタロウです
年齢は 7 歳です
```

Question12_2

Question12_1 で作成した Cat クラスに以下の内容を追加してください(フィールド名は任意)。

- ・ 身長を保持する double 型のフィールド。
- ・ 体重を保持する double 型のフィールド。
- ・ 好きな食べ物を保持する String 型のフィールド。

また、Question12_02 クラスで Cat クラスをオブジェクト化し、コンソール上にフィールドの値を表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question12

クラス名...Cat、Question12_02

実行結果

身長は 52.3cm です

体重は 4.8kg です

好きな食べ物はささみです

13 引数

Question13_1

以下の条件を満たす Dog クラスを作成しましょう。

- ・ String 型の引数を持ち、それを画面出力する showName()メソッドの作成。

また、Question13_01 クラスで Dog クラスをオブジェクト化し、showName()メソッドを利用して、コンソール上に実行結果を表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question13

クラス名...Dog、Question13_01

実行結果

名前はダニエルです

Question13_2

Question13_1 で作成した Dog クラスに以下の内容を追加してください。

- ・ int 型の引数を持ち、それを画面出力する showAge()メソッドの作成。
- ・ int 型と String 型の引数を持ち、それを画面出力する showProfile()メソッドの作成。

また、Question13_02 クラスで Dog クラスをオブジェクト化し、showAge()メソッドと showProfile()メソッドを利用して、コンソール上に実行結果を表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question13

クラス名...Dog、Question13_02

実行結果

年齢は 5 歳です ※showAge()メソッド

年齢は 5 歳、好きな食べ物はジャーキーです ※showProfile()メソッド

14 戻り値

Question14_1

以下の条件を満たす Calculator クラスを作成しましょう。

- ・ 合計値を求める sum()メソッドの作成。
- ・ 引数は int 型の足される数と、int 型の足す数の 2 種類(引数名は任意)。
- ・ 2 つの引数の足し算の計算結果を戻り値にする。戻り値の型は int 型。

また、Question14_01 クラスで Calculator クラスをオブジェクト化し、sum()メソッドを利用して、コンソール上に実行結果を表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question14

クラス名...Calculator、Question14_01

実行結果

足し算の結果は 3 です ※数値は引数によって異なる

Question14_2

以下の条件を満たす Triangle クラスを作成しましょう。

- ・ 三角形の面積を求める triangleCalc()メソッドの作成。
- ・ 引数は int 型の底辺と、int 型の高さの 2 種類(引数名は任意)。
- ・ 三角形の面積の計算結果を戻り値にする。戻り値の型は int 型。

また、Question14_02 クラスで Triangle クラスをオブジェクト化し、コンソール上に実行結果を表示しましょう。
記述にあたっては以下の仕様に従うものとします。

パッケージ名...question14

クラス名...Triangle、Question14_02

実行結果

三角形の面積は 6 です ※面積は引数によって異なる

Question14_3

以下の条件を満たす Circle クラスを作成しましょう。

- ・ 円の面積を求める circleCalc()メソッドの作成。
- ・ 引数は double 型の半径の 1 種類(引数名：radius)。
- ・ 円の面積の計算結果を戻り値にする。戻り値の型は double 型。

また、Question14_03 クラスで Circle クラスをオブジェクト化し、コンソール上に実行結果を表示しましょう。
記述にあたっては以下の仕様に従うものとします。

パッケージ名...question14

クラス名...Circle、Question14_02

実行結果

円の面積は 28.26 です ※面積は引数によって異なる

15 アクセス制限

Question15_1

下記プログラムは正常に動作しているものの、アクセス修飾子に関する記述が不適切です。

アクセス修飾子を書き換え、ソースコードを正しく修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question15

クラス名...Question15_01

```
package question15;

class Question15_01 {
    public int num;
    public double gas;

