

A1

プログラム4

サンプルファイルはこちら

exercise01 ▶

after ▶

Exercise01.java

コンソールアプリケーションでユーザーから入力された文字列を受け取り、受け取った文字列を組み込んで作成されたメッセージを表示するための手順に関する理解を問う問題です。

問題文のリスト1(プログラム1)には、コンソールからの入力を扱うためのConsoleオブジェクトを取得するための処理が含まれていないため、コンパイル時にエラーが発生します。また、名前を入力を促すメッセージを表示するためのコード(6行目)がユーザーからの入力を受け取るためのコード(5行目)よりも後に表示されているため、プログラムの流れとしても正しくありません。

問題文のリスト2(プログラム2)では6行目でユーザーからの入力を受け取っていますが、それを変数nameに代入するための処理が抜けています。このため、8行目で使用している変数nameが初期化されていない状態となり、コンパイル時にエラーが発生します。

問題文のリスト3(プログラム3)にはリスト1(プログラム1)同様、コンソールからの入力を扱うためのConsoleオブジェクトを取得するための処理が含まれていないため、コンパイル時にエラーが発生します。

従って、正しいプログラムは問題文のリスト4(プログラム4)となります。実行結果は、図1のようになります。

```

C:\Java10\exercise01\after> java Exercise01
名前を入力してください
Tora
Toraさん、こんばんわ!
C:\Java10\exercise01\after>

```

▲図1: 実行結果

A2

1. + 2. * 3. / 4. -

サンプルファイルはこちら

exercise02 ▶

after ▶

Exercise02.java*

Javaのプログラム内で四則演算を行う際に用いる演算子に関する理解を問う問題です。

このプログラムではすでにおなじみとなったSystem.out.println(式)という命令文を使って、足し算、掛け算、引き算、割り算を行った結果をコンソール画面に表示します。

足し算、掛け算、割り算、引き算のうち、足し算を指示する演算子(+)と引き算を指示する演算子(-)は、小学校の算数で習うものと同じですが、掛け算を指示する演算子(*)と割り算を指示する演算子(/)は独自の記号を使用することに注意してください。

従って問題文のプログラムコードの空欄には、が+、が*、が/、が-となります。

実行結果は、図1のようになります。

*本来であれば、「Exercise02.java」ですが、都合上「Exercise02.java」としています。ご了承ください。

```

C:\Java10\exercise02\after>java Exercize02
16
12
24
750
C:\Java10\exercise02\after>

```

▲図1: 実行結果

A3

1. String (Stringの後に半角空きが入る)
2. println

サンプルファイルはこちら

exercise03 ▶

after ▶

Exercise03.java

コンソール画面上にメッセージを表示する処理についての理解を問う問題です。

こうした処理を行う方法はいくつかありますが、このプログラムではまず画面に表示するためのメッセージを文字列として作成し、一旦変数に代入した上で、その変数を利用してメッセージを表示しています。

メッセージを変数に代入する処理は、問題文のリスト1の3行目で行っています。まずは右辺で”と”で囲んだ文字列を指定し、これを左辺で宣言した変数に代入していますね。□1□の中には、変数 strMessage のデータ型名を代入します。メッセージは文字列ですから、ここに入るのはStringです。

続く4行目では、このstrMessageを画面に表示するための処理を行っています。コンソールアプリケーションの標準出力先(コンソール画面上)に文字列を表示するには、System.out.println(文字列)という命令文を使うのでした。従って、□2□に入るのはprintlnとなります。実行結果は、図1のようになります。

```

C:\Java10\exercise03\after>java Exercise03
Javaで楽しくプログラミング
C:\Java10\exercise03\after>

```

▲図1: 実行結果

A4

1. オブジェクト
2. クラス
3. new演算子

オブジェクト指向プログラミングにおける、クラスとオブジェクトの関係についての理解を問う問題です。

オブジェクト指向プログラミングでは、プログラムをオブジェクトとして作成し、これを組み合わせることでより大きなプログラムを作成していきます。ただし、Javaではオブジェクトをはじめからオブジェクトとして作成するわけではありません。まずはオブジェクトの設計図であるクラスを作成し、これをもとにオブジェクト(またはインスタンスと呼びます)を作成します。

