

第

9

章

総仕上げ問題

- 試験番号：1Z0-818
 - 試験時間：65分
 - 問題数：60問
- 合格ライン：60%

1. 次のようなAクラスとBクラスが定義されている。Main.javaファイルのコンパイル後に生成されるクラスはどれか。正しいものを選びなさい。(1つ選択)

```
1. public class A { }
```

```
1. public class B extends A {}
```

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         A a = new A();  
4.     }  
5. }
```

- A. Aクラス
- B. Aクラス、Mainクラス
- C. Aクラス、Bクラス、Mainクラス
- D. Aクラス、Bクラス

⇒ P37

2. 次のプログラムを実行し、「T」が出力されるようにしたい。5行目に挿入するコードとして、正しいものを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         Game game = new Game();  
4.  
5.         // insert code here  
6.         System.out.print("T");  
7.     } else {  
8.         System.out.print("F");  
9.     }  
10. }  
11. }  
12. class Game {  
13.     boolean flag;  
14. }
```

- A. if(game) {
- B. if(game == "false") {
- C. if(game.flag) {
- D. if(game.flag == "false") {
- E. if(!game.flag) {

⇒ P37

- ☐ 3. 次のプログラムをコンパイル、実行したときの結果として正しいものを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         int a = 3 * 5;  
4.         int b = 25 - 10;  
5.         if (a > b) System.out.println("A");  
6.         if (a < b) System.out.println("B");  
7.         if (a = b) System.out.println("C");  
8.         else System.out.println("D");  
9.     }  
10. }
```

- A. 「C」と表示される
- B. 「D」と表示される
- C. コンパイルエラーが発生する
- D. 実行時に例外がスローされる

⇒ P38

- ☐ 4. 次のプログラムの4行目に挿入するコードとして、正しいものを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         String[] sa = {"a", "b", "c"};  
4.         // insert code here  
5.         System.out.println(s);  
6.     }  
7. }
```

- A. for (s : sa)
- B. for (sa : s)
- C. for (String s : sa)
- D. for (sa : String s)
- E. for (String s : String[] sa)
- F. for (String[] sa : String s)

⇒ P38




5. 次のプログラムをコンパイル、実行したときの結果として正しいものを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         int a = 0;  
4.         do {  
5.             a++;  
6.             System.out.print("hi ");  
7.         } while(a < 4);  
8.     }  
9. }
```

- A. hiが3回表示される
- B. hiが4回表示される
- C. hiが5回表示される
- D. コンパイルエラーが発生する
- E. 実行時に例外がスローされる

⇒ P38

-  6. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Super {
2.     static String name;
3.     public void print() {
4.         System.out.println("Super : " + name);
5.     }
6. }
```

```
1. public class Sub extends Super {
2.     public static void print() {
3.         System.out.println("Sub : " + name);
4.     }
5.     public static void main(String[] args) {
6.         Super a, b;
7.         a = new Super();
8.         b = new Sub();
9.
10.        a.name = "A";
11.        b.name = "B";
12.
13.        a.print();
14.        b.print();
15.    }
16. }
```

- A. Super : B
Super : B
- B. Sub : B
Sub : B
- C. Super : A
Sub : B
- D. コンパイルエラーが発生する

⇒ P39

7. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Parent {  
2.     void printInfo() {  
3.         System.out.println("parent");  
4.     }  
5. }
```

```
1. public class Child extends Parent {  
2.     void printInfo() {  
3.         System.out.println("child");  
4.     }  
5.     public static void main(String[] args) {  
6.         Parent a = new Child();  
7.         a.printInfo();  
8.     }  
9. }
```

- A. 「parent」と表示される
- B. 「child」と表示される
- C. 「parent」「child」と表示される
- D. コンパイルエラーが発生する
- E. 実行時に例外がスローされる

⇒ P39

- ☐ 8. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         System.out.println(args[1] + "," + args[2]);  
4.     }  
5. }
```

【実行方法】

```
java Main easy normal hard
```

- A. 何も表示されない
- B. 「easy,normal」と表示される
- C. 「normal,hard」と表示される
- D. コンパイルエラーが発生する
- E. 実行時に例外がスローされる

⇒ P40

- ☐ 9. 次のプログラムの2行目に挿入するコードとして、正しいものを選びなさい。(2つ選択)

```
1. public interface Sample {  
2.     // insert code here  
3. }
```

- A. String name;
- B. private void setName(String name);
- C. void getName();
- D. public static void sayHello(String name);
- E. public void print();

⇒ P40

□ 10. 次のうち、クラス宣言として有効なものを選びなさい。（3つ選択）

- A. `public class A {}`
- B. `class B extends java.lang.Object {}`
- C. `public class C extends java.lang.* {}`
- D. `final class D {}`
- E. `public class E implements Object {}`

⇒ P41

□ 11. 次のプログラムを実行すると、「B, A」と表示されるようにしたい。
以下の空欄にあてはまるコードを選びなさい。（1つ選択）

```
1. public class B extends A {  
2.     String name = "B";  
3.     public void print() {  
4.         System.out.println(name + ", " + );  
5.     }  
6.     public static void main(String[] args) {  
7.         B b = new B();  
8.         b.print();  
9.     }  
10. }  
11. class A {  
12.     String name = "A";  
13. }
```

- A. `this.name`
- B. `super.name`
- C. `this(name)`
- D. `super().name`
- E. `A.name`

⇒ P41

- 12.** 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         String a1 = "A";  
4.         String a2 = new String("A");  
5.         String a3 = new String("A");  
6.  
7.         if(a1 == a2) {  
8.             System.out.print("a1 == a2 ");  
9.         } else {  
10.            System.out.print("a1 != a2 ");  
11.        }  
12.        if(a2 == a3) {  
13.            System.out.print("a2 == a3");  
14.        } else {  
15.            System.out.print("a2 != a3");  
16.        }  
17.    }  
18. }
```

- A. 「a1 == a2 a2 == a3」と表示される
- B. 「a1 != a2 a2 == a3」と表示される
- C. 「a1 == a2 a2 != a3」と表示される
- D. 「a1 != a2 a2 != a3」と表示される

⇒ P42

13. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class B extends A {  
2.     void test() {  
3.         System.out.println("B");  
4.     }  
5.     public static void main(String[] args) {  
6.         B b = new B();  
7.         A a = b;  
8.         a.test();  
9.     }  
10. }  
11. class A {  
12.     void test() {  
13.         System.out.println("A");  
14.     }  
15. }
```

- A. 「A」と表示される
- B. 「B」と表示される
- C. コンパイルエラーが発生する
- D. 実行時に例外がスローされる

⇒ P43

- 14.** 次のプログラムをコンパイル、実行した結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         int i = 1;  
4.         double d = 1.0;  
5.         double num = i * d;  
6.         switch(num) {  
7.             case 1 :  
8.                 System.out.print("1");  
9.             case 1.0 :  
10.                 System.out.print("1.0");  
11.                 break;  
12.             default :  
13.                 System.out.print("default");  
14.         }  
15.     }  
16. }
```

- A. 「1」と表示される
- B. 「1.0」と表示される
- C. 「default」と表示される
- D. コンパイルエラーが発生する

⇒ P43

- 15.** Webベースのシステムを開発するには、どのJavaテクノロジーを使えばよいか。最適なものを選びなさい。(1つ選択)

- A. Java SE
- B. Java ME
- C. Java EE
- D. Java DB

⇒ P43

16. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Sample {  
2.     String name;  
3.     public void print() {  
4.         System.out.println(name);  
5.     }  
6.     public static void main(String[] args) {  
7.         Sample s1;  
8.         Sample s2;  
9.         s1.name = "A";  
10.        s2.name = "B";  
11.        s1.print();  
12.        s2.print();  
13.    }  
14. }
```

- A. 「A」「B」と表示される
- B. 「B」「B」と表示される
- C. コンパイルエラーが発生する
- D. 実行時に例外がスローされる

⇒ P44

17. JVM (Java Virtual Machine) に関する説明として、正しいものを選びなさい。(1つ選択)

- A. クラスファイルを読み込み、機械語にコンパイルする
- B. ソースコードを事前にコンパイルしてから実行する
- C. 実行可能ファイルを生成する
- D. クラスファイルを逆アセンブルする

⇒ P44

- 18. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         int i = 3;  
4.         while(i >= 0) {  
5.             System.out.println(i--);  
6.         }  
7.     }  
8. }
```

- A. 「2」「1」「0」と表示される
- B. 「3」「2」「1」と表示される
- C. 「3」「2」「1」「0」と表示される
- D. 「3」「2」「1」「0」「-1」と表示される
- E. 2が無限に表示される
- F. 3が無限に表示される

⇒ P45

- 19. 次のうち、コンパイルエラーとなるものを選びなさい。(1つ選択)

- A. byte a = -100;
- B. short b = 70000;
- C. int c = 100000000;
- D. long d = 1234567L;

⇒ P45

- 20. 次のうち、privateキーワードで修飾できないものを選びなさい。(3つ選択)

- A. クラスのコンストラクタ
- B. インタフェースのメソッド
- C. クラスの抽象メソッド
- D. クラスの具象メソッド
- E. インタフェースの定数
- F. クラスの変数

⇒ P46

21. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. class B extends A {  
2.     int b, c;  
3.     B(int num) {  
4.         b = num;  
5.     }  
6.     B(int num, int num2) {  
7.         this(num);  
8.         c = num2;  
9.     }  
10. }
```

```
1. class A {  
2.     int a;  
3.     A() { a = 1; }  
4.     A(int num) { a = num; }  
5. }
```