    private void setNum(int num) {
        this.num = num;
    }

    void setGas(double gas) {
        this.gas = gas;
    }

    private int getNum() {
        return num;
    }

    double getGas() {
        return gas;
    }
}
```


Question15_2

以下の条件を満たす Profile クラスを作成しましょう。

- ・ アクセス制限のついた年齢フィールド(フィールド名は任意)
- ・ アクセス制限のついた名前フィールド(フィールド名は任意)
- ・ 年齢フィールドの getter メソッド
- ・ 年齢フィールドの setter メソッド
- ・ 名前フィールドの getter メソッド
- ・ 名前フィールドの setter メソッド

また、Question15_02 クラスで Profile クラスをオブジェクト化し、コンソール上に実行結果を表示しましょう。
記述にあたっては以下の仕様に従うものとします。

パッケージ名...question15

クラス名...Profile、Question15_02

実行結果

私の名前はマイケルです

年齢は 20 歳です

16 オーバーロード

Question16_1

下記プログラムには test()メソッドが存在します。

新たに次の仕様を持つ test()メソッドを 2 つ追加しましょう。

1 つ目 : String 型の引数を 1 つ持ち、受け取った値を表示する(引数名は任意)

2 つ目 : int 型と String 型の引数を持ち、「【int 型の引数】回目の【String 型】」と表示する(引数名は任意)

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question16

クラス名...Question16_01

```
package question16;

public class Question16_01 {
    public void test() {
        System.out.println("テスト");
    }
}
```

17 コンストラクタ

Question17_1

下記プログラムにはコンストラクタが存在します。

新たにコンストラクタを2つ追加しましょう。引数および処理は自由とします。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question17

クラス名...Question17_01

```
package question17;

public class Question17_01 {
    Question17_01() {
        System.out.println("コンストラクタです");
    }
}
```

18 Static メンバ

Question18_1

下記プログラムはコンパイルエラーが発生しています。

エラー箇所を特定し、ソースコードを正しく修正しましょう。

なお、「Question18_01 クラスのオブジェクトを生成する」、「新規にクラスを作成する」という手段はとらないでください。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question18

クラス名...Question18_01

```
package question18;

public class Question18_01 {
    int num;

    void message() {
        System.out.println("メッセージを表示します");
        System.out.println("num の初期値は" + num + "です");
    }

    public static void main(String[] args) {
        message();
    }
}
```

19 クラスライブラリ

Question19_1

コンソール上に実行結果を表示するコードを記述しましょう。

1 行目の実行結果の文字数を取得し、表示すること。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question19

クラス名...Question19_01

実行結果

おはようございます。

文字数は 10 になります。

Question19_2

コンソール上に実行結果を表示するコードを記述しましょう。

1 行目の実行結果の 4 番目の文字を取得し、表示すること。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question19

クラス名...Question19_02

実行結果

こんにちは、こんばんは。

4 文字目は'ち'になります

Question19_3

コンソール上に実行結果を表示するコードを記述しましょう。

1 行目の実行結果から「ゲコ」という文字列が最初に出現する箇所を取得し、表示すること。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question19

クラス名...Question19_03

実行結果

カエルがゲコゲコとゲコ池で鳴いている。

ゲコという文字は先頭から 5 番目です

Question19_4

コンソール上に実行結果を表示するコードを記述しましょう。

StringBuilder クラスを使用して、1 行目の実行結果に「ございました」を連結し、表示すること。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question19

クラス名...Question19_04

実行結果

ありがとう

文字列の追加を行います

ありがとうございました

Question19_5

コンソール上に実行結果を表示するコードを記述しましょう。
変数 num と sum の値を比較し、値が大きい方を表示すること。
記述にあたっては以下の仕様に従うものとします。

パッケージ名...question19

クラス名...Question19_05

使用する変数...下記参照

```
int num = 30;  
int sum = 50;
```

実行結果

```
数値の比較をします  
変数 num と sum で大きい方の値は 50 です
```

Question19_6

コンソール上に現在の日時を表示するコードを記述しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question19

クラス名...Question19_06

実行結果

今日の日付を表示します。

yyyy 年 MM 月 dd 日 ※y,M,d には実行時の日時が入ります

Question19_7

下記プログラムは、郵便番号を入力させる処理を記述したものです。

これに正規表現を使って、入力された郵便番号が以下の形式に合致しているかどうかをチェックするコードを追加し、実行結果を表示しましょう。

- ・ 「xxx-xxxx」形式になっているか
- ・ 半角数字になっているか

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question19

クラス名...Question19_07

```
package question19;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Question19_07 {
    public static void main(String[] args) throws IOException {
        System.out.println("郵便番号を入力してください。");
        System.out.println("入力例:xxx-xxxx");
        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));
        String str = reader.readLine();
    }
}
```

実行結果

郵便番号を入力してください。

入力例:xxx-xxxx

534-7716

true

Question19_8

下記プログラムは、半角英字を入力させる処理を記述したものです。

これに正規表現を使って、入力された値が半角英字かどうかをチェックするコードを追加してください。

(大文字小文字は考慮しません)

また、半角英字なら実行結果 1 を、そうでないなら実行結果 2 を表示してください。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question19

クラス名...Question19_08

