Java
練習問題

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。

プロジェクト名…java_exercises

パッケージ名…question01

クラス名…Question1_1

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_exercises

パッケージ名…question01

クラス名…Question1_2

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…Question2_1

実行結果

```
出力の練習
１回目
```

Question2_2

System.out.print()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…Question2_2

実行結果

出力の練習　２回目

Question2_3

System.out.println()と System.out.print()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。

System.out.println()は 2 回、System.out.print()は 1 回使用すること。

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…question02

クラス名…Question2_3

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…Question3_1

```
package question03;

public class Question3_1
    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りません。

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…Question3_2

```
package question03;

public class Question3_2 {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 文の直前に 1 行コメントを記述しましょう。

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…question03

クラス名…Question3_3

```
package question03;

public class Question3_3 {

	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

System.out.print()とエスケープシーケンスを使用して、実行結果を表示するコードを記述しましょう。

ただし System.out.print()は 1 度のみの使用とし、System.out.println()は使用してはいけません。

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…Question4_1

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

System.out.print()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。

「桁」「回」は文字リテラル、「10」「30.8」は整数リテラル又は浮動小数点リテラルを使用し、

エスケープシーケンスを組み合わせてそれぞれ別のコードで出力すること。

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…question04

クラス名…Question4_2

実行結果

```
10 桁
30.8 回
```

5　変数

Question05_1

整数と浮動小数型変数を 4 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_1

Question05_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、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_2

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
```

Question05_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ょう。

パッケージ名…question05

クラス名…Question05_3

実行結果

```
12
1.6
こんにちは
true
```

Question05_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。

パッケージ名…question5

クラス名…Question05_4

実行結果

```
明日から 3 連休！！！20
円周率はいつでも 3.14 です。
20＋30＝50
A20
30÷20 の余りは、10 です。
```

※3 行目の「50」と 5 行目の「10」は計算した値を出力してください。

6　標準入力

Question06_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。

キーボードから入力された値を文字列として使用すること。

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…question06

クラス名…Question06_1

入力される値…こんばんは

実行結果

こんばんは夜の 20 時です

Question06_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。

キーボードから入力された値を整数として使用すること。

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…question06

クラス名…Question06_2

入力される値…30

実行結果

```
今年で 30 歳になります
```

Question06_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。

キーボードから入力された値を少数として使用すること。

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…question06

クラス名…Question06_3

入力される値…25.5

実行結果

```
サイズが 25.5 の靴を購入します
```

Question06_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。

キーボードから入力された値を文字列として使用すること。また 3 回入力を行う。

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_4

入力される値（1 回目）… 知恵は

入力される値（2 回目）… 万代の

入力される値（3 回目）… 宝

実行結果

知恵は万代の宝

7　式と演算子

Question07_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。

四則演算を使用し、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_1

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
```

Question07_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。

インクリメント・デクリメント演算子を System.out.println()文の中で使用し、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する。

パッケージ名…question07

クラス名…Question07_2

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
```

Question07_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。

代入演算子の加算代入を使用する形に修正しましょう。表示結果は変えないこと。

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…question07

クラス名…Question07_3

```
package question07;

public class Question07_3 {
	public static void main(String[] args) {

		int num = 10;
		num = num + 5;

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

Question07_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。

キャストによる明示的な型変換を行い、変数 inum の中身を表示すること。

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_4

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
```

Question07_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。

Int 型から dobule 型の変数に値の代入を行い、変数 dnum の中身を表示すること。

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_5

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
```

Question07_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_6

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
```

Question07_7

コンソール上から任意の数字を入力し、1.10 をかけてから「税込み ○○円」（整数で表示）と出力するプログラムを記述しましょう。

なお、計算後に小数点以下の値は切り捨ててください。

パッケージ名…question07

クラス名…Question07_7

実行結果

３４０　※入力値 税込み３６７円

Question07_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。

- 各商品の税込価格の合計値
- 上記の合計値から求めた税込価格の平均値

なお、計算後に小数点以下の値は切り捨ててください。

パッケージ名…question07

クラス名…Question07_8

実行結果

```
100　※入力値
200　※入力値
300　※入力値
合計648円
平均 216 円
```

8　分岐処理

Question08_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…question08

クラス名…Question08_1

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
```

Question08_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…question08

クラス名…Question08_2

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
```