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         B b = new B(2, 3);  
4.         System.out.println(b.a + ", " + b.b + ", " + b.c);  
5.     }  
6. }
```

- A. 「0,2,3」と表示される
- B. 「1,2,3」と表示される
- C. 「2,0,3」と表示される
- D. 「2,2,3」と表示される

⇒ P47

22. privateメソッドへのアクセスに関する説明として、正しいものを選びなさい。(1つ選択)

- A. 同じクラスのメソッドからアクセスできる
- B. サブクラスのメソッドからアクセスできる
- C. 同じパッケージに属するすべてのクラスからアクセスできる
- D. スーパークラスのメソッドからアクセスできる

⇒ P48

- 23.** 次のプログラムのコンパイルを成功させるためには、空欄にどのコードを追加すればよいか。正しいものを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         hello();  
4.     }  
5.      void hello() {  
6.         System.out.println("hello");  
7.     }  
8. }
```

- A. public
- B. static
- C. final
- D. 何も必要ない

→ P48

- 24.** 次のプログラムをコンパイル、実行したときの説明として、正しいものを選びなさい。(1つ選択)

```
1. class A {  
2.     private int num = 10;  
3.     void print() {  
4.         System.out.println(num);  
5.     }  
6. }
```

```
1. public class B extends A {  
2.     public static void main(String[] args) {  
3.         B b = new B();  
4.         b.print();  
5.     }  
6. }
```

- A. numがprivateで修飾されているためコンパイルエラーになる
- B. printメソッドがBクラスに定義されていないためコンパイルエラーになる
- C. 実行時に例外がスローされる
- D. 10が表示される

→ P49

25. 次のプログラムのコンパイルを成功させるには、Testクラスの5行目にどのコードを追加すればよいか。正しいものを選びなさい。（1つ選択）

```
1. class Test {  
2.     private String a,b;  
3.     private String c = "C";  
4.     public Test() {  
5.         // insert code here  
6.     }  
7.     public Test(String b) {  
8.         this.b = b;  
9.     }  
10.    public void print() {  
11.        System.out.println(a + "," + b);  
12.    }  
13. }
```

```
1. public class Sample {  
2.     public static void main(String[] args) {  
3.         Test t = new Test();  
4.         t.print();  
5.     }  
6. }
```

- A. a = "A";
this("B");
- B. this("B");
a = "A";
- C. this(c);
a = "A";
- D. a = "A";
this.Test(c);
- E. Test("B");
a = "A";

⇒ P49

- 26.** 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. class Sample {  
2.     public String val;  
3. }
```

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         Sample s = new Sample();  
4.         if (s.val == "") {  
5.             s.val = "test";  
6.         }  
7.         System.out.println(s.val);  
8.     }  
9. }
```


- A. 「null」と表示される
- B. 「test」と表示される
- C. コンパイルエラーが発生する
- D. 実行時に例外がスローされる

⇒ P50

- 27.** アクセス修飾子privateで修飾できる要素として、正しいものを選びなさい。(1つ選択)

- A. クラス、フィールド、コンストラクタ、具象メソッド
- B. クラス、フィールド、抽象メソッド
- C. 抽象クラス、フィールド、具象メソッド
- D. インタフェース、具象メソッド
- E. フィールド、抽象メソッド

⇒ P50

-  **28.** 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)


```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         System.out.println(args[0] + args[1]);  
4.     }  
5. }
```

【実行方法】

```
java Main test
```

- A. 「test」と表示される
- B. 「Main test」と表示される
- C. 「test null」と表示される
- D. 実行時に例外がスローされる

⇒ P51

-  **29.** 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public interface A {  
2.     int test(int x, int y);  
3. }
```

```
1. public class B implements A {  
2.     public int test(int a, int b) {  
3.         return (a * b) /2;  
4.     }  
5. }
```

```
1. public class C implements A {  
2.     public int test(int c, int d) {  
3.         return (int)(3.2 * (c * d));  
4.     }  
5. }
```

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         A[] array = {new B(), new C()};  
4.         System.out.print(array[0].test(3, 2) + " ");  
5.         System.out.println(array[1].test(3, 2));  
6.     }  
7. }
```

- A. Cクラスのコンパイルに失敗する
- B. 「3 19」が表示され、例外がスローされる
- C. Mainクラスの4行目でコンパイルエラーになる
- D. 「3 19」が表示される
- E. Bクラスの3行目でコンパイルエラーになる

→ P51

30. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Shape {  
2.     public void print() {  
3.         System.out.println("shape");  
4.     }  
5. }
```

```
1. public class Triangle extends Shape {  
2.     public void print() {  
3.         System.out.println("triangle");  
4.     }  
5. }
```

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         Shape s = new Shape();  
4.         Triangle t = (Triangle) s;  
5.         t.print();  
6.     }  
7. }
```

- A. 「shape」と表示される
- B. 「triangle」と表示される
- C. コンパイルエラーが発生する
- D. 実行時に例外がスローされる

⇒ P52

31. 次のメソッドを正しくオーバーロードしているメソッドを選びなさい。(1つ選択)

```
public void test(int a, int b) {}
```

- A. `public void test(int a) {}`
- B. `public int test(int a, int b) {}`
- C. `public void sample() {}`
- D. `public test(int a, int b) {}`

⇒ P52

32. 次の説明のうち、正しいものを選びなさい。(4つ選択)

- A. パッケージ宣言は必須ではない
- B. パッケージ宣言は、ソースファイルの先頭行に記述しなければならない
- C. 1つのソースファイル内に、インタフェースとクラスの両方を定義できる
- D. インポート宣言は1つだけ記述できる
- E. インポート宣言はソースファイルのどこに記述してもよい
- F. 1つのソースファイル内に、finalクラスは1つだけ定義できる
- G. ソースファイルの名前は、public宣言されたクラスの名前と一致させなければならない

⇒ P53

33. 次のプログラムを実行し、コンソールに1から4の値を順に表示したい。空欄にあてはまるコードを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         for (int i = 0;  ) {  
4.             System.out.println(i);  
5.         }  
6.     }  
7. }
```

- A. ++i < 5;
- B. i++ < 5;
- C. i < 5; i++
- D. i < 5; ++i

⇒ P54

34. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。（1つ選択）

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         int a = 10;  
4.         int b = 20;  
5.         if (a != 10)  
6.             System.out.println("A");  
7.         else if(a < b)  
8.             System.out.println("B");  
9.         else  
10.            System.out.println("C");  
11.     }  
12. }
```

- A. 「A」と表示される
- B. 「B」と表示される
- C. 「C」と表示される
- D. 「B」「C」と表示される
- E. コンパイルエラーが発生する

→ P55

35. 次のクラスを継承したクラスが持つべきメソッドとして、正しいものを選びなさい。（1つ選択）

```
abstract class Sample {  
    public abstract void test();  
}
```

- A. public void test() {}
- B. void test() {}
- C. public abstract void test() {}
- D. public void test(String val) {}

→ P55

- 36.** 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Sample {
2.     static String val = "sample";
3.     public Sample(String val) {
4.         this.val = val;
5.     }
6. }
```

```
1. public class Main {
2.     public static void main(String[] args) {
3.         Sample s = new Sample();
4.         Sample s2 = new Sample("test");
5.         System.out.println(s.val);
6.         System.out.println(s2.val);
7.     }
8. }
```

- A. 「null」「test」と表示される
- B. 「sample」「test」と表示される
- C. 「test」「test」と表示される
- D. コンパイルエラーが発生する

→ P56

- 37.** フィールドを適切にカプセル化し、値が不用意に変更されないように定義したい。適切に記述されているコードを選びなさい。(1つ選択)

- A. `public abstract int a;`
- B. `public final int b;`
- C. `private static int c;`
- D. `private final int d;`
- E. `private abstract int e;`

→ P56

38. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         int[] array = {2, 4, 6, 8};  
4.         int[] array2 = {1, 3, 5, 7, 9};  
5.         array = array2;  
6.         for (int i = 0; i < array.length; i++) {  
7.             System.out.println(array[i]);  
8.         }  
9.     }  
10. }
```

- A. 「2」「4」「6」「8」「1」と表示される
- B. 「2」「4」「6」「8」「9」と表示される
- C. 「1」「3」「5」「7」と表示される
- D. 「1」「3」「5」「7」「9」と表示される
- E. コンパイルエラーが発生する
- F. 実行時に例外がスローされる

⇒ P57

39. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         int x = 10;  
4.         test(x);  
5.         System.out.println(x);  
6.     }  
7.     private static void test(int a) {  
8.         x++;  
9.     }  
10. }
```

- A. 10が表示される
- B. 11が表示される
- C. コンパイルエラーが発生する
- D. 実行時に例外がスローされる

⇒ P57

40. 次のプログラムを実行して、0から2までを順に表示したい。4行目に挿入するコードとして、正しいものをを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         int x = 0;  
4.         // insert code here  
5.     }  
6. }
```

- A. while(++x < 3) { System.out.println(x); };
- B. while(x++ < 3) { System.out.println(x); };
- C. do while(++x < 3) { System.out.println(x); };
- D. do { System.out.println(x); } while(++x < 3);

→ P57

41. 次のプログラムを実行し、1から5までを順に表示したい。空欄①と②にあてはまるコードを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         int[] array = {1, 2, 3, 4, 5};  
4.         for ( ; i < array.length;  ) {  
5.             System.out.println(array[i]);  
6.             i++;  
7.         }  
8.     }  
9. }
```

- A. ① int i = 0 ② i++
- B. ① int i = 1 ② i++
- C. ① int i = 0 ② なし
- D. ① int i = 1 ② なし

→ P58

- ☐ 42. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。（1つ選択）

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         String str = null;  
4.  
5.         if(str == null) {  
6.             System.out.println("if");  
7.         } else if(str == "null") {  
8.             System.out.println("else if");  
9.         } else {  
10.            System.out.println("else");  
11.        }  
12.    }  
13. }
```