```
package question19;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Question19_08 {
    public static void main(String[] args) throws IOException {
        System.out.println("半角英字を入力してください。");
        BufferedReader reader = new BufferedReader(new
        InputStreamReader(System.in));
        String str = reader.readLine();
    }
}
```

実行結果 1

半角英字を入力してください。

Abc

Abc は半角英字です。

実行結果 2

半角英字を入力してください。

あいう

あいうは半角英字ではありません。

20 クラス型の変数

Question20_1

実行結果の通りに表示したいのですが、期待している結果になりません。

Question20_01 クラスを修正し、実行結果が表示されるようにしましょう。

記述にあたっては以下の仕様に従うようにしてください。

パッケージ名...question20

クラス名...Test、Question20_01

```
package question20;

public class Test {
    private String str;

    public String getStr() {
        return str;
    }

    public void setStr(String str) {
        this.str = str;
    }
}
```

```
package question20;

public class Question20_01 {
    public static void main(String[] args) {
        Test test1 = new Test();
        Test test2 = new Test();

        test1.setStr("test1");
        test2.setStr("test2");
        test2 = test1;

        System.out.println("1つ目は" + test1.getStr() + "です");
        System.out.println("2つ目は" + test2.getStr() + "です");
    }
}
```

実行結果

```
1 つ目は test1 です
2 つ目は test2 です
```

21 継承

Question21_1

下記プログラムは文字列を保持するクラスです。

下記クラスを継承したクラスを作成しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question21

クラス名...Inheritance、Question21_01

```
package question21;

public class Inheritance {
    private String hobby;

    void setHobby(String hobby) {
        this.hobby = hobby;
    }

    String getHobby() {
        return hobby;
    }
}
```

Question21_2

コンソール上に実行結果を表示する為、Question21_1 で作成したクラスのフィールド hobby に値を格納し、hobby の値を表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question21

クラス名...Question21_02

実行結果

趣味はサッカーです

Question21_3

Question21_1 で作成したクラス Inheritance にコンストラクタを追加し、実行結果のメッセージを表示する処理を記述してください。

その後、クラス Question21_01 のオブジェクトを生成し、コンソール上に実行結果を表示しましょう。
記述にあたっては以下の仕様に従うものとします。

パッケージ名...question21

クラス名...Inheritance、Question21_03

実行結果

Inheritance クラスのコンストラクタが実行されました

22 オーバーライド

Question22_1

show()メソッドを定義した Parent クラスを作成し(show()メソッドの処理は自由)、Parent クラスを継承した Question22_01 クラスを作成してください。

Question22_01 クラスでは、実行結果のメッセージを表示するよう show()メソッドをオーバーライドしてください。

その後、Question22_02 クラスを作成し、コンソール上に実行結果を表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question22

クラス名...Parent、Question22_01、Question22_02

実行結果

オーバーライドした show()メソッドです

23 Object クラス

Question23_1

Frog クラスを作成し、実行結果のメッセージを表示するよう toString()メソッドをオーバーライドしてください。
その後、Question23_01 クラスを作成し、toString()メソッドを使用せずに、コンソール上に実行結果を表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question23

クラス名...Frog、Question23_01

実行結果

井の中の蛙、大海を知らず

Question23_2

Question23_1 で作成した Frog クラスのオブジェクトを 2 つ生成し、比較してください。

比較の結果、同じものであれば実行結果 1 を、違うものであれば実行結果 2 をコンソール上に表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question23

クラス名...Question23_02

実行結果 1

変数 frog1 と変数 frog2 は同じものです

実行結果 2

変数 frog1 と変数 frog2 は違うものです

Question23_3

実行結果の通りに表示したいのですが、期待している結果になりません。

プログラムを修正し、実行結果が表示されるようにしましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question23

クラス名...Question23_03

```
package question23;

public class Question23_03 {
    public static void main(String[] args) {
        String str = new String("エリマキトカゲ");

        if ("エリマキトカゲ" == str) {
            System.out.println("文字列は エリマキトカゲ です");
        }
    }
}
```

実行結果

文字列は エリマキトカゲ です

Question23_4

コンソール上に実行結果を表示する為、下記 Test クラスを使用して、以下の条件を満たすプログラムを作成してください。

- ・ Test クラスのオブジェクトを 2 つ作成する。
- ・ 作成した Test クラスのオブジェクトのハッシュコードを出力する。

記述にあたっては以下の仕様に従うようにしてください。

パッケージ名...question23

クラス名...Test、Question23_04