インスタンスを作成する方法はいくつかありますが、最も基本的なのは、クラスの中に定義されているコンストラクタを呼び出す方法です。コンストラクタを呼び出す際には、new演算子を使います。

A5

1. import java.io.InputStreamReader; 2. new BufferedReader (new InputStreamReader(

サンプルファイルはこちら

exercise05 ▶

after ▶

Exercise05.java

コンソールアプリケーションでユーザーから入力された文字列を受け取り、受け取った文字列を組み込んで作成されたメッセージを表示するための手順に関する理解を問う問題です。

ただし、この問題ではユーザーからの入力を受け取るのに、本書で利用したConsoleオブジェクトではなく、BufferedReaderというオブジェクトを使用しています。

BufferedReaderは文字型ストリームからバッファリングしながらテキストを読み込むためのオブジェクトです。本書内ではBufferedReaderをテキストファイルからデータを読み出す処理に利用していましたが、使い方によっては標準入力(この場合はコンソール画面)からデータを読み出すために使用することもできるのです。

標準入力からデータを読み出すためのBufferedReaderオブジェクトを作成しているのは、問題文のリスト1の11行目の部分です。BufferedReaderオブジェクトを作るには、引数にInputStreamReaderオブジェクトを指定する必要がありますが、このInputStreamReaderオブジェクトを作成する際にSystem.inを指定することで、標準入力からデータを読み出すためのBufferedReaderオブジェクトを作成することができます。

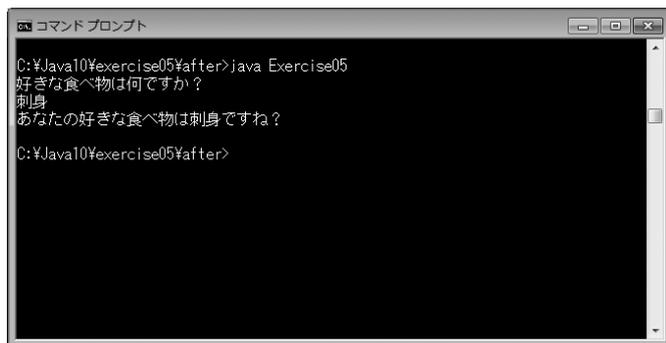
従って問題文のリスト1の には、次のようなコードが入ります。

```
new BufferedReader (new InputStreamReader (
```

また、InputStreamReaderクラスを使うためにはクラスをimportする必要があるため、問題文のリスト1の には、次のようなコードが入ります。

```
import java.io.InputStreamReader;
```

実行結果は、図1のようになります。



▲図1：実行結果

A6

1.分岐構文 2.繰り返し構文 3. if 4. if ~else 5. if ~else if 6. switch
7.for 8. while 9. do ~while 10.拡張for文
(なお 3. 4. 5. 6. は順不同、8. 9. は順不同)

Javaの制御構文に関する理解を問う問題です。

Javaの制御構文には、大きく分けて分岐構文と繰り返し構文の2つの種類があります。

分岐構文というのは、与えられた条件に従って実行処理を分けるための制御構文です。また、繰り返し構文は任意の処理を複数回繰り返して実行するための制御構文です。

分岐構文にはif文、if ~else文、if ~else if文、およびswitch文があります。

繰り返し構文にはfor文、while文、do ~while文があります。また、for文にはJ2SE 5.0で追加された拡張for文と呼ばれる構文があり、これを使うと、配列やリストなどの集合値から要素を順に取り出して処理するコードを、より簡潔に記述することができます。

従って問題文の空欄には、 1 分岐構文、 2 繰り返し構文、 3 if、 4 if ~ else、 5 if ~ else if、 6 switch、 7 for、 8 while、 9 do ~ while、 10 拡張for文、が入ります(3 ~ 6 は順不同、 8 9 は順不同)。