- A. 「if」と表示される
- B. 「else if」と表示される
- C. 「else」と表示される
- D. コンパイルエラーが発生する
- E. 実行時に例外がスローされる

⇒ P58

- ☐ 43. Javaの継承に関する説明として、正しいものを選びなさい。（2つ選択）


- A. 多重継承ができる
- B. 1つのクラスは、複数のサブクラスから継承されることができる
- C. 継承したクラスは、継承元のすべてを引き継ぐ
- D. サブクラスからさらにサブクラスを作ることができる

⇒ P59

- ☐ 44. 次のうち、abstractで修飾できないものを選びなさい。（2つ選択）

- A. クラス
- B. 変数
- C. メソッド
- D. パッケージ

⇒ P59

-  **45.** 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         Sample s = new Sample();  
4.         int result = s.test() + s.getNum();  
5.         System.out.println(result);  
6.     }  
7. }
```

```
1. class Sample {  
2.     private static int num;  
3.     public static int getNum() {  
4.         return ++num;  
5.     }  
6.     public int test() {  
7.         return getNum();  
8.     }  
9. }
```

- A. 3が表示される
- B. 例外がスローされ、何も表示されない
- C. Sampleクラスのtestメソッドでコンパイルエラーになる
- D. Mainクラスのmainメソッドでコンパイルエラーになる

➡ P60

46. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。（1つ選択）

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         Sample[] samples = {new Test(), new Exam(), new Test()};  
4.         for (Sample s : samples) {  
5.             s.test();  
6.         }  
7.     }  
8. }
```

```
1. interface Sample {  
2.     void test();  
3. }
```

```
1. class Test implements Sample {  
2.     public void test() {  
3.         System.out.println("A");  
4.     }  
5. }
```

```
1. class Exam {  
2.     public void test() {  
3.         System.out.println("B");  
4.     }  
5. }
```

- A. 「A」「B」「A」と表示される
- B. Sampleインタフェースのコンパイルに失敗する
- C. Examクラスのコンパイルに失敗する
- D. Mainクラスのコンパイルに失敗する
- E. 実行時に例外がスローされる

→ P61

- 47.** 次のうち、インタフェースに定義できないものを選びなさい。(2つ選択)

- A. `public String a = "A";`
- B. `abstract String b;`
- C. `private String c = "C";`
- D. `void setData(String data);`
- E. `abstract void setData(String data);`
- F. `public void setData(String data);`

→ P61

- 48.** 次のプログラムを実行し、「test」と1回だけ表示したい。空欄にあてはまるコードを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         int i = 4;  
4.         while(i++ < ) {  
5.             System.out.println("test");  
6.             i++;  
7.         }  
8.     }  
9. }
```

- A. 4
- B. 6
- C. 7
- D. 8

→ P62

49. `com.sample.controller`パッケージに属するクラスを使う、`com.sample.view`パッケージに属するクラスを定義したい。定義方法として正しいものを選びなさい。(1つ選択)

- A. `package com.sample.view;`
`import com.sample.controller;`
- B. `import com.sample.controller;`
`package com.sample.view;`
- C. `package com.sample.view;`
`import com.sample.controller.*;`
- D. `import com.sample.controller.*;`
`package com.sample.view;`

⇒ P62

50. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         for(int i = 5; i > 0; i--) {  
4.             System.out.print("e");  
5.         }  
6.     }  
7. }
```

- A. 「eeee」と表示される
- B. 「eeeee」と表示される
- C. 何も表示されない
- D. コンパイルエラーが発生する
- E. 実行時に例外がスローされる

⇒ P63

51. 次のうち、正しい説明を選びなさい。(2つ選択)

- A. サブクラスに、スーパークラスのメソッドと名前が同じで引数が異なるメソッドを定義することを「オーバーロード」と呼ぶ
- B. サブクラスに、スーパークラスのメソッドと名前が同じで引数が異なるメソッドを定義することを「オーバーライド」と呼ぶ
- C. サブクラスに、スーパークラスのメソッドとシグニチャが同じメソッドを定義することを「オーバーロード」と呼ぶ
- D. サブクラスに、スーパークラスのメソッドとシグニチャが同じメソッドを定義することを「オーバーライド」と呼ぶ

→ P63

52. 次のプログラムに関する説明として、正しいものを選びなさい。(1つ選択)

```
1. public class Sample {  
2.     private static int num = 10;  
3.     public void Sample() {  
4.         this(10);  
5.     }  
6.     private Sample(int n) {  
7.         num = n;  
8.     }  
9. }
```

- A. コンストラクタからstaticフィールドにアクセスしているためにコンパイルエラーになる
- B. コンストラクタの戻り値型をvoidにしているためコンパイルエラーになる
- C. コンストラクタをprivateで修飾しているためコンパイルエラーになる
- D. メソッドからコンストラクタを呼び出しているためコンパイルエラーになる

→ P63

53. 次のプログラムのうち、コンパイルエラーになるのは何行目か。正しいものを選びなさい。(3つ選択)

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         int[] array1 = { 1, 2, 3 };  
4.         int array2 = array1;  
5.         int[] array3 = new int[3];  
6.         array3 = new int[5];  
7.         int[] array4 = new int(4);  
8.         int[3] array5;  
9.     }  
10. }
```

- A. 3行目
- B. 4行目
- C. 5行目
- D. 6行目
- E. 7行目
- F. 8行目

⇒ P64

- 54.** 次のプログラムの実行後に、クラスのインスタンスはいくつ生成されるか。正しい個数を選びなさい。(1つ選択)

```
1. public class Item {  
2.  
3. }
```

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         Item item1 = new Item();  
4.         Item item2 = new Item();  
5.         Item item3 = null;  
6.         item1 = null;  
7.         item2 = null;  
8.  
9.     }  
10. }
```

- A. 0個
- B. 1個
- C. 2個
- D. 3個

→ P65

- 55.** スーパークラスの要素と同じ名前で見定義できるサブクラスの要素として、正しいものを選びなさい。(2つ選択)

- A. フィールド
- B. privateで修飾されていないコンストラクタ
- C. privateで修飾されたコンストラクタ
- D. finalで修飾されていないメソッド
- E. finalで修飾されたメソッド

→ P65

□ 56. 次のうち、クラス名として使えるものを選びなさい。(3つ選択)

- A. A#
- B. \$B
- C. C%
- D. D9
- E. E-
- F. F_

⇒ P65

□ 57. メソッドのシグニチャを構成する要素として、誤っているものを選びなさい。(3つ選択)

- A. 修飾子
- B. メソッド名
- C. 引数の数
- D. 引数の型
- E. 引数の順序
- F. 引数の名前
- G. 戻り値型

⇒ P66

□ 58. 次のうち、ポリモーフィズムにもっとも関係がある用語を選びなさい。(1つ選択)

- A. 継承
- B. オーバーライド
- C. オーバーロード
- D. インタフェースの継承

⇒ P66

59. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Main {
2.     public static void main(String[] args) {
3.         int i = 2;
4.         System.out.println((i += 2) + (i++));
5.     }
6. }
```

- A. 4が表示される
- B. 6が表示される
- C. 8が表示される
- D. 9が表示される

⇒ P66

60. 次のプログラムをコンパイル、実行したときの結果として、正しいものを選びなさい。(1つ選択)

```
1. public class Main {
2.     public static void main(String[] args) {
3.         Sample s = new Sample();
4.         long data = s.test(10);
5.         System.out.println(data);
6.     }
7. }
```

```
1. class Sample {
2.     public int test(int a) {
3.         return a * 2;
4.     }
5.     public long test(int b) {
6.         return b * 3;
7.     }
8. }
```

- A. 20が表示される
- B. 30が表示される
- C. コンパイルエラーが発生する
- D. 実行時に例外がスローされる

⇒ P67

第9章 総仕上げ問題

解 答

1. B

→ P2

依存関係のあるクラスのコンパイルに関する問題です。

Mainクラスをコンパイルするとき、このクラスの3行目では、newキーワードを使ってAクラスのインスタンスを生成しています。このように利用しているクラスがある場合、利用するクラスのクラスファイルがなければ、コンパイラはその利用しているクラスも、利用しているクラスをコンパイルすると同時にコンパイルします。よって、Aのクラスファイルも生成されます。BクラスはAクラスを継承しています。3行目を「A a = new B();」と記述してAのサブクラスであるBのインスタンスを生成するのであれば、Mainクラスの実行にはBクラスが必要です。そのため、Mainクラスをコンパイルすれば、同時にBクラスもコンパイルされます。しかし、設問のコードではBクラスを使っていないため、Mainクラスをコンパイルしても、Bクラスはコンパイルされることはありません。

以上のことから、選択肢**B**が正解です。

2. E

→ P3

フィールドの初期値に関する問題です。

フィールドはローカル変数と違い、明示的に値を代入しなかった場合には**デフォルト値**で初期化されます。設問のGameクラスのflagフィールドは、boolean型のデフォルト値であるfalseで初期化されています。

設問では、if文の条件式がtrueを戻すことで、「T」をコンソールに表示できます。そのため、game.flagがfalseであることを条件にした選択肢**E**が正解です。

選択肢Aは、Gameのインスタンスへの参照を持つ変数gameを条件式に記述しているだけで、boolean型の値を戻す条件式ではありません。

選択肢Bは、Gameのインスタンスへの参照が、falseという文字列のインスタンスとの同一性を評価しています。

選択肢Cは、Gameのインスタンスが持っているflagフィールドの値を評価していますが、flagフィールドの値はfalseであるため、elseブロックが実行されることになります。

選択肢Dは、flagフィールドの値と文字列 "false" への参照を比較していますが、boolean型と参照型には互換性がないため、比較そのものができません。

以上のことから、これらの選択肢は誤りです。

第9章

総仕上げ問題（解答）

3. C

→ P4

if文による条件分岐に関する問題です。if文の条件式は、**boolean型**の値を返す式でなければいけません。しかし、設問のコードでは7行目で代入演算子「=」を使って変数aに変数bの値を代入しているため、int型の値が戻されます。そのため、設問のコードはコンパイルエラーとなります。以上のことから、選択肢**C**が正解です。

もし、左右オペランドの値が等しいかどうかを調べるのであれば、==演算子を使わなければいけません。

4. C

→ P4

拡張for文の構文に関する問題です。**拡張for文**は、カッコ「()」の中に変数と集合（コレクションもしくは配列）をコロン「:」で区切って記述します。コロンをはさんだ左側に変数、右側に集合を記述するため、集合を右側に記述していない選択肢B、D、Fは誤りです。

選択肢Eは、コロンをはさんだ右側に配列型変数を宣言しています。コロンの右側に記述するのは、集合への参照（が代入された変数）です。よって、誤りです。

また、拡張for文内で使用する変数は、拡張for文のカッコ内で宣言しなければいけません。よって、選択肢Aは誤りです。

以上のことから、選択肢**C**が正解です。

5. B

→ P5

do-while文に関する問題です。while文とdo-while文は、繰り返し条件の判定を、繰り返し処理を実行する前に判定するか、それとも繰り返し処理を実行したあとに判定するかの違いがあります。

ただし、2つの違いは、初回の繰り返し処理が実行されない可能性があるときのみ生じます。**do-while文の場合は、必ず1回はdoブロック内の処理が実行されます**。while文の場合は、条件により繰り返し処理が実行されないこともあります。設問のコードでは、変数aが0から始まり、4よりも小さい間繰り返し処理が実行されます。そのため、設問のdo-while文は、次のように書き換えることができます。

例 設問のコードをwhile文に書き換え