Question08_3

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 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　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。

3　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…question08
クラス名…Question08_3

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
```

Question08_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…question08

クラス名…Question08_4

※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。

①に４をいれて実行すると 0～3 の数値がランダムで決まります。

```
package question08;

import java.util.Random;

public class Question08_4 {
    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");
        }
    }
}
```

Question08_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…question08

クラス名…Question08_7

```
package question08;

import java.util.Scanner;
import java.io.IOException;

public class Question08_7 {
    public static void main(String[] args) throws IOException {

        Scanner stdIn = new Scanner(System.in);
        System.out.println("1 か 2 を入力してください");
        int num = stdIn.nextInt();

        System.out.println("もう一度 1 か 2 を入力してください");
        int num2 = stdIn.nextInt();

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

Question08_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。

条件分岐の記述を”||”演算子を使用して if 文 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…question08

クラス名…Question08_6

```
package question08;

import java.util.Scanner;
import java.io.IOException;

public class Question08_6 {
    public static void main(String[] args) throws IOException {

        Scanner stdIn = new Scanner(System.in);
        System.out.println("1 か 2 を入力してください");

        int num = stdIn.nextInt();

        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");
        }
    }
}
```

Question08_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…question08

クラス名…Question08_7

```
package question08;

import java.util.Scanner;
import java.io.IOException;


public class Question08_7 {
    public static void main(String[] args) throws IOException {

        Scanner stdIn = new Scanner(System.in);
        System.out.println("1 以上の数値を入力してください");

        int num = stdIn.nextInt();

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

Question08_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。

条件分岐の記述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…question08

クラス名…Question08_8

```
package question08;

import java.util.Scanner;
import java.io.IOException;


public class Question08_8 {
    public static void main(String[] args) throws IOException {

        Scanner stdIn = new Scanner(System.in);
        System.out.println("1を入力してください");
        int num = stdIn.nextInt();

        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");
        }
    }
}
```

Question08_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。

パッケージ名…question08

クラス名…Question08_9

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　※入力値
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　※入力値
12 は奇数です。
```

Question08_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なさい。

パッケージ名…question08

クラス名…Question08_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。
```

9　繰り返し

Question09_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、ループ変数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…question09
　クラス名…Question09_1

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
```

Question09_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…question09

クラス名…Question09_2

```
package question09;

public class Question09_2 {
	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");
	}
}
```

Question09_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…question09

クラス名…Question09_3

```
package question09;

public class Question09_3 {
	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");
	}
}
```

Question09_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…question09
　クラス名…Question09_4

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
```

Question09_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…question09

クラス名…Question09_5

```
package question09;

public class Question09_5 {
	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
```

Question09_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…question09

クラス名…Question09_6

```
package question09;

public class Question09_6 {
	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
```

Question09_9

「＊」を用いて、たて 3 つ、横 5 つの四角形を出力してみましょう

なお、 ネストした for 文を利用してください。

パッケージ名…question09

クラス名…Question09_9

実行結果

```
＊＊＊＊＊
＊＊＊＊＊
＊＊＊＊＊
```

Question09_10

for 文を利用して以下の様に表示するプログラムを作成してみましょう。

ただし、標準出力の記述はソースコード中に 1 箇所のみとします。

パッケージ名…question09

クラス名…Question09_10

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

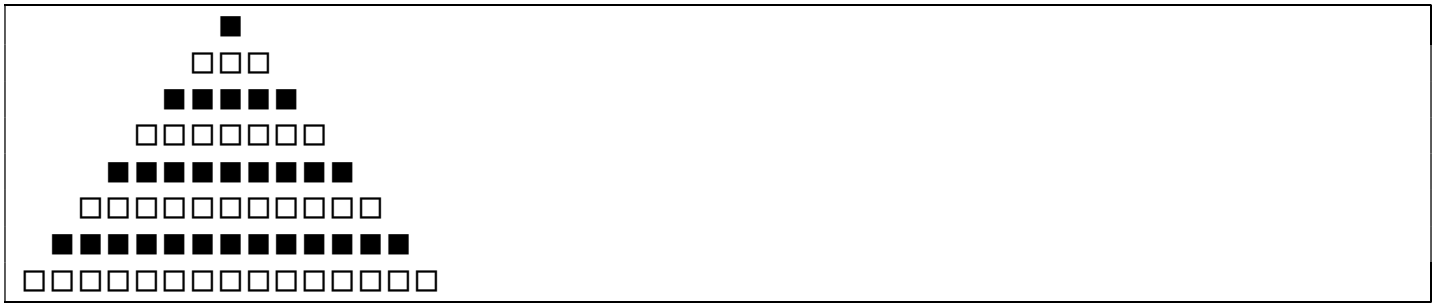
Question09_11

for 文を利用して以下の様に表示するプログラムを作成しなさい。

ただし、標準出力の利用は 1 回のみとする。

パッケージ名…question09

クラス名…Question09_11



Question09_12

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。

パッケージ名…question09
クラス名…Question09_12