```
package question23;
```

```
public class Test {  
}
```

実行結果（同じ値が表示されること）

1 つ目のハッシュコードは question23.Test@1234567 です

2 つ目のハッシュコードは question23.Test@abcdefg です

※ハッシュコードは実行する端末ごとに異なる

24 抽象クラス

Question24_1

指定されたクラス名で抽象クラスを作成しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question24

クラス名...Question24_01

宣言するメンバ...メッセージを表示する抽象メソッド show()

Question24_2

Question24_1 で作成した抽象クラスを継承した、Display クラスを作成しましょう。

その際、実行結果のメッセージを表示するようオーバーライドしてください。

その後、Display クラスを使用し、コンソール上に実行結果を表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question24

継承クラス名...Display

実行クラス名...Question24_02

実行結果

馬には乗ってみよ人には添うてみよ

25 インターフェイス

Question25_1

指定された名前でインターフェイスを宣言しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question25

インターフェイス名...Question25_01

宣言するメソッド...引数なし・戻り値 void の display()メソッド

Question25_2

実行結果のメッセージを表示するよう、Question25_01 インターフェイスを実装したクラス Display を作成しましょう。

その後、Question25_02 クラスを作成し、コンソール上に実行結果を表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question25

インターフェイス実装クラス名...Display

実行クラス名...Question25_02

実行結果

インターフェイスを実装しました

Question25_3

新規でインターフェイス Preparation を宣言し、引数なし・戻り値 void の show()メソッドを定義しましょう。

その後、Preparation と Question25_01 の 2 つのインターフェイスを実装するクラス Show を作成しましょう。

その際、実行結果のメッセージを表示するようオーバーライドしてください。

そして最後に、Question25_03 クラスを作成し、コンソール上に実行結果を表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question25

インターフェイス...Preparation、Question25_01(作成済み)

インターフェイス実装クラス名...Show

実行クラス名...Question25_03

実行結果

インターフェイス Preparation を実装しました ※show()メソッド

インターフェイス Question25_01 を実装しました ※display()メソッド

Question25_4

インターフェイス Talk_1 と Talk_2 を宣言しましょう。

Talk_1 には引数なし・戻り値 void の bark()メソッドを記述し、Talk_2 には引数なし・戻り値 void の cry()メソッドを記述すること。

また、一方のインターフェイスを継承して、もう一方のインターフェイスを宣言してください(どちらがどちらを継承するかは任意)。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question25

インターフェイス...Talk_1

インターフェイス...Talk_2

Question25_5

Question25_4 で宣言したサブインターフェイスを実装しましょう。

その際、実行結果のメッセージを表示するようオーバーライドしてください。

その後、Question25_05 クラスを作成し、コンソール上に実行結果を表示しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question25

サブインターフェイス実装クラス名...Voice

実行クラス名...Question25_05

実行結果

犬が吠えました ※bark()メソッド

猫が鳴きました ※cry()メソッド

26 パッケージ

Question26_1

下記プログラムは1 ファイルに3 クラス記述しているプログラムです。
クラスを3 分割し、下記のパッケージ構造になるように配置しましょう。
記述にあたっては以下の仕様に従うものとします。

question26 パッケージ

- └ Question26_01 クラス
- └ question2602 パッケージ
 - └ Question26_02 クラス
- └ question2603 パッケージ
 - └ Question26_03 クラス

```
package question26;

public class Question26_01 {
    public void question1() {
        System.out.println("おはようございます");
    }
}

public class Question26_02 {
    public void question2() {
        System.out.println("こんにちは");
    }
}

public class Question26_03 {
    public void question3() {
        System.out.println("こんばんは");
    }
}
```

27 インポート

Question27_1

コンソール上に実行結果を表示するコードを記述しましょう。

Question26_1 で作成した Question26_01 クラスをインポートし、使用すること。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question27

クラス名...Question27_01

実行結果

おはようございます

Question27_2

コンソール上に実行結果を表示するコードを記述しましょう。

Question26_1 で作成した Question26_02 クラスと Question26_03 クラスをインポートし、使用すること。
記述にあたっては以下の仕様に従うものとします。

パッケージ名...question27

クラス名...Question27_02

実行結果

こんにちは

こんばんは

28 例外

Question28_1

下記プログラムは、int 型に変換する際に例外が発生するコードです。

例外が発生した際に実行結果のメッセージを表示するよう例外処理を追加し、コンソール上に実行結果を表示させましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question28

クラス名...Question28_01