```
while(a < 4) {
    a++;
    System.out.print("hi ");
}
```

処理の内容は、前述のとおり変数aが0から始まり、4よりも小さい間繰り返し続けます。ただし、繰り返し処理の最初でインクリメントしているため、変数aは1から繰り返し出力されることになります。よってコンソールには1、2、3、4の順に表示され、4になった時点で条件式がfalseを戻して繰り返しを抜けます。

以上のことから、選択肢**B**が正解です。

6. D

→ P6

オーバーライドに関する問題です。インスタンスメソッドをstaticメソッドでオーバーライドするとコンパイルエラーになります。

設問のコードでは、Superクラスで定義されたprintメソッドを、Subクラスでstaticで修飾してオーバーライドしています。そのため、このコードはコンパイルエラーが発生します。よって、選択肢**D**が正解です。

7. B

→ P7

オーバーライドに関する問題です。サブクラスでスーパークラスのメソッドをオーバーライドすると、サブクラスのインスタンスを使う場合には、オーバーライドしたメソッドが実行されます。

設問のコードでは、Parentクラスを継承したChildクラスでprintlnInfoメソッドをオーバーライドしています。そのため、Childクラスのインスタンスを使うとChildクラスでオーバーライドしたメソッドが実行されます。よって、選択肢Aは誤りです。

設問のコードで間違えやすいのは、変数の「型」のメソッドが呼び出されるか、生成した「インスタンス」のメソッドが呼び出されるかという点です。変数の型は、インスタンスの扱い方を決めていただけであり、どの型で扱っても、実際に動作するのはインスタンスそのものです。たとえば、設問のコードであれば、ChildクラスのインスタンスをParent型で扱っても、動作するのはChildのインスタンスです。以上のことから、ChildクラスのprintlnInfoメソッドが実行され、コンソールには「child」と表示されます。よって、選択肢**B**が正解です。

8. C

→ P8

コマンドライン引数に関する問題です。

コマンドライン引数は、javaコマンド実行時に渡す値のことです。javaコマンドの第1引数には、実行するクラス名を記述します。第2引数以降は、スペース区切りで値を列挙すると、mainメソッドの引数として、String型しか扱わない配列型の参照が渡されます。

設問の実行方法では、Mainクラスの実行時に3つのコマンドライン引数を渡しています。1つ目はeasy、2つ目はnormal、3つ目はhardという文字列です。これらの文字列は、mainメソッドの引数であるString型の配列の要素として格納されているため、mainメソッド内で取り出すには、引数argsを使ってアクセスします。

配列の添字は0から始まるため、設問のコードのように「args[1]」と「args[2]」のように添字を指定すると、2つ目と3つ目の値を取り出して表示することになります。そのため、コンソールには「normal,hard」と表示されます。よって、選択肢Cが正解となります。

9. C, E

→ P8

インタフェースのメンバに関する問題です。

- A. フィールドを宣言しています。インタフェースに定義するフィールドは初期化しなければいけません。そのため、このコードはコンパイルエラーとなります。
- B. メソッドを宣言しています。インタフェースにはメソッドの実装を記述できず、宣言だけを記述します。ただし、インタフェースに定義できるメソッドの宣言は、publicなものだけです。このメソッド宣言はprivateで修飾しているので、コンパイルエラーとなります。
- C. このメソッド宣言は、publicで修飾されていません。このように明示的にpublicを記述しなかったとしても、コンパイラによって自動的にpublicが追加されるためコンパイル可能です。
- D. staticなメソッドの宣言です。staticメソッドは、インタフェースに定義できません。よって、コンパイルエラーとなります。
- E. publicなメソッドの宣言です。このコードは、メソッドの実装を持たない宣言のみの定義です。よって、インタフェースに定義できるコンパイル

可能なコードです。

したがって、選択肢**C**と**E**が正解です。

10. A、B、D

→ P9

クラスの宣言方法に関する問題です。

選択肢Aは、もっとも一般的なクラス宣言です。よって、選択肢**A**は有効なクラス宣言です。

選択肢Bは、`java.lang.Object`クラスを継承している点とアクセス修飾子がデフォルトになっている点が選択肢Aと異なります。すべてのクラスは`java.lang.Object`クラスのサブクラスとなることが言語仕様で決められています。選択肢Aのようにプログラマーが明示的に継承を記述しなかった場合でも、コンパイラが自動的に`java.lang.Object`クラスを継承するコードを追加します。また、クラス宣言時に使えるアクセス修飾子は、`public`とデフォルトの2種類です（インナークラスの場合は、これら以外の修飾子も使えます）。以上のことから、選択肢**B**は有効なクラス宣言です。

選択肢Cは、継承の宣言時にアスタリスク「*」を使ってワイルドカード指定をしています。が、`java.lang`パッケージのいずれかのクラスを継承するというような宣言はできず、継承するクラスを明示しなければいけません。選択肢Cは無効なクラス宣言です。

選択肢Dは、`final`クラスの宣言です。`final`クラスはサブクラスを作ることができないクラスのことです。たとえば`java.lang.String`クラスは`final`クラスの一例です。よって、選択肢**D**は有効なクラス宣言です。

選択肢Eは、`Object`クラスをimplementsしようとしています。`implements`は、インタフェースを実現するときに使うキーワードで、クラスに対して使うことはできません。よって、選択肢Eは無効なクラス宣言です。

11. B

→ P9

サブクラスのインスタンスからスーパークラスのインスタンスにアクセスするための方法に関する問題です。サブクラスをインスタンス化すると、サブクラスとスーパークラスの両方のインスタンスが生成され、両方で1つのインスタンスとして扱われます。そのため、サブクラスのインスタンスからスーパークラスのインスタンスにアクセスすることが可能です。

サブクラスのインスタンスからスーパークラスのインスタンスにアクセスす

第9章

総仕上げ問題（解答）

るタイミングは次の2とおりです。

- ・ サブクラスのコンストラクタ内でスーパークラスのコンストラクタを呼び出す
- ・ サブクラスのメソッド内でスーパークラスのメソッドやフィールドにアクセスする

1つ目のタイミングでは、スーパークラスのコンストラクタを`super()`で呼び出します。2つ目のタイミングでは、`super`を使ってアクセスします。

設問のコードで「B, A」と表示するには、サブクラスの`print`メソッド内で、スーパークラスのフィールドにアクセスしなければいけません。そのため、選択肢Bのように`super`を使ってアクセスします。選択肢Dは、スーパークラスのコンストラクタを呼び出しています。以上のことから、選択肢Bが正解で、選択肢Dは誤りです。

選択肢Aは、インスタンス自身を表す`this`を使っています。設問のコードでは、`this`はBクラスのインスタンスを表すこととなります。そのため、コンソールには「B, B」と表示されます。よって、選択肢Aは誤りです。

選択肢Cは、コンストラクタ内で同じクラスに定義されている別のコンストラクタを呼び出すコードです。このようなコンストラクタを呼び出すコードは、コンストラクタ内でしか呼び出せないため、コンパイルエラーとなります。よって、選択肢Cも誤りです。

選択肢Eのような「クラス名.フィールド名」という書式は、`static`フィールドにアクセスするときの記述方法です。しかし、設問のAクラスに定義された`name`フィールドはインスタンスフィールドです。よって、このコードはコンパイルエラーが発生します。以上のことから、選択肢Eは誤りです。

12. D

→ P10

==演算子と同一性に関する問題です。

同じ内容の文字列リテラルを代入してStringのインスタンスを作った場合には、コンスタントプールの仕組みにより、その都度、新しいインスタンスを生成せずに、1つのインスタンスを使い回します。一方、`new`キーワードを使った場合には、たとえ同じ内容の文字列を持っていたとしても、必ず新しいインスタンスを生成します。

設問では、3つの文字列を生成しています。3行目では文字列リテラルを代入し、4行目と5行目では`new`キーワードを使ってインスタンスを生成していま

す。文字列リテラルは1回しか使われていないため、この3つの文字列は、別々のインスタンスで扱われることになります。よって、3つの変数a1、a2、a3には、異なるインスタンスへの参照が代入されています。

7行目と11行目のif文の条件式では**==演算子**を使って変数a1、a2、a3の同一性を比較していますが、これらは両方ともfalseを戻すため、elseブロックの処理が実行され、コンソールには「a1 != a2 a2 != a3」が表示されます。よって、選択肢**D**が正解です。

13. B

→ P11

オーバーライドとポリモーフィズムに関する問題です。サブクラスのインスタンスをスーパークラス型で扱っても、動作するのはサブクラスのインスタンスです。変数の型は、あくまでも扱い方を宣言するもので、インスタンスの型を変更するものではありません。そのため、スーパークラスのメソッドをサブクラスでオーバーライドした場合には、変数がスーパークラス型であっても、オーバーライドしたメソッドが有効になります。