```
1
2
Fizz
4
Buzz
Fizz
7

～省略～

14
FizzBuzz
16

～省略～

98
Fizz
Buzz
```

10 配列

Question10_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…question10

クラス名…Question10_1

Question10_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。

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_2

```
package question10;

public class Question10_2 {
    public static void main(String[] args) {

        int english = 88;
        int mathematics = 62;
        int history = 54;
        int science = 76;
        int geography = 45;

        int sum = english + mathematics + history +
                science + geography;

        System.out.println("テストの合計点は" + sum + "点です");
    }
}
```

実行結果

```
テストの合計点は 325 点です
```

Question10_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。

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_3

```
package question10;

public class Question10_3 {
	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
```

Question10_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。

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…Question10_4

```
package question10;

public class Question10_4 {
    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段目の値は" + num1[0] + "です");
        System.out.println("1段目の値は" + num1[1] + "です");
        System.out.println("1段目の値は" + num1[2] + "です");

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

実行結果

```
1 段目の値は 10 です
1 段目の値は 20 です
1 段目の値は 30 です
2 段目の値は 40 です
2 段目の値は 50 です
2 段目の値は 60 です
```

Question10_5

下記プログラムは配列の要素の中身を昇順にバブルソートするプログラムです。

拡張 for 文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。

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_5

```
package question10;

public class Question10_5 {
    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
```

11 コレクションフレームワーク

Question11_1

下記プログラムは配列を使用して、コンソール上に表示するプログラムです。

配列を使用している箇所全て 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…question11

クラス名…Question11_1

```
package question11;

public class Question11_1 {
    public static void main(String[] args) {

        String[] strArray = new String[3];
        strArray[0] = "みかん";
        strArray[1] = "ぶどう";
        strArray[2] = "いちご";

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

Question11_2

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

適当な箇所に 1 文を追加し、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…question11

クラス名…Question11_2

```
package question11;

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

public class Question11_2{
  public static void main(String[] args) {

    List<String> animal = new ArrayList<String>();

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

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

実行結果

```
動物は[イヌ,フクロウ]がいます。
```

Question11_3

以下の通りにキーと値を紐づけ（キー： 値）、

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。

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…question11

クラス名…Question11_3

なお、HashMap を利用すること。

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
```

Question11_4

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

for 文を使用している箇所を、拡張 for 文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…question11

クラス名…Question11_4

```
package question11;

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

public class Question11_4 {
    public static void main(String[] args) {

        List<String> strList = new ArrayList<>();
        strList.add("orange");
        strList.add("grape");
        strList.add("strawberry");

        for(int i = 0 ; i< strList.size() ; i++){
            System.out.println(strList.get(i));
        }
    }
}
```

12 デバッグ

Question12_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…question12

クラス名…Question12_1

```
package question12;

public class Question12_1 {
    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");
    }
}
```

13 クラスの基本

Question13_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,age に値を代入する。
- Cat クラスの show()メソッドで名前と年齢を表示する。

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…question13

クラス名…Question13_1

```
package question13;

class Cat {

    String name;
    int age;

    void show() {
        System.out.println("名前は" + name + "です");
        System.out.println("年齢は" + age + "歳です");
    }
}
```

実行結果

```
名前はコタロウです
年齢は 7 歳です
```

Question13_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。

また、Question14_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。

・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。

・身長と体重と好きな食べ物を表示する profile()メソッドの作成をする。

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…question13

クラス名…Question13_2

修正した Cat クラスを使用してプロフィールをコンソール上に表示すること。

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
```