A7

1. for文 2. while文 3. do ~ while文 4. ループカウンタ(またはループ変数) 5. break

Javaの繰り返し構文、およびbreak文についての理解を問う問題です。

繰り返し構文は文字通り繰り返し処理を行う構文で、Javaの繰り返し構文には、for文、while文、do ~ while文があります。

for文では、ループカウンタと呼ばれる変数を使って繰り返しの回数を管理します。ループカウンタには多くの場合、int型などの整数が用いられます。

while文は構文自体に繰り返しの回数を管理する仕組みがありません。従って、「今、何回処理を繰り返したのか」ということをプログラムの中で別途管理し、必要なタイミングで繰り返し処理のブロックから抜け出す仕掛けを組み込む必要があります。処理のブロックから抜け出すためには、break文を使います。

do ~ while文は、while文とよく似た制御構文ですが、繰り返し処理を実行するかしないかを判断するタイミングが若干異なります。

while文では処理を実行する前に条件判定を行うのに対し、do ~ while文ではまず処理を実行してから条件判定を行います。do ~ while文を使用すると、条件式の評価結果にかかわらず、必ず1回は処理を実行させることができます。

従って問題文の空欄には、 1 for文、 2 while文、 3 do ~ while文、 4 ループカウンタ(またはループ変数)、 5 break、が入ります。

A8

1. コンパイルエラー 2. 実行時エラー 3. 例外処理 4. 例外 5. try 6. catch

Javaのプログラムにおけるエラーの種類、および例外処理についての基本的な知識を問う問題です。

プログラムを作成する際、入力したソースコードに文法的な誤りがあり、それが原因でコンパイルが行えないことをコンパイルエラーと呼びます。これに対し、プログラムコードに文法的な誤りはないものの、いざ実行するとエラーが発生するような場合、そのエラーを実行時エラーと呼びます。

実行時エラーは、実行時にプログラムに与えたパラメータが正しくない、外部から読み込んだデータが不正なフォーマットであった、というような状況の下で起こります。

Javaには、こうした実行時エラーに対応するために例外処理と呼ばれる仕組みが用意されています。例外処理では、プログラムに発生する「予期せぬ事態」を例外として扱います。例外の実態は例外オブジェクトと呼ばれるオブジェクトです。例外オブジェクトの設計図となる例外クラスは、java.lang.Throwableクラスのサブクラスとして定義します。

なお、発生した例外を捉えてしるべき処理を行うにはtry ~ catch文を使用し、tryブロックの中で発生したエラーに対する処理をcatchブロックの中に記述します。

従って問題文の空欄には、 1 コンパイルエラー、 2 実行時エラー、 3 例外処理、 4 例外、 5 try、 6 catch、が入ります。

A9

1. if (input == intAnswer) { 2. } else if (input < intAnswer) { 3. } else {

サンプルファイルはこちら

 exercise09 ▶

 after ▶

 Exercise09.java

if ~ else if文を使った分岐構文の組み立てについての理解を問う問題です。

このプログラムは本書のLESSON 12で作成したコンソール画面上で動く数当てゲームのプログラムで、ゲームの

中で何らかの数を「答え」として用意しておき、それをユーザーが当てるといった簡単なゲームを実行するものです。

「答え」は乱数を使ってその都度生成し、ユーザーから入力された「答え」が生成された解答と一致すれば正解、一致しない場合は不正解として、結果をメッセージで表示します。また、不正解の場合は入力された答えが解答よりも大きかったか小さかったかを表示します。

問題文のリスト1のif文による分岐構文は、17～23行目に記述されています。

まず、18行目では「お見事、当たりです!」というメッセージを表示していますので、その直前の□1にはアタリの状況を表す条件文、つまり「ユーザーの入力とコンピュータが用意した回答が等しい」ことを示す条件を指定する必要があります。

20行目では「残念、小さすぎました!」というメッセージを表示していますので、その直前の□2には、「ハズレ」であり、かつユーザーの入力がコンピュータの用意した回答よりも小さい」ということを示す条件を指定する必要があります。

22行目では「残念、大きすぎました!」というメッセージを表示していますので、その直前の□3には、「ハズレ」であり、かつユーザーの入力がコンピュータの用意した回答よりも大きい」ということを示す条件を指定する必要があります。

このように3つの状況に処理を分岐させる処理はif～else if文を用いて記述できるため、□1には、

```
if (input == intAnswer) {
```

