章末問題解答

1章

【1】 (1) public static void　(2) String[ ] または String...

【2】 (1) javac　(2) Program

【3】 2 行目の void main の前に，public static が付けられていないことと，main の引数に `String[] args` が指定されていないため．
修正法は，2 行目を `public static void main(String[] args) {`に変える．

2章

【1】 配列 `a` が new 演算子を用いて作成されていない．1 行目を次のように変える．

```
int[] a=new int[2];
```

【2】 (a) `i <= j`　(b) `j <= k`

【3】 (1) 00100000　(2) 11110111　(3) 00111111　(4) 00001111
〔解説〕 (3) と (4) は，このままプログラムをつくり表示すると，11111111 という結果が得られます．ここでは，8 bit で表現された値に対して，それぞれの演算を行ったものとして考えてください．

【4】 ABBD

【5】
```
false
5
```

【6】 `i=(int)j;`

【7】 3 行目で変数 `sum` が宣言されているが，値が初期化されていない．そこで，このプログラムをコンパイルすると，`sum` の値が初期化されていないというコンパイルエラーが出される．

3章

【1】 (2)，(3)，(5)，(7)

【2】 3 行目で 1 整数引数のコンストラクタを定義している．したがって，無引数のコンストラクタは自動生成されない．しかし，5 行目で無引数のコンストラクタを呼び出している．ins2 が 6 行目で宣言されているが，インスタンスはつくられていない．しかし，7 行目で ins2 のフィールド ival を参照している．

【3】 ival はフィールドであるが，クラスメソッド getVal がそのフィールドを参照している（5 行目）．ival はクラス B のフィールドであるが，9 行目でクラス名を指定してその値を参照しようとしている．7 行目の main メソッドの定義に public 修飾子が書かれていない．コンパイルは通るが実行時にエラーとなる．

【4】 2 行目で a には final 修飾子が付けられている．つまり，a は定数である．しかし，5 行目で

その値を変更しようとしている．

【5】 クラス変数には，クラス全体で一つのメモリ領域が割り当てられる．したがって，このクラスのすべてのインスタンスで共有される．一方，フィールドは各インスタンスごとに独自のメモリ領域が割り当てられるため，個々のインスタンスごとに別々の値をもつことができる．

【6】 クラスメソッドは，インスタンスを作成することなく呼び出すことができるメソッドである．逆にインスタンスを作成しないため，クラスメソッドからはフィールドの値を参照できない．

【7】
```
void swap(int i,int j,int[] a) {
  int temp=a[i]; a[i]=a[j]; a[j]=temp;
}
```

【8】 SomeObject クラスは 9 行目で 2 引数のコンストラクタが定義されている．したがって，デフォルトコンストラクタは自動生成されない．しかし，4 行目で無引数コンストラクタを呼び出しているため，実行例に示すエラーが生じた．無引数のコンストラクタをクラス SomeObject に追加すればよい．

4章

【1】 (3)，(4)，(6)，(8)

【2】 (a) `extends P`

【3】 コンストラクタは，そのクラスのインスタンスを生成する．コンストラクタは，クラス名と同じ名前で定義され，メソッドのように戻り型を書かない．

【4】
```
class Involution {
  public static int cube(int i) {
    return i*i*i;
  }
}
```

【5】 オーバーロードは，メソッド名が同じで，引数の型や数が異なるメソッドを複数定義すること．オーバーライドは，スーパークラスで定義されているメソッド（シグネチャが一致する）を，サブクラスで再定義すること．

【6】 **解表 4.1** に示す．

解表 4.1

プログラム	コンパイル時	実行時
`ClassC c=new ClassC();` `ClassP p=c;`	○	○
`ClassP p=new ClassP();` `ClassC c=p;`	×	×
`ClassC c=new ClassC();` `ClassP p=(ClassP)c;`	○	○
`ClassP p=new ClassP();` `ClassC c=(ClassC)p;`	○	×

【7】 同じパッケージ内から参照できる．別の表現をすると，同じフォルダ（ディレクトリ）に属すクラスからは参照できる．

【8】 public が付けられていれば，他のクラスのメソッドからそのクラスのフィールドを参照できる．しかし，private が付けられていれば，そのクラスに属すメソッドからしか値を参照できない．

【9】 同じパッケージ内のクラスのメソッドに加えてサブクラスのメソッドからも呼び出すことができる．

【10】 SuperClass 内の printVal と setVal のアクセス修飾が private になっているため．このアクセス修飾を protected（または無指定や public）に変える．

5章

【1】 (2)，(4)

【2】 サブクラスの定義で，クラスは一つしか継承できない．一方，インタフェースは複数継承できる．クラスはフィールドとメソッドの実装を記述できる．一方，インタフェースは定数のクラス変数と，シグネチャのみのメソッドを定義する．