```
package question28;

public class Question28_01 {
    public static void main(String[] args) {
        String str = "こんにちは";
        int num = Integer.parseInt(str);
        System.out.println("変換したら" + num + "になりました");
    }
}
```

実行結果

例外が発生しました

Question28_2

下記プログラムは、例外処理をしていないためコンパイルエラーが発生しています。

例外が発生した際に「例外が発生しました」と表示するよう例外処理を追加し、コンソール上に実行結果を表示させましょう。

また、例外の発生有無に関わらず、必ず「システムを終了します」と表示するようにもしてください。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question28

クラス名...Question28_02

```
package question28;

import java.io.BufferedReader;
import java.io.InputStreamReader;

public class Question28_02 {
    public static void main(String[] args) {
        BufferedReader br = new BufferedReader(new
            InputStreamReader(System. in));

        System.out.println("数値を入力してください");
        String str = br.readLine();
        Integer.parseInt(str);
    }
}
```

実行結果

数値を入力してください

abc

例外が発生しました

システムを終了します

Question28_3

独自の例外クラスを作成しましょう。

作成した独自例外は Question28_03 で発生させてください。

例外が実行した際に「独自例外が発生しました。」と表示するようにコンストラクタを作成してください。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question28

クラス名...Question28_03、TestException(独自例外クラス)

実行結果

```
question28.TestException: 独自例外が発生しました。  
    at question28.Question28_03.main(Question28_03.java:6)
```

Question28_4

Question28_3 で作成した独自例外クラス「TestException」の例外を発生させるプログラムを新しく作成しましょう。Int 型の配列変数「numArray」の要素に値を代入する際に、配列の要素外の添字を指定した上で独自例外「TestException」を発生させるプログラムを作成してください。

例外を発生させる際、例外クラスのコンストラクタの引数は無しとします。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question28

実行クラス名...Question28_04

29 コレクションフレームワーク

Question29_1

下記プログラムは配列を使用して、コンソール上に各要素を表示するプログラムです。

配列を使用している箇所を全て、ArrayList を使用する形に修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question29

クラス名...Question29_01

```
package question29;

public class Question29_01 {
    public static void main(String[] args) {
        String[] fruits = new String[3];

        fruits[0] = "みかん";
        fruits[1] = "ぶどう";
        fruits[2] = "いちご";

        System.out.println(fruits[0]);
        System.out.println(fruits[1]);
        System.out.println(fruits[2]);
    }
}
```

Question29_2

下記プログラムは、ArrayList を使用してコンソール上に表示するプログラムです。

実行結果の通りになるように、適当な箇所に 1 文を追加し、ArrayList の要素を削除しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question29

クラス名...Question29_02

```
package question29;

import java.util.ArrayList;
import java.util.List;

public class Question29_02 {
    public static void main(String[] args) {
        List<String> animals = new ArrayList<String>();

        animals.add("イヌ");
        animals.add("クマ");
        animals.add("フクロウ");

        System.out.println("動物は" + animals + "がいます。");
    }
}
```

実行結果

動物は[イヌ, フクロウ]がいます。

Question29_3

下記プログラムに拡張 for 文を追加し、ArrayList の要素を全てコンソール上に表示しましょう。
記述にあたっては以下の仕様に従うものとします。

パッケージ名...question29

クラス名...Question29_03

```
package question29;

import java.util.ArrayList;
import java.util.List;

public class Question29_03 {
    public static void main(String[] args) {
        List<String> fruits = new ArrayList<>();

        fruits.add("orange");
        fruits.add("grape");
        fruits.add("strawberry");
    }
}
```

実行結果

```
orange
grape
strawberry
```

Question29_4

User クラスを作成し、ArrayList のジェネリクスに指定してください。
実行結果通りになるように要素の追加、出力の処理を記述してください。
記述にあたっては以下の使用に従うものとします。

パッケージ名...question29

クラス名...Question29_04、User

```
package question29;

public class User {
    private String name;
    private int age;