□2には、

```
} else if (input < intAnswer) {
```

□3には、

```
} else {
```

が入ります。各実行結果は、図1から3のようになります。

```

C:\Java10\exercise09\after>java Exercise09
コンピュータが思い浮かべた数字を当ててね!
当たりだと思う数を1～10から選んで入力しよう
7
残念、小さすぎました! 答えは10です
C:\Java10\exercise09\after>

```

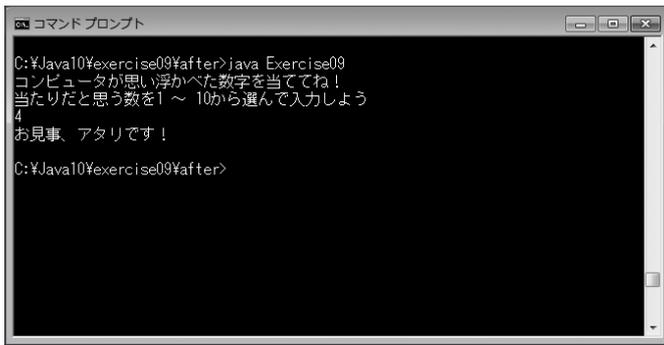
▲図1: 実行結果 (ハズレ: 入力値が小さい場合)

```

C:\Java10\exercise09\after>java Exercise09
コンピュータが思い浮かべた数字を当ててね!
当たりだと思う数を1～10から選んで入力しよう
6
残念、大きすぎました! 答えは10です
C:\Java10\exercise09\after>

```

▲図2: 実行結果 (ハズレ: 入力値が大きい場合)



```

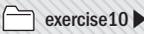
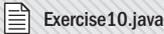
C:\Java10\exercise09\after>java Exercise09
コンピュータが思い浮かべた数字を当ててね！
当たりだと思う数を1～10から選んで入力しよう
4
お見事、当たりです！
C:\Java10\exercise09\after>

```

▲図3: 実行結果 (アタリの場合)

A10

1. 4 2. 0 3. 1 4. 2 5. 3 6. 1

サンプルファイルはこちら   

Javaの配列に関する基本的な理解を問う問題です。

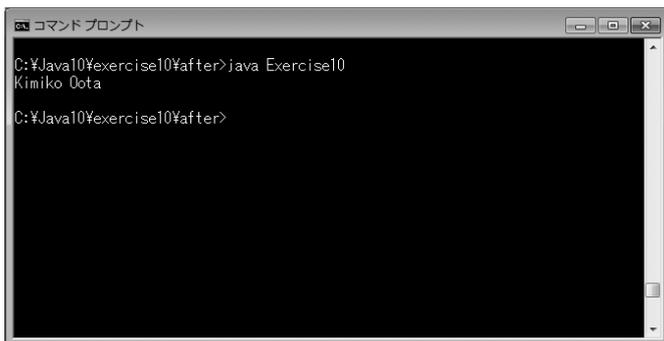
このプログラムはString型の配列に人名を格納し、その後で配列の先頭から2番目に入っている値を取得してコンソール画面に表示します。

Javaの配列は一種のオブジェクトで、new演算子を使って作成します。ただし、クラスからオブジェクトを作成する際には若干手順が異なり、次のように配列に含まれる要素の数を指定してオブジェクトを作成します。

```
String[] list = new String[5];
```

配列の要素にアクセスするには、インデックスと呼ばれる連番を使います。インデックスは、たとえていうなら配列の各要素に割り当てられた「部屋番号」のようなものです。Javaの配列のインデックスは0から始まるため、先頭の要素のインデックスは0、2番目は1、3番目は2、というように順序を表す数値よりも1つ小さな値になることに注意する必要があります。従って問題文のプログラムコードの空欄には、4、0、1、2、3、1、がそれぞれ入ります。

実行結果は、図1のようになります。



```

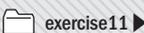
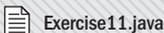
C:\Java10\exercise10\after>java Exercise10
Kimiko Oota
C:\Java10\exercise10\after>