【3】 あるクラス定義において，別のクラスを extends しなければならない場合．

【4】
```
abstract class Foo {
  public static final double a=10.0;
  public static final int b=5;
  public abstract void printA();
  public abstract void printB();
}
```

【5】
```
class Point implements Entity {
  int x,y;
  public void show() {
    System.out.println("x:"+x+" y:"+y);
  }
}
```

【6】 2 行目で i が定数で初期化されていない．3 行目でコンストラクタが定義されている．

【7】 インタフェースで定義されているメソッドは，public 修飾子をもっているとみなされる．インタフェースを実装したクラス内では，メソッドをオーバーライドする際に，インタフェースより弱い修飾子を指定できない．そこで，5 行目を `public void disp() {`に変える．

6章

【1】 (3)，(4)，(6)，(7)

【2】 import 文を書く．この場合には，つぎの文をファイルの先頭に置く．
```
import java.io.PrintWriter;
```

【3】 Parent クラスは package1 に属すクラスであり，このクラスを他のパッケージに属すクラス（この例では，プログラム 6.3 に示すように package2 に属す Child）から参照することを想定しているため．

【4】 A：例外が発生する可能性がある文． B：発生する可能性がある例外のクラス． C：例外が発生したときに実行する文． D：例外が発生しても，しなくても実行する必要がある文．

【5】 コンパイル時および実行時にクラスパスを指定する．具体的には，コンパイル時と実行時につぎのクラスパス指定をする． `-cp .;\home`

7章

【1】 (1)，(3)，(5)

【2】 start メソッドの最下行に，`stage.show();` を記述する．

【3】 (1) (b)　(2) 全体：(d)，サムネイル：(c)　(3) (a)

【4】 (1) FlowPane　(2) レイアウトに VBox を用いる．

8章

【1】 (1)，(5)

【2】 チェックボックスは，複数の項目を同時に選択可能である．一方，ラジオボタンは同じグループ内のボタンを，同時には一つしか選択できない．

【3】 **(a)** `new Label()`

(b) `event -> charCount()`

(c) `tf,lb`

(d) `lb.getText()`

9章

【1】 27 行目を以下の文で置き換える．

```
gc.fillRect(x,y,10,10);
```

【2】 (2)，(3)，(4)

【3】 `getButton()` の戻り値により知ることができる．この値が PRIMARY なら左, SECONDARY なら右，MIDDLE なら中央が押されたことを表す．

10章

【1】 (1)，(3)，(6)

【2】 (A) `throws`
(B) `"text.txt"`
(C) `pw.close()`

【3】 (A) `new Scanner(str)`
(B) `try {`
(C) `sin.nextInt()`
(D) `sin.nextDouble()`

【4】 `System.out.printf("10 進数:%d,16 進数:%x\n",i,i);`

【5】 UNICODE（UTF–16）

【6】 連結リスト構造や，2 分探索木などをファイル上に実現したとき，連結リストではつぎの要素が置かれている場所を，2 分探索木では子ノードや親ノードが置かれている場所のデータを順に読みたい．このような場合にランダムアクセスファイルが用いられる．

11 章

【1】
```
double r=Math.sin(Math.toRadians(30.0));
```

【2】 Math クラスのコンストラクタは private が指定されており，他のクラスからコンストラクタを呼び出せない．したがって，インスタンス a に対してメソッドを適用する使い方もできない．

【3】 文 A `ClassA<Integer> objA=new ClassA<>(10)`
文 B `ClassA<String> objB=new ClassA<>("String Data")`

【4】 基本データ型 int はクラスオブジェクトではない．一方，クラス Integer はクラスオブジェクトである．

【5】 4 行目をつぎのように変更する．
```
ArrayList<Double> ary=new ArrayList<>();
```

【6】 (1)–(b)　(2)–(c)　(3)–(d)　(4)-(a)

【7】 配列は作成時にサイズを決定する必要がある．一方，ArrayList は要素数が増えればサイズが自動的に拡張される．

12 章

【1】 Thread 以外のスーパークラスが必要ないとき，Thread クラスを拡張する．一方，別のクラスを拡張する必要があるとき，Runnable インタフェースを実装する．

【2】 Thread クラスのコンストラクタ内で start メソッドを呼び出す．または，Thread クラスのオブジェクトに対して start メソッドを呼び出す．

【3】 `alter()` メソッドに `synchronized` 修飾子を付ける．

【4】 `interrupt()` メソッドは，Thread の実行を終了させる．しかし，`interrupt()` メソッドを呼び出すことによりただちに Thread の実行が終了するわけではない．