設問のコードでは、Aクラスを継承したBクラスを定義し、さらにtestメソッドをオーバーライドしています。そのため、A型で扱っていても、Bのインスタンスのtestメソッドを呼び出すと、オーバーライドしたメソッドが実行されます。よって、コンソールには「B」が表示されます。以上のことから、選択肢**B**が正解です。

14. D

→ P12

switch文に関する問題です。

設問のコード3行目ではint型の変数iを、4行目ではdouble型の変数dをそれぞれ初期化しています。5行目では、iとdの乗算結果をdouble型の変数numに代入しています。この乗算では、int型の1がdouble型に精度を合わせてdouble型に変換されるため、numの値は1.0になります。

switch文の式は、char、byte、short、int、Character、Byte、Short、Integer、String、列挙型を戻す式でなければいけません。設問の式にはdouble型のnumが指定されているため、コンパイルエラーになります。よって、選択肢**D**が正解です。

15. C

→ P12

Javaのエディションに関する問題です。Javaには、Java SE（Java Platform, Standard Edition）、Java EE（Java Platform, Enterprise Edition）、Java ME（Java Platform, Micro Edition）の3つのエディションがあります。選択肢DのJava DB

は、Javaが提供する簡易DBMS（データベース管理システム）であり、エディションの名前ではありません。よって、誤りです。

選択肢Aの**Java SE**はほかの2つのエディションのベースのエディションで、Javaの基本的な機能を提供します。実際の業務アプリケーションなどでは、Java SEを応用したJava EEやJava MEが使われます。Webベースのシステムを開発するには、**Java EE**を使うのが一般的です。Java SEは基本機能を提供しているため、Java SE単独でもWebベースのシステムを開発できないことはありませんが、非効率的で適していません。以上のことから、選択肢**C**が最適と言えます。

選択肢Bの**Java ME**は「マイクロエディション」と呼ばれ、ロボットなどの産業機器、モバイル機器で使うためのエディションです。よって、誤りです。

16. C

→ P13

ローカル変数の初期化に関する問題です。初期化されていないローカル変数を参照するとコンパイルエラーになります。一方、インスタンスフィールドとして宣言した変数は、自動的にデフォルト値で初期化されます。そのため、フィールドの場合には初期化しなくてもコンパイルエラーとはなりません。間違えやすいので注意しましょう。

設問のコードでは、mainメソッド内で2つのSample型変数s1とs2を宣言しています。しかし、これらの変数は初期化されることなく、次の行でnameフィールドに値を代入しようとしています。そのため、このコードはコンパイルエラーが発生します。以上のことから、選択肢**C**が正解です。

17. A

→ P13

JVM（Java Virtual Machine）の機能に関する問題です。

CやC++は、ソースコードをコンパイルしておき、プログラムを実行するOSごとに実行可能ファイルを生成する「事前コンパイル方式」を採用しています。一方、Javaは「実行時コンパイル方式」を採用し、インタープリタとしての役割を持つ**JVM**を介して、ソースコードをコンパイルしながら実行します。このときにJVMは、ソースコードのコンパイル後に生成されるクラスファイルを読み込み、機械語にコンパイルして実行します。

逆アセンブルとは、機械語からソースコードに変換することです。JVMには逆アセンブルの機能はありません。

以上のことから、選択肢**A**が正解です。

18. C

→ P14

後置デクリメントの実行タイミングに関する問題です。デクリメント演算子も、前置と後置で実行のタイミングが異なります。気を付けなければいけないのは、後置されたときです。後置の場合は、処理が実行されてからデクリメントが実行されることに注意しましょう。

設問のコードでは、変数*i*が3で初期化され、デクリメントされながら表示されていきます。このデクリメント演算子は後置されているため、*i*の値は表示されてから1減ります。よって、最初に表示されるのは3です。以上のことから、数列が2から始まる選択肢Aは誤りです。

while文の条件式は、変数*i*の値が0以上であればtrueを戻すため、3、2、1、0と順に表示し、その後、*i*の値が-1になった時点でwhile文から抜けます。以上のことから、選択肢Cが正解です。

19. B

→ P14

データ型の範囲に関する問題です。すべてのデータ型の範囲を正確に覚える必要はありませんが、範囲が狭いbyte型とshort型の2つについては覚えておきましょう。

byte型は-128～127まで、short型は-32768～32767までの値を表すことができます。覚えられない場合には、byte型が8ビット、short型はその倍と覚えておいて、次のように2進数で表してみるとよいでしょう。

【byte型とshort型の範囲】

byte									0	0	0	0	0	0	0	0
short	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10進数	32768	16394	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1

このように2進数で0、1がいくつ並ぶかを考え、10進数に直すとちょうど最後のビットが表す値を倍にすると、その型が表せる数値の範囲となります。その半分が正の値、もう半分が負の値となります。

たとえば8ビットのbyte型の場合、2進数で表す値の最小が「00000000」、最大が「11111111」です。最大「11111111」のとき、最後の8ビット目が表すのは128という値です。つまり、「10000000」は128を表しています。この最後のビットが表す数を倍にした256がbyte型が表せる数字の範囲であり、byteは256通りの数値を表すことができます。byte型は正の値と負の値の両方を表すことができるため、256通りのうち半分を正の値（0～127）、残り半分を負の値（-128～-1）のために使います。正の値は0を含んで128通りなのがポイントです。

こうすれば、細かな値を覚える必要はありませんので、迷ったらこのような表を書いてみましょう。

設問のコードでは、選択肢Bがshort型の変数にその範囲を超える値70000を代入しています。この値はint型、もしくはlong型の変数でしか扱えません。よって、**選択肢B**はコンパイルエラーになります。

20. B、C、E

→ P14

アクセス制御の範囲に関する問題です。

クラスの**コンストラクタ**は、オーバーロードして複数定義することが可能です。そのとき、外部のクラスに公開したいコンストラクタと公開したくないコンストラクタに分けることができます。公開したいコンストラクタには**public**を、公開したくない、もしくは公開範囲を限定したいコンストラクタにはそれ以外の修飾子を付けることができます。もちろん、**private**で修飾することも可能です（**選択肢A**）。

なお、**private**で修飾されたコンストラクタは、外部のクラスからは使えませんが、同じクラスに定義されている別のコンストラクタから**this()**を使って呼び出すことが可能です。

インタフェースは、外部に公開する宣言部分だけを抜き出したものです。そのため、**private**を使ってメソッドを非公開にすることはできません。同様にインタフェースに定義する定数も公開するためのものです。よって、定数も非公開にすることはできません（**選択肢B、E**）。

抽象メソッドは、サブクラスでの実装を強制するためのメソッドです。**private**で修飾した抽象メソッドは、サブクラスでオーバーライドできなくなり、実装を提供することができません（**選択肢C**）。

実装を持たない抽象メソッドに対して、**具象メソッド**は実装を持ちます。外部のクラスに非公開にしたいメソッドは**private**で修飾します（**選択肢D**）。

クラスの変数は設計上の理由がない限り、データ隠蔽の原則に従って、ほかのクラスから直接変更されたりしないように**private**で修飾するのが一般的です（**選択肢F**）。

サブクラスのコンストラクタ内でのスーパークラスのコンストラクタ呼び出しに関する問題です。

コンストラクタは、インスタンスを利用する前の準備をする特別なメソッドです。2つのクラスに継承関係が成り立つとき、サブクラスのインスタンスを準備するより前に、スーパークラスのインスタンスの準備を整えなければいけません。そのため、スーパークラスのコンストラクタを実行したあとに、サブクラスのコンストラクタを実行する必要があります。これを実現するために、サブクラスのコンストラクタの先頭行では、スーパークラスのコンストラクタ呼び出しをしなければいけないことが言語仕様で決められています。

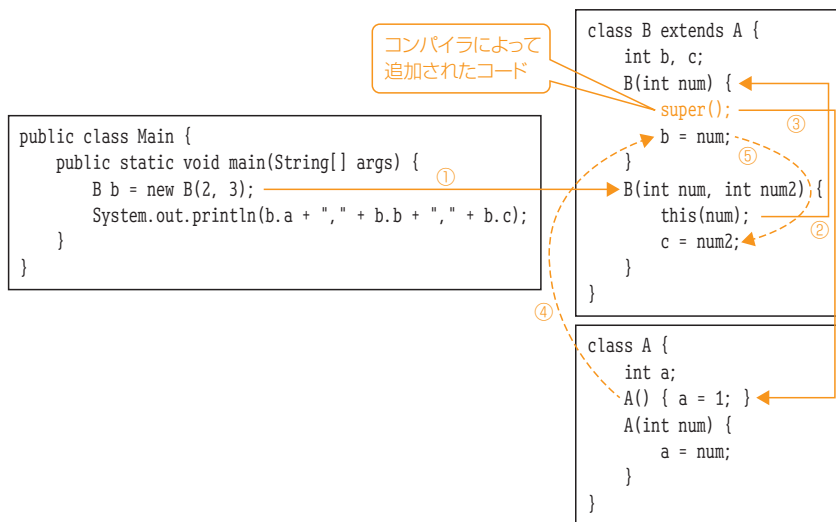
設問のコードでは、Mainクラスの3行目で、int型の引数を2つ受け取るコンストラクタを使ってBクラスのインスタンスを生成しています。このコンストラクタでは、**this**を使って別のコンストラクタを呼び出しています。サブクラスのコンストラクタブロック内でスーパークラスのコンストラクタ呼び出しよりも前に記述できるのは、オーバーロードした別のコンストラクタを呼び出すコードです。オーバーロードした別のコンストラクタの先頭行では、スーパークラスのコンストラクタを呼び出す必要があります。

プログラマーがスーパークラスのコンストラクタを呼び出すコードを記述しなかった場合、コンパイラによって自動的に追加されます。そのため、コンパイル後のコンストラクタの定義は次のようになります。

例 コンパイラによって自動的にコードが追加される