14 引数

Question14_1

コンソール上に実行結果を表示する為、犬の名前を管理する Dog クラスを作成して、

以下の条件を満たすプログラムを記述しましょう。

・名前を保持する String 型のフィールド。

・名前をフィールドに代入する setName()メソッドの作成。名前（String 型）を指定するための引数を持つ。

・名前を表示する show()メソッドの作成、引数はなし。

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…Question14_1

Dog クラスを使用して名前をコンソール上に表示すること。

実行結果

```
名前はダニエルです
```

Question14_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。

また、Question14_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。

- 年齢を保持する int 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。
- 年齢と好きな食べ物をフィールドに代入する setProfile()メソッドの作成。年齢（int 型）と好きな食べ物（String 型）を指定するための引数を持つ。
- 年齢と好きな食べ物を表示する profile()メソッドの作成、引数はなし。

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…Question14_2

修正した Dog クラスを使用してプロフィールをコンソール上に表示すること。

実行結果

```
年齢は 5 歳です
好きな食べ物はジャーキーです
```

15 戻り値

Question15_1

コンソール上に実行結果を表示する為、鳥の名前を管理する Bird クラスを作成して、
以下の条件を満たすプログラムを記述しましょう。

- 名前を保持する String 型のフィールド。
- 名前を代入する setName()メソッドの作成、引数は鳥の名前で String 型。
- 名前を取得する getName()メソッドの作成、戻り値は String 型

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_1

Bird クラスを使用して、フィールド name の中身をコンソール上に表示すること。

実行結果

名前はぴーちゃんです

Question15_2

コンソール上に実行結果を表示する為、三角形の面積を求める Triangle クラスを作成して、
以下の条件を満たすプログラムを記述しましょう。

- 三角形の面積を求める triangleCalc()メソッドの作成、引数は底辺と高さの二種類の int 型。
- 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。

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_2

Triangle クラスを使用して計算結果を取得し、Question15_2 クラスでコンソール上に表示すること。

実行結果

```
三角形の面積は 6 です
```

Question15_3

下記の要件に沿って Phone クラスと Question15_3 クラスを作成しましょう。

【Phone クラス】

以下のフィールドを定義してください。

Tel フィールド　　：電話番号、String 型

mailAddress フィールド　：メールアドレス 、String 型

以下のメソッドを定義してください。

setTel()メソッド　　：引数の値を tel フィールドに代入する。

setMail()メソッド　　：引数の値を mailAddress フィールドに代入する。

setInfo()メソッド　　：2 つの引数の値を tel フィールドと mailAddress フィールドに代入する。

getTel()メソッド　　：tel フィールドの値を返す。

getMail()メソッド　　：mailAddress フィールドの値を返す。

【 Question15_3 クラス】

main()メソッド内に以下の処理を記述してください。

1. Phone クラスのオブジェクトを 1 つ作成する。
2. setTel()メソッドを呼び出して、オブジェクトの電話番号に値を代入する。
3. setMail()メソッドを呼び出して、オブジェクトのメールアドレスに値を代入する。
4. setInfo()メソッドを呼び出して、オブジェクトの電話番号とメールアドレスに値を代入する。
5. getTel()メソッドを呼び出して、オブジェクトの電話番号の値を取得する。
6. getMail()メソッドを呼び出して、オブジェクトのメールアドレスの値を取得する。
7. 取得した電話番号の値を出力する。
8. 取得したメールアドレスの値を出力する。

※代入する電話番号、メールアドレスの内容は自由に決めてください。

パッケージ名…question15

クラス名…Phone、Question15_3

Triangle クラスを使用して計算結果を取得し、Question15_2 クラスでコンソール上に表示すること。

実行結果

```
09098765432
suzuki@tokyo.com
```

16 アクセス制限

Question16_1

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

アクセス制限を書き換え、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…question16

クラス名…Question16_1

```
package question16;

class Question16_1 {

	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;
	}
}
```

Question16_2

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 … 引数は文字列、受け取った文字列を表示する

2 つ目 … 引数は数値と文字列、「【数値】回目の【文字列】」と表示する

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…question16

クラス名…Question16_2

```
package question16;

public class Question16_2 {

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

}
```

Question16_3

下記プログラムはプロフィールを表示するクラスです。

新たにフィールドの初期化を行うコンストラクタを追加し、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…question16