```

▲図1: 実行結果

A11

1. <String> 2. add 3. get 4. 1

サンプルファイルはこちら   

Javaのコレクションフレームワークに含まれるArrayListクラスの使い方について、基本的な理解を問う問題です。

このプログラムは、Q10のプログラムを配列ではなく、ArrayListクラスを使って実現したものです。

ArrayListは可変長配列、つまり配列のサイズを自由に増減させられる配列を表すオブジェクトです。Javaの配列は固定長ですから、作成した時点でサイズが決定されます。これに対してArrayListは、要素を格納したり削除したりすると、それに合わせてサイズが変わります。

ArrayListオブジェクトに要素を追加するには、addメソッドを使います。また、格納した要素を取得するには、getメソッドを使います。getメソッドは引数に取得したい要素のインデックスを指定しますが、このインデックスは配列同様、先頭が0から始まることに注意してください。

なお、J2SE 5.0以降の環境下では、ArrayListオブジェクトを作成する際に、要素として格納する予定のオブジェクトのデータ型を <データ型> という形で指定する必要があります。

従って問題文のプログラムコードの空欄には、 <String>、 add、 get、 1、がそれぞれ入ります。

実行結果は、図1のようになります。

```

コマンドプロンプト
C:\Java10\exercise11\after> java Exercise11
Kimiko Ota
C:\Java10\exercise11\after>
  
```

▲図1：実行結果

A12

1. クラス
2. ディレクトリ、またはフォルダ
3. 関連するクラスをまとめる
4. クラス名の衝突を防ぐ
5. アクセスをコントロールする
6. 階層構造

クラスとオブジェクト、およびパッケージの関係に関する基本的な理解を問う問題です。

Java言語で用いられるパッケージとは、簡単にいうとクラスを入れておくための入れ物のようなものです。パッケージの実体はディレクトリ（Windowsでいうフォルダ）で、名前を持たない無名パッケージと名前を指定して作成される名前付きパッケージの2つの種類があります。

クラスをパッケージに入れて管理することで、関連するクラスをまとめたり、クラス名の衝突を防いだり、パッケージ内のクラスやメンバに対するアクセスを制御したりすることが可能となります。なお、パッケージには階層構造を持たせることができます。例えばtest.partsというパッケージでは、testというパッケージの配下にpartsというパッケージが入っています。

従って問題文の空欄には、 クラス、 ディレクトリまたはフォルダ、 関連するクラスをまとめる、 クラス名の衝突を防ぐ、 アクセスをコントロールする、 階層構造、という用語が入ります。

A13

1. クラス
2. オブジェクトまたはコンストラクタ
3. オブジェクトの初期化
4. デフォルト

コンストラクタに関する基本的な理解を問う問題です。

Javaのクラスにはコンストラクタと呼ばれるものを定義できます。コンストラクタは、クラスからオブジェクト（インスタンス）を生成する際に一度だけ呼び出される特殊なメソッドのようなもので、クラス名と同じ名前を付けて作成されます。コンストラクタ内には、主にオブジェクトを初期化するための処理を定義します。

コンストラクタは必要に応じて異なる引数リストを取るものを複数定義することができます（これをコンストラクタのオーバーロードと呼びます）。

逆に、1つも定義しないことも可能です。コンストラクタを明示的に定義しなかった場合は、引数を取らないデフォルトのコンストラクタが暗黙のうちに作成されることになっています。

ただし、明示的に何らかのコンストラクタを定義した場合はデフォルトのコンストラクタは作成されません。従っ

て、引数を持たないコンストラクタも使用したい場合は、クラス内で明示的に引数を取らないコンストラクタを定義し直す必要があります。

そのため問題文の空欄には、 クラス、 オブジェクトまたはコンストラクタ、 オブジェクトの初期化、 デフォルト、という用語が入ります。

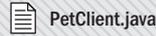
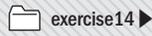
A14

1. package pet 2. type 3. name 4. type 5. name 6. type 7. name

8. type 9. name

(なお 1. のパッケージ名(pet)は別の名前でも可)

サンプルファイルはこちら



クラスの構造、およびフィールドに関する基本的な理解を問う問題です。

このプログラムはPetクラスと、PetClientクラスの2つのクラスで作成されています。Petクラスはペットを表すクラスで、ペットの種類と名前を格納するフィールドを備えています。