```
B(int num) {  
    super();    ← コンパイラによって追加されたコード  
    b = num;  
}  
B(int num, int num2) {  
    this(num);  
    c = num2;  
}
```

※次ページに続く



BクラスのスーパークラスであるAクラスの引数なしのコンストラクタでは、変数aに1を代入しています。また、Bクラスのint型の引数を1つだけ受け取るコンストラクタでは変数bに2を、引数を2つ受け取るコンストラクタでは変数cに3を代入しています。そのため、コンソールには、「1,2,3」が表示されます。以上のことから、選択肢**B**が正解です。

22. A

→ P15

アクセス修飾子に関する問題です。

privateで修飾されたメソッドは、継承関係の有無にかかわらず、同じクラスからしかアクセスできません。また、同じパッケージに属していても、privateメソッドにはアクセスできません。以上のことから、選択肢**A**が正解で、選択肢BとCは誤りです。

サブクラスからスーパークラスのメンバにはアクセスができますが、スーパークラスからサブクラスのメンバにはアクセスできません。よって、選択肢Dはアクセス修飾子の種類にかかわらず誤りです。

23. B

→ P16

staticメソッドの特徴に関する問題です。

staticメソッドは、インスタンスを生成しなくても使えるメソッドです。一方、staticが付いていないメソッド（インスタンスメソッド）は、インスタンスを

生成しなければ使えません。インスタンスの生成後に、同一クラス内のstaticメソッド（インスタンスを生成しなくても呼び出し可能）から、staticが付いていないメソッドを呼び出してしまうと、インスタンスが存在しないのに、staticが付いていないメソッドが呼び出されてしまうという矛盾が生じてしまいます。そのためJavaでは、staticメソッドからはstaticメソッドにしかアクセスできないというルールがあります。

設問のコードでは、staticメソッドであるmainメソッドからアクセスするために、halloメソッドはstaticで修飾されている必要があります。したがって、選択肢Bが正解です。

24. D

→ P16

アクセス制御と継承に関する問題です。

Aクラスのprivateなnumフィールドにアクセスしているのは、同じクラスに定義されているprintメソッドです。privateなフィールドは異なるクラスからはアクセスできませんが、同じクラスのメソッドからはアクセスできます。よって、選択肢Aは誤りです。

サブクラスは、スーパークラスに定義されているprivate以外のフィールドやメソッドを引き継ぎます。そのため、設問のAクラスを継承したBクラスはnumフィールドを引き継ぎませんが、printメソッドは引き継ぎます。Bクラスの4行目に記述されているprintメソッドの呼び出しは正しいコードです。よって、選択肢Bは誤りです。

2つのクラスが継承関係にあるとき、サブクラスはスーパークラスのprivateでないメンバにアクセスできます。設問のコードであれば、Aクラスを継承したBクラスからは、numフィールドにはアクセスできませんが、printメソッドにはアクセスできます。

以上のことから、設問のコードは正常に動作します。したがって、選択肢Dが正解です。

25. B

→ P17

オーバーロードした別のコンストラクタを呼び出す方法に関する問題です。

オーバーロードした別のコンストラクタを呼び出すにはthisを使います。選択肢Eのようにメソッド名でコンストラクタ呼び出しはできないので注意してください。また、選択肢Dはインスタンス自身を表す特別な変数thisを使ってメソッド呼び出しの形式でコンストラクタ呼び出しをしています。このような方法でコンストラクタは呼び出せません。よって、選択肢DとEは誤りです。

オーバーロードした別のコンストラクタを呼び出すコードは、必ずコンストラクタブロック内の先頭行になければいけません。そのため、選択肢Aのようにほかの処理を先に呼び出すことはできません。

オーバーロードした別のコンストラクタを呼び出すときに、フィールドの値を引数として使うことはできません。そのため、選択肢Cはコンパイルエラーになります。

以上のことから、選択肢Bが正解です。

26. A

→ P18

==演算子の比較対象に関する問題です。

==演算子で比較するのは変数の中身です。オブジェクト型変数の場合、変数の中には参照、つまりインスタンスへのリンクが入っています。そのため、==演算子では、2つの変数の参照先が同じであるかどうかを比較することになります。

フィールドは、明示的に初期化されていなくても、インスタンス生成時にデフォルト値で初期化されます。設問のSampleクラスのvalフィールドは、オブジェクト型の一種であるString型なので、オブジェクト型変数のデフォルト値であるnullで初期化されます。

Mainクラスの4行目のif文では、文字数0のStringオブジェクトである空白文字列を使っています。このif文の条件式では、Sampleクラスのインスタンスのvalフィールドの値（null）と、空白文字列であるStringインスタンスへの参照先が同じであるかどうか比較しています。文字列リテラルは、プログラムの実行時にStringのインスタンスとして自動的に生成されるため、参照先がnullになることはありません。よって、このif文の条件式はfalseを返し、valフィールドの値はデフォルト値であるnullのままということになります。以上のことから、選択肢Aが正解です。

27. A

→ P18

アクセス修飾子に関する問題です。

アクセス修飾子**private**は、同一クラス内でしかアクセスできないものに付ける修飾子です。抽象クラスやインタフェースは、継承されたり実装されたりすることを前提としています。抽象メソッドは、サブクラスでオーバーライドすることを前提としています。このため、これらはprivateで修飾できません。以上のことから、選択肢Aが正解です。

コマンドライン引数に関する問題です。

コマンドライン引数は、javaコマンド実行時に渡すことができる引数のことです。渡されたコマンドライン引数の値は、String配列に格納され、mainメソッドの引数として参照が渡されます。

設問のjavaコマンドでは、実行するクラス名に続けて「test」というコマンドライン引数が渡されています。そのため、mainメソッドの引数として渡されるString配列は1つだけ要素を持っていることになります。

設問のコードでは、添字0と1の値をコンソールに出力しようとしていますが、前述のとおり、このString配列は1つしか要素を持っていません。そのため、このコードは実行時に配列の要素外アクセスを表す例外（ArrayIndexOutOfBoundsException）がスローされます。以上のことから、選択肢Dが正解です。

インタフェースの実現と型変換に関する問題です。

インタフェースを実現（implements）したクラスは、インタフェースに宣言されているすべてのメソッドを実装しなければいけません。

設問のコードでは、インタフェースAを定義し、それを実現したBとCという2つのクラスを定義しています。どちらのクラスもインタフェースに宣言したメソッドを正しく実装しています。なお、変数名がAとBとCの各クラスで異なりますが、メソッドのシグニチャが同じであれば、変数名の違いは問題にはなりません。

Mainクラスの3行目では、BクラスとCクラスのインスタンスを生成し、2つのインスタンスへの参照を持つ配列を作っています。その後、配列の1つ目の要素、つまりBのインスタンスのtestメソッドを呼び出しています（4行目）。このとき、引数には3と2が渡されているため、Bのtestメソッドからは「 $(3 * 2) / 2$ 」の結果である3が戻されます。

Mainクラスの5行目では、配列の2つ目の要素、つまりCのインスタンスのtestメソッドを呼び出しています。このときも引数には3と2が渡されているため、Cのtestメソッドでは、まず式「 $3.2 * (3 * 2)$ 」が実行され、その結果、19.2がint型に変換されることで19（小数点以下切り捨て）が戻されます。

以上のことから、コンソールには「3 19」が表示されます。よって、選択肢Dが正解です。

30. D

→ P21

キャストに関する問題です。明示的にキャスト式を記述することで、コンパイラに「互換性の問題はない」ことを保証します。

設問のコードのMainクラスでは、Shapeクラスのインスタンスを生成し、Shape型の変数にその参照を代入しています（3行目）。4行目では、ShapeクラスのサブクラスであるTriangleクラス型の変数を用意し、Shape型変数を持っている参照を代入しようとしています。このとき、ShapeクラスはTriangleクラスだけが持っている定義（サブクラスとしての差分）を持っていないため、コンパイラは型変換できないと判断します。しかし、キャスト式を記述して型変換が安全にできることを明示しているため、このコードではコンパイルエラーは発生しません。

しかし、実際に実行しようすると、ShapeのインスタンスはTriangleクラスだけが持っている定義（サブクラスとしての差分）を持っていないため、参照先にあるインスタンスをTriangle型で扱うことができません。そのため、実行時に例外ClassCastExceptionがスローされます。以上のことから、選択肢Dが正解です。

31. A

→ P21

オーバーロードに関する問題です。

オーバーロードとは、引数が異なる同じ名前のメソッドを複数定義する「多重定義」のことです。JVMは、実行するメソッドの判別に、メソッド名と引数（数、型、順番）からなるシグニチャを使います。戻り値型はシグニチャには含まれないことに注意しましょう。

- A. 同じメソッド名で、引数の数が異なります。
- B. メソッド名と引数が同じで戻り値型が異なります。よって、これはオーバーロードしたメソッドではなく、「同名のメソッドが2つある」としてコンパイルエラーになります。
- C. メソッド名が異なります。よって、オーバーロードではありません。
- D. 戻り値型が定義されていません。よって、コンパイルエラーになります。

したがって、選択肢Aが正解です。

ソースファイルの規則に関する問題です。

パッケージに属するクラスを定義するときには、**パッケージ宣言**をソースファイルの先頭行に記述します。パッケージ宣言よりも先にコメント以外の何らかのコードを記述すると、コンパイルエラーになります（選択肢B）。