クラス名…Question16_3

Question16_3 クラスで Profile クラスをオブジェクト化し、コンソール上に表示すること。

```
package question16;

public class Question16_3 {

	private String name;
	private int age;

	public void show() {
		System.out.println("私の名前は" + name + "です");
		System.out.println("年齢は" + age + "歳です");
	}
}
```

実行結果

```
私の名前はマイケルです
年齢は 20 歳です
```

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 個新規追加しましょう。引数は自由とします。

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_1

```
package question17;

public class Question17_1 {

	Question17_1() {
		System.out.println("コンストラクタです");
	}

}
```

Question17_2

下記プログラムにはコンストラクタが存在します。

他クラスからオブジェクト化されないように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…question17

クラス名…Question17_2

```
package question17;

public class Question17_2 {

	public Question17_2(){
		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。

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_1

※なお、「Question18_1 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。

```
package question18;

public class Question18_1 {

	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 クラス型の変数

Question19_1

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ましょう。

・Test クラスのオブジェクトを二つ作成する。

・作成したオブジェクト同士の代入を行う。代入先はどちらでも可とする。

記述にあたっては以下の仕様に従うものとする。

パッケージ名…question19

クラス名…Question19_1

```
package question19;

public class Test {

}
```

実行結果（同じ値が表示される）　※値は端末毎による

```
一つめのメモリーは question19.Test@0000000 です
二つめのメモリーは question19.Test@0000000 です
```

20 カプセル化

Question20_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カプセル化に反しています。

誤っている部分を特定し、ソースコードをより安全な状態にしましょう。

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…question20

クラス名…Question20_1

```
package question20;

public class Question20_1 {

	int num = 0;
	double sum = 0.0;

	void num(int num2) {
		num = num2;
	}

	int num() {
		return num;
	}

	void setgetSample(double Sample) {
		sum = Sample;
	}

	short bufferedReaderShortNumber() {
		return (short) sum;
	}
}
```

21 パッケージ

Question21_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、サブパッケージに配置するように修正しましょう。

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…question21

サブ パッケージ名…question21_2 , question21_3

クラス名…Question21_1 , Question21_2 , Question21_3

```
package question21;

public class Question21_1 {
	public void question1() {
		System.out.println("おはようございます");
	}
}

public class Question21_2 {
	public void question2() {
		System.out.println("こんにちは");
	}
}

public class Question21_3 {
	public void question3() {
		System.out.println("こんばんは");
	}
}
```

22 インポート

Question22_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。

Question21_1 で作成した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…question22

クラス名…Question22_1

実行結果

```
おはようございます
```

Question22_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。

Question21_1 で作成したサブパッケージ内の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…question22

クラス名…Question22_2

実行結果

```
こんにちは
こんばんは
```

23 継承

Question23_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…question23

クラス名…Question23_1

```
package question23;

public class Inheritance {
	private String hobby;

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

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

Question23_2

コンソール上に実行結果を表示する為、Question23_1 で作成したクラスを使用し、

格納した変数の中身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…question23

クラス名…Question23_2

```
package question23;

public class Question23_1 extends Inheritance {

}
```

実行結果

```
Inheritance クラスを継承したクラスがオブジェクト化されました
趣味はサッカーです
```

Question23_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。

作成したクラス Inheritance にコンストラクタを追加し、

コンストラクタに実装したメッセージを表示させましょう。

クラス Question23_1 をオブジェクト生成すること。

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…question23

クラス名…Question23_3

実行結果

Inheritance クラスを継承したクラスがオブジェクト化されました

24 オーバーライド

Question24_1

Parent クラスを継承するクラスを作成し、コンソール上に実行結果を表示するように

show()メソッドをオーバーライドしましょう。

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_1（Parent クラスのサブクラス）

Question24_2（Question24_1 クラスのオブジェクトを生成し、show()メソッドを呼び出すためのクラス。）

実行結果

```
他己紹介をします
```

25 Object クラス

Question25_1

下記プログラムはオブジェクト生成したクラスをコンソール上に表示しています。

下記クラスを toString()メソッドを使用する形で修正し、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…question25

クラス名…Question25_1

```
package question25;

public class Frog {
}
```

```
package question25;

public class Question25_1 {
	public static void main(String[] args) {
		Frog frog = new Frog();
		System.out.println(frog);
	}
}
```

実行結果