PetClientクラスはこのPetクラスからPetオブジェクトを作成して利用するアプリケーションとなるクラスです(mainメソッドを持っています)。

PetClientクラスの中ではPetオブジェクトにフィールドに値を設定したり、それを参照したりして、Petの種類や名前を画面に表示させる処理を行っています。

なお、問題文のリスト1の1行目の部分に入る可能性があるのは、import文かパッケージ宣言のどちらかですが、このプログラムの中ではimport文の必要なクラスは使用していないため、ここにはパッケージ文が入ると考えられます。

従って問題文の空欄には、 package pet、 type、 name、 type、 name、 type、 name、 type、 name (のパッケージ名(pet)は別の名前でも可)が入ります。

実行結果は、図1のようになります。

```

C:\Java10\exercise14\after>javac pet\PetClient.java
C:\Java10\exercise14\after>java pet.PetClient
ペットの種類は犬、名前はボチです。
ペットの種類は猫、名前はタマです。
C:\Java10\exercise14\after>

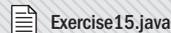
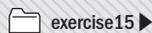
```

▲図1: 実行結果

A15

解答はリスト1の通り。

サンプルファイルはこちら



リスト1 解答

```

001 import javax.swing.*;
002 import java.awt.*;
003

```

```

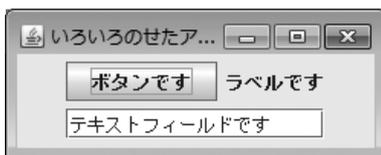
004 public class Exercise15 {
005
006     public static void main(String[] args) {
007
008         // JFrameクラスのインスタンスを作成
009         JFrame frame = new JFrame("練習アプリケーション");
010
011         // レイアウトマネージャ (FlowLayout) のインスタンスを作成
012         FlowLayout f_layout = new FlowLayout();
013
014         // コンテナ (フレーム) にレイアウトをセット
015         frame.setLayout(f_layout);
016
017         // JButtonクラスのインスタンスを作成
018         JButton button = new JButton("This is Button");
019
020         // JLabelクラスのインスタンスを作成
021         JLabel label = new JLabel("This is Label");
022
023         // JTextFieldクラスのインスタンスを作成
024         JTextField text = new JTextField("This is TextBox", 20);
025
026         // フレームにコンポーネントを追加
027         frame.add(button);
028         frame.add(label);
029         frame.add(text);
030
031         // フレーム (ウィンドウ) が閉じる際の処理を定義
032         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
033
034         // フレームのサイズを設定して表示
035         frame.setSize(300, 150);
036         frame.setVisible(true);
037     }
038 }

```

Swingアプリケーション、およびSwingコンポーネントに関する基本的な理解を問う問題です。

問題文のリスト1のプログラムを実行すると、ボタン、ラベル、テキストボックスを配置した小さなSwingアプリケーションが表示されます(図1)。

このプログラムの構造はそのままに、フレームのタイトル、フレームのサイズ、ボタンのキャプション、ラベルのキャプション、テキストフィールドの中に表示するデフォルトの文字列、テキストフィールドの列数を変更するというのが、この練習問題の意図です。



▲図1: 変更前のアプリケーション

まず、フレームのタイトルを設定しているのは、問題文のリスト1の9行目にある、JFrameクラスのコンストラクタを呼び出している箇所です。コンストラクタの引数に指定する文字列がフレームのタイトルとなるため、この行の「いろいろのせたアプリケーション」を「練習アプリケーション」に変更します。

次にフレームのサイズですが、これは35行目でsetSizeメソッドを呼び出して指定しています。この行の「250, 100」の部分を変更すれば、幅300、高さ150のウィンドウを作成できます。

ボタンのキャプションは18行目、ラベルのキャプションは21行目でそれぞれ指定していますので、この部分を表で

示された文字列に変更します。また、テキストフィールドの中に表示するデフォルトの文字列、およびテキストフィールドの列数は、24行目で指定します。

いずれも、コンポーネントを作成する際に呼び出すコンストラクタの引数として指定していることに注意してください。

変更した後のアプリケーションは図2のようになります。



▲図2: 変更後のアプリケーション