クラスは必ず何らかのパッケージに属さなければいけません。たとえばパッケージ宣言を省略した場合でも、デフォルトパッケージ（無名パッケージ）に属します。クラスがデフォルトパッケージに属する場合は、パッケージ宣言を記述しません（選択肢A）。

1つのソースファイル内には、クラスやインタフェース、列挙型を複数定義できます。ただし、publicなクラス、インタフェース、列挙型は、1つのファイルに1つしか記述できません。ほかのアクセス修飾子を持つクラスであれば、定義できるクラスやインタフェース、列挙型の数に制限はありません（選択肢C）。

一方、選択肢Fは、finalクラスは1つのソースファイルに1つしか定義できないとしています。finalクラスは継承を制限し、サブクラスを定義できないクラスですが、前述のとおり、publicでなければ1つのソースファイルに定義できる数に制限はありません。

インポート宣言は、完全修飾クラス名での記述を省略してクラス名だけで記述できるようにするためのものです。Javaのプログラムでは、標準クラスライブラリをはじめ、多くのクラスを利用します。その際、完全修飾クラス名で記述すると可読性が著しく下がるため、利用するクラスごとにインポート宣言を記述するのが一般的です。アスタリスク「*」を使ってクラス名を省略することも可能ですが、パッケージが異なれば、パッケージごとにインポート宣言を記述します（選択肢D）。

インポート宣言は、パッケージ宣言のあと、クラス宣言の前に記述しなければいけません（選択肢E）。

ソースファイルの名前は、「クラス名.java」でなければいけません（選択肢G）。なお、1つのソースファイルに複数のクラスを宣言した場合は、**publicなクラスのクラス名**と同じファイル名+拡張子（.java）としなければいけません。

for文の条件式に関する問題です。

for文のカッコ「()」の中には、初期化式、条件式、反復式の3つを記述します。いずれも省略可能です。なお、条件式を省略すると無限ループになるため、処理の中でbreakを使ってループを抜ける処理を記述するなど、プログラムの制御に注意してください。

設問のコードの条件では、1から表示が始まらなければいけません。しかし、初期化式で変数*i*の値は0で初期化されているため、そのままでは0から表示されることになります。そのため、繰り返し処理を実行する前に値を1増やさなければいけません。for文は、初期化式→条件式→繰り返し処理→反復式の順に動作するため、1から表示するために、条件式で値を1増やす必要があります。

選択肢CとDは、条件式で変数*i*の値が5よりも小さいかどうかを判定しているだけで、1増やす処理をしていません。そのため、これらのコードでは表示が0から開始してしまいます。よって、選択肢CとDは誤りです。

設問のコードの条件では、4までの数値を表示しなければいけません。そこで、選択肢AとBの**インクリメント演算子**の前置と後置の違いに着目します。インクリメント演算子が**前置**の場合は、変数の値を1増やしてからその値が5よりも小さいかを比較します。そのため、1から表示が始まって、4を表示したあと、条件式で値を1増やすことで条件式がfalseを戻すためにfor文を抜けます。以上のことから、選択肢Aが正解です。

選択肢Bはインクリメント演算子が後置されています。**後置**の場合は変数の値をコピーしてから変数の値を1増やし、コピーした値を戻します。そのため、初回の条件式判定では、変数*i*の値(0)をコピーし、続いて変数の値を1増やしたのち(0→1)、コピーした値(0)を使って「0 < 5」という条件式として評価します。したがって、コンソールへの表示は1から始まります。その後、繰り返し処理が継続して変数*i*の値が4になったときに、その値(4)をコピーし、続いて変数の値を1増やしたのち(4→5)、コピーした値(4)を使って「4 < 5」という条件式を評価します。この式はtrueを戻すため、コンソールには変数*i*の値である5が表示され、その後の条件式でfor文を抜けます。以上のことから、選択肢Bでは1から5までの数値が表示されることになります。よって、選択肢Bは誤りです。

ifを使った分岐処理に関する問題です。

条件によって処理を分岐するには、**if文**を使います。**if-else if**を使えば、複数の条件を指定することができます。設問のコードでは、**if-else if-else文**を使い、2つの条件とその他の場合で分岐しています。

1つ目の条件式では、変数aの値が10でなければtrueを戻します。しかし、変数aの値は10であるため、この条件式の結果はfalseとなります。よって、コンソールにAが表示されることはありません。

2つ目の条件式では、変数aの値が変数bよりも小さければtrueを戻します。変数aの値は10、変数bの値は20であるため、この式はtrueを戻します。そのため、コンソールにはBが表示されます。よって、選択肢Bが正解で、選択肢Cは誤りです。

switch文の場合は、caseラベルに対応した処理でbreak文を省略すると、一致したcaseラベル以降のすべての処理を実行するといったことができますが、if文ではできません。そのため、選択肢DのようにB、Cの順に表示されることはありません。よって、選択肢Dも誤りです。

抽象メソッドとオーバーライドに関する問題です。

抽象クラスに定義した**抽象メソッド**は、そのクラスを継承した具象クラスがメソッドをオーバーライドしなければいけません。抽象メソッドは**abstract**で修飾しますが、オーバーライドした具象メソッドではabstractを付けてはいけません。そのため、選択肢Cは誤りです。

メソッドをオーバーライドするときは、元のメソッドと同じか、それよりも緩いアクセス修飾子を付けなければいけません。設問のtestメソッドはpublicであるため、publicかそれよりも緩いアクセス修飾子にしなければいけません。選択肢Bは、アクセス修飾子なしのデフォルトが適用され、パッケージ内からしかアクセスができない指定になっています。よって、選択肢Bは誤りです。

オーバーライドはメソッドの再定義であるため、シグニチャは元のメソッドの定義と一致させなければいけません。Sampleクラスのtestメソッドは引数を受け取らない定義になっていますが、選択肢DのtestメソッドはString型の引数を受け取ります。これではメソッドのオーバーライドではなく、オーバー

ロードとして扱われます。よって、選択肢Dも誤りです。

以上のことから、選択肢Aが正解です。

36. D

→ P24

デフォルトコンストラクタに関する問題です。

プログラマーがコンストラクタを定義しなかった場合に、コンパイラが自動的に追加するコンストラクタが**デフォルトコンストラクタ**です。デフォルトコンストラクタは、引数なしで定義されます。プログラマーがコンストラクタを定義した場合は、デフォルトコンストラクタは追加されることはありません。

設問のSampleクラスでは、String型の引数を受け取るコンストラクタを定義しているため、デフォルトコンストラクタは追加されません。しかし、Mainクラスの3行目では引数なしのコンストラクタを使ってインスタンスを生成しようとしています。このような引数なしのコンストラクタはSampleクラスに存在しないため、このコードはコンパイルエラーになります。したがって、選択肢Dが正解です。

Sampleクラスに引数なしのコンストラクタの定義を追加すれば、Mainクラスはコンパイル可能となり、実行すると「test」「test」と表示されます。

37. D

→ P24

カプセル化、データ隠蔽に関する問題です。

カプセル化したあとに、値が不用意に変更されないようにするためには、データを隠蔽する必要があります。Javaでは、データを隠蔽するためにはフィールドのアクセス修飾子を**private**にします。よって、選択肢AとBは誤りです。publicで修飾すると、すべてのクラスからアクセスできるようになり、いつどのタイミングでどのような値に変更されたかを管理できなくなってしまいます。

abstractは、抽象クラスや抽象メソッドの宣言のための修飾子です。よって、選択肢Eのようにフィールドをabstractで修飾することはできません。

カプセル化とは、インスタンス（オブジェクト）単位で関係するデータと、それを使う処理をまとめることです。staticは、クラス単位で共通の変数を用意するもので、カプセル化の概念とは関係がありません。よって、選択肢Cは誤りです。

以上のことから、選択肢Dが正解です。

38. D

→ P25

配列型変数に関する問題です。

配列型変数は、配列オブジェクトへの参照を扱うための変数です。Javaでは、配列は複数の値を一度に扱うオブジェクトの一種であるという点に注意しましょう。

設問のコードでは、「2, 4, 6, 8」または「1, 3, 5, 7, 9」という値を持った2つの配列を作り、それぞれarrayとarray2という変数にその配列への参照を代入しています。その後、array2=array1に代入しているため、この2つの変数はどちらも同じ配列（4行目で生成した配列）への参照を持っていることとなります。

このため、コンソールには1, 3, 5, 7, 9の順に表示されます。よって、選択肢**D**が正解です。

39. C

→ P25

ローカル変数のスコープに関する問題です。

ローカル変数とはメソッド内に定義した変数のことで、メソッド内で宣言した箇所以降で利用できます。

設問のコードでは、mainメソッドで変数xを宣言しています。この変数はmainメソッドの中だけで有効なローカル変数です。4行目ではtestメソッドを呼び出すときに変数xを渡していますが、これはあくまでも変数xの中の値（10）をコピーして渡すだけであって、変数そのものを渡しているわけではありません。そのため、testメソッド内で宣言していない変数xを使うことはできません。以上のことから、このコードはコンパイルエラーになります。よって、選択肢**C**が正解です。

40. D

→ P26

do-while文を使った繰り返し処理に関する問題です。

設問では0から2までの値を順に表示しなければいけません。選択肢AやBのようにwhile文の条件式の中でインクリメントしてしまうと、繰り返し処理の前に条件式が処理され、1から表示が始まってしまいます。よって、選択肢AとBは誤りです。

do-while文は、まず繰り返し処理を実行してから条件式を判定する、後判定