```
井の中の蛙、大海を知らず
```

Question25_2

下記プログラムはオブジェクト生成したクラスの比較結果を出力しています。

比較する箇所を修正し、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...question25

クラス名...Question25_2

```
package question25;

public class Question25_2 {
	public static void main(String[] args) {
		Frog frog1 = new Frog();
		Frog frog2 = frog1;

		if (frog1 == frog2) {
			System.out.println("変数 frog1 と 2 は同じものです");
		}
	}

}
```

実行結果

```
変数 frog1 と 2 は同じものです
```

Question25_3

下記プログラムは String 型の文字列比較結果を出力しています。

比較する箇所を修正し、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…question25

クラス名…Question25_3

```
package question25;

public class Question25_3 {
    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("エリマキトカゲは人気があります");
        }

    }
}
```

実行結果

```
エリマキトカゲは人気があります
```

Question25_4

コンソール上に次の出力結果を表示するコードを記述しましょう。

同パッケージに含まれている Frog クラスの階層を表示すること。

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…question25

クラス名…Question25_4

実行結果

今回パッケージに含まれているのは class question25.Frog です

26 抽象クラス

Question26_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…question26

クラス名…Question26_1

宣言するメンバ…メッセージを表示する抽象メソッド show()

Question26_2

コンソール上に実行結果を表示する為、

Question26_1 で作成した抽象クラスを継承したクラス Display 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…question26

継承クラス名…Display

実行クラス名…Question26_2

実行結果

馬には乗ってみよ人には添うてみよ

Question26_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。

Display クラスのオブジェクトを作成し、Question26_1 クラスと instanceof 演算子を使って比較しましょう。

また、同様に Question26_2 クラスのオブジェクトを作成し Question26_2 クラスと比較しましょう。

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…question26

実行クラス名…Question26_3

実行結果

```
Display クラスと Question26_1 クラスは等しいです
作成した変数は Question26_2 クラスになります
```

27 インターフェース

Question27_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…question27

インターフェイス名…Question27_1

アクセス修飾子…public

宣言するメソッド…戻り値 void の display()メソッド

Question27_2

コンソール上に実行結果を表示する為、

Question27_1 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。

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

インターフェイス実装クラス名…Display

実行クラス名…Question27_2

実行結果

インターフェイスを実装しました

Question27_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。

新規でインターフェイス Preparation を宣言し、

Question27_1 と Preparation 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。

また、インターフェイス Preparation に show()メソッドを記述すること。

記述にあたっては以下の仕様に従うものとします。

パッケージ名…question27

インターフェイス…Preparation

インターフェイス実装クラス名…Show

実行クラス名…Question27_3

実行結果

```
インターフェイスを実装しました
インターフェイスを 2 つ実装しました
```

Question27_4

インターフェイスを 2 つ宣言しましょう。

1 つめのインターフェイスで bark()メソッドを記述し、

2 つめのインターフェイスで cry()メソッドを記述すること。

その後、インターフェイス同士で継承しましょう。

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…question27

インターフェイス…Talk_1

インターフェイス…Talk_2

Question27_5

Question27_4 で宣言した、サブインターフェイスを実装しましょう。

bark()メソッドと cry()メソッドをオーバーライドして使用すること。

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

インターフェイス実装クラス名…Voice

実行クラス名…Question27_5

実行結果

```
犬が吠えました
猫が鳴きました
```

28 例外

Question28_1

下記プログラムは、整数型に変換するときに例外が発生するコードです。

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_1

```
package question28;

public class Question28_1 {
    public static void main(String[] args) {

        String str = "こんにちは";
        int num = stdIn.nextInt();

        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。

例外処理を追加し、コンソール上にメッセージを表示させましょう。

また、例外の発生有無に関わらず、必ず処理の終了メッセージをコンソール上に表示すること。

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…question28

クラス名…Question28_2

```
package question28;

import java.util.Scanner;

public class Question28_2 {
    public static void main(String[] args) {

        Scanner stdIn = new Scanner(System.in);
        System.out.println("数値を入力してください");

        int num = stdIn.nextInt();
    }
}
```

実行結果

```
文字を入力してください
文字を入力
例外が発生しました
システムを終了します
```