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

実行結果

氏名は鈴木太郎、年齢は 23 歳です。
氏名は渡辺花子、年齢は 25 歳です。
氏名は田中次郎、年齢は 27 歳です。

Question29_5

HashMap を利用し、実行結果の通りに各要素をコンソール上に表示するコードを記述しましょう。

キーと値の紐づけは以下の通りしてください。

キー	:	値
orange	:	100
grape	:	200
strawberry	:	300

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question29

クラス名...Question29_05

実行結果

みかんの価格は 100 円です

ぶどうの価格は 200 円です

いちごの価格は 300 円です

30 デバッグ②

Question30_1

下記プログラムは実行しても、下記想定通りの動作が行われず、例外が発生してしまいます。
デバッグを使用し、誤っているソースコードを修正しましょう。

想定結果

名前は鈴木、年齢は 31 歳、職業は料理人です。
食事を作ります。
名前は田中、年齢は 28 歳、職業は警察官です。
地域や人々の安全を守ります。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question30

抽象クラス名...Constant、Worker

クラス名...Question30_01、Display、Chef、Police

```
package question30;

public abstract class Constant {
    public static final String CHEF = "料理人";
    public static final String POLICE = "警察官";
    public static final String TEACHER = "教師";
}
```

```
package question30;

public abstract class Worker {

    protected String job;
    protected String name;
    protected int age;

    protected Worker(String job, String name, int age) {
        this.job = job;
        this.name = job;
        this.age = age;
    }

    public void showIntroduction() {
        System.out.println("名前は" + name + "、年齢は" + age + "
歳、職業は" + job + "です。");
    }

    public abstract void doWork();
}
```

```
package question30;

public class Question30_01 {
    public static void main(String[] args) {

        Worker[] workers = new Worker[3];

        workers[0] = new Chef("鈴木", 31);
        workers[1] = new Police("田中", 28);

        Display.displayWorkers(workers);
    }
}
```

```
package question30;

public class Display {

    public static void displayWorkers(Worker[] workers) {
        for (int i = 1; i <= workers.length; i++) {

            workers[i].showIntroduction();
            workers[i].doWork();
        }
    }
}
```

```
package question30;

public class Chef extends Worker {

    public Chef(String name, int age) {
        super(Constant.CHEF, name, age);
    }

    public void doWork() {
        System.out.println("食事を作ります。");
    }
}
```

```
package question30;

public class Police extends Worker {

    public Police(String name, int age) {
        super(Constant.TEACHER, name, age);
    }

    public void doWork() {
        System.out.println("地域や人々の安全を守ります。");
    }
}
```

31 コーディング規約

Question31_1

下記プログラムは正常に動作しているものの、一部ソースコードがコーディング規約に反しています。

誤っている部分を特定し、ソースコードを読みやすく修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question31

クラス名...Question31_01

```
package question31;

public class QUESTION31_01 {
    public static void main(String ARGS[])
    {
        int intNumber = 80;
        double doublenumber = 10.5;
        double SUM = intNumber + doublenumber;

        System.out.println("合計値は" + SUM + "です");
    }
}
```

Question31_2

下記プログラムは正常に動作しますが、ソースコードの一部がコーディング規約に反しています。

誤っている部分を特定して、ソースコードを読みやすく修正しましょう。

記述にあたっては以下の仕様に従うものとします。

パッケージ名...question31

クラス名...Question31_02

```
package question31;

public class Question31_02 {
    public static void main(String args[]) {
        int NUM;
        NUM = 5 * 6;
        int MESSAGE_NO = 0;
        String str[] = { "30以上50未満", "25以上30未満" };

        if (MESSAGE_NO == 0) {
            if (NUM >= 30 && NUM < 50) {
                System.out.println(str[MESSAGE_NO]);
            }
        } else if (MESSAGE_NO==1) {
            if (NUM>=25&&30<NUM) {
                System.out.println(str[MESSAGE_NO]);
            }
        }

        System.out.println("処理を終了します");
    }
}
```

Question31_3

下記プログラムは正常に動作しているものの、一部ソースコードがコーディング規約に反しています。
誤っている部分を特定し、ソースコードを読みやすく修正しましょう。（コメントは Javadoc 形式）
記述にあたっては以下の仕様に従うものとします。

パッケージ名...question31

クラス名...Question31_03

```
package question31;

public class Question31_03 {
    int inum = 0;
    private double dnum = 0.0;

    void inum(int inum) {
        inum = inum;
    }

    public int inum() {
        return inum;
    }

    void setgetSample(double dnum) {
        this.dnum = dnum;
    }

    double bufferedReaderNumber () {
        return dnum;
    }
}
```