第9章

の繰り返し構文です。そのためインクリメントは、最初に0が表示し終わったあとに実行されます。

選択肢Cは、doの直後に条件式を記述しています。このコードは構文の誤りとしてコンパイルエラーになります。以上のことから、選択肢Cは誤りで、選択肢Dが正解となります。

41. C

→ P26

for文による繰り返し処理の制御に関する問題です。

for文は、初期化式、条件式、反復式の3つで構成されていますが、設問のコードでは初期化式と反復式の2つが空欄となっています。

配列の要素には1から5までの数値が入っているため、1つ目の要素から順に表示していけば設問の条件に合致します。添字は0から始まるため、初期化式では変数iの値を0にしなければいけません。よって、選択肢BとDは誤りです。

次に、反復式は条件式を評価する前に行う処理を記述します。設問のコードでは、繰り返し処理の中でコンソールに値を表示したあと、変数iの値をインクリメントしているため、反復式でインクリメントを行うと、値が2ずつ増えることになります。条件に合致させるには、反復式では何も処理する必要がありません。よって、選択肢Cが正解です。

42. A

→ P27

nullリテラルに関する問題です。

nullリテラルとは、参照型変数がインスタンスへの参照を持たないことを表すための値です。

設問のコード3行目では、String型変数strがnullリテラルで初期化されているため、参照を保持していません。ifブロックの条件式では、変数strの値がnullであるかを評価しています。条件式の結果はtrueを戻すため、コンソールには「if」と表示されます。よって、選択肢Aが正解です。

7行目のelse ifブロックの条件式のようにダブルクォーテーション「"」で囲むと、nullリテラルではなく、nullという文字列として扱われます。よって、選択肢Bは誤りです。

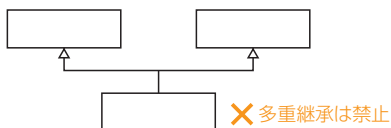
43. B、D

→ P27

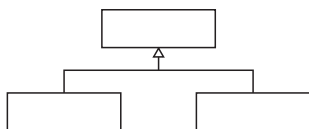
継承に関する問題です。

継承は、あるクラスを拡張した新しいクラスを定義することです。拡張したクラスは、拡張の元となったクラスの特徴を引き継ぎます。ただし、すべてを引き継ぐわけではありません。privateで修飾されたフィールドやメソッド、コンストラクタは引き継ぎません（選択肢C）。

Javaでは、複数のクラスを一度に継承する多重継承は許可されていません。継承は、たいへん便利な機能である一方で、使い方を間違えると変更にも弱い設計につながる可能性があるからです（選択肢A）。



このようにJavaでは多重継承が禁止されている一方で、1つのクラスを拡張した複数のクラスを定義することは可能です（選択肢B）。



継承したクラス（サブクラス）をさらに拡張したクラスを定義することは可能です（選択肢D）。

44. B、D

→ P27

修飾子に関する問題です。

abstractは、宣言だけで具体的な処理内容を持たない抽象メソッド、もしくはその具体的な処理内容を持たないメソッドを持つ抽象クラスの定義にだけ付けることができる修飾子です。よって、変数やパッケージをabstractで修飾することはできません。

以上のことから、選択肢BとDが正解です。

第9章

staticメソッドとインスタンスメソッドの違いに関する問題です。

同一クラスにおいて、staticメソッドからは、staticフィールドやstaticメソッドにしかアクセスできません。一方、インスタンスメソッドからは、staticの有無にかかわらずフィールドやメソッドにアクセスできます。

設問のSampleクラスのgetNumメソッドはstaticメソッドであるため、staticなメンバにしかアクセスできません。このメソッドがアクセスしているフィールドnumはstaticで修飾されているため、このコードに問題はありません。

また、同じクラスのtestメソッドからgetNumメソッドが呼ばれていますが、前述のとおりインスタンスメソッドからstaticメソッドにアクセスすることは可能です。よって、このコードも問題がなく、Sampleクラスは正常にコンパイルできます（選択肢C）。

Mainクラスのmainメソッドでは、Sampleクラスのインスタンスを生成し、インスタンスへの参照を使ってstaticメソッドとインスタンスメソッドを呼び出しています。しかし、このようなインスタンスへの参照を使ったstaticメソッドの呼び出しは、コンパイル時に次のようにクラス名を使った形に変更されます。

コンパイル前 `s.getNum()` → コンパイル後 `Sample.getNum()`

そのため、staticメソッドをインスタンスへの参照を使って呼び出しても問題ははありません。よって、Mainクラスも正常にコンパイルできます（選択肢D）。

staticフィールドもインスタンスフィールドと同様に、**デフォルト値で初期化**されます。int型のデフォルト値は0であるため、Sampleクラスのnumフィールドも0で初期化されます。

mainメソッドでは、まずSampleクラスのインスタンスのtestメソッドを呼び出しています。このコードでは、さらにgetNumメソッドを呼び出しており、その中でnumフィールドの値を1増やしてから戻します。そのため、このtestメソッドを実行した結果は1が戻されます。

次にSampleクラスのgetNumメソッドを呼び出しています。このコードは前述のとおり、numフィールドの値を1増やしてから戻します。numフィールドの値は先ほどのtestメソッドの呼び出し時に1になっているため、さらに1増えて2が戻されます。

その結果、変数resultには1+2の結果である3が代入されることになります。以上のことから、コンソールには3が表示されます。よって、選択肢Aが正解です。

46. D

→ P29

インタフェースを使ったポリモーフィズムに関する問題です。

ポリモーフィズムが成り立つのは、継承関係（extends）または実現関係（implements）のどちらかの関係にあるときだけです。

設問のコードでは、Sampleインタフェースを定義し、mainメソッドではそのインタフェース型で扱う配列を作っています。また、その配列の要素をTestクラスとExamクラスのインスタンスへの参照で初期化しています。

TestクラスはSampleインタフェースを実現しているため、Sample型で扱う配列の要素として使用可能です。一方、ExamクラスはSampleインタフェースとは無関係であるため、Sample型で扱う配列の要素として使うことはできません。そのため、Mainクラスでコンパイルエラーが発生します。よって、選択肢Dが正解です。

47. B、C

→ P30

インタフェースのメンバに関する問題です。

インタフェースには、ほかのクラスに公開するためのフィールドとメソッドを宣言できます。よって、**public**以外のアクセス修飾子で修飾することはできません。選択肢Dのようにアクセス修飾子を記述しなかった場合は、コンパイル時にpublicが付加されます。選択肢Cのように明示的に非公開（private）と記述した場合には、コンパイラによって変更されることはなく、コンパイルエラーが発生します。よって、選択肢Cはインタフェースには定義できません。

abstractで修飾できるのは、**抽象メソッド**もしくは**抽象クラス**だけです。インタフェースに宣言できるメソッドは、抽象メソッドだけです。そのためインタフェースに宣言するメソッド宣言でabstractが省略されていても、コンパイラによって自動的に修飾されます。よって、選択肢DやFは、インタフェースで宣言できます。また、選択肢Eのように明示的に記述することも可能です。フィールドはabstractで修飾できません。よって、選択肢Bのコードはインタフェースに定義できません。

選択肢Aのようにフィールドを宣言すると、**static**と**final**キーワードが自動的

に付加され、「`public static final String a = "A"`」という定数になります。インタ フェースには変数は定義できませんが、値が変わらない**定数**であれば定義できます。よって、選択肢Aは誤りです。

48. B

→ P30

インクリメント演算子の実行タイミングとwhile文に関する問題です。

設問のコードでは、変数*i*の値が4で初期化され、条件式で1増えて5になっています。その後、コンソールにtestという文字を表示し、さらにインクリメントされて値が6になっています。このタイミングでwhile文を抜ければ設問の条件のとおり、testを1回だけ表示できることになります。

「test」と1回表示したあと、条件式に戻るときの変数*i*の値は6になっています。そのため、選択肢Bのように「6よりも小さければ」という条件式にすれば、while文を抜けます。以上のことから、**選択肢Bが正解**です。

49. C

→ P31

パッケージ宣言とインポート宣言に関する問題です。

パッケージ宣言は、ソースファイルの**先頭行**に記述しなければいけません。パッケージ宣言よりも前に記述できるのは**コメント**だけです。そのため、**選択肢BやDのようにパッケージ宣言の前にインポート宣言を記述することはできません**。

インポート宣言では、省略表記したいクラスの**完全修飾クラス名**で記述するか、クラス名の部分だけを**ワイルドカード**で記述するかのどちらかの書式で記述します。

設問の条件は、「com.sample.controllerパッケージに属するクラスを使う」であり、具体的なクラス名は記述されていません。そのため、「com.sample.controller.*」のようにワイルドカードで記述します。**選択肢Aのようにパッケージ名だけを記述しても、具体的にどのようなクラスをインポートするかわからないため、選択肢Aはコンパイルエラーとなります**。以上のことから、**選択肢Cが正解**です。

50. B

→ P31

for文に関する問題です。

初期値から値が増えていくことの多い**for文**ですが、設問では逆に値を減らす反復式を持つfor文を使っています。このfor文では、変数*i*の初期値5から始まり、1ずつ値を減らしていき、*i*の値が0になったタイミングでfor文を抜けます。つまり、*i*の値が5、4、3、2、1のときに、コンソールに「e」が表示されます。よって、eは5回表示されるため、選択肢**B**が正解となります。

51. A、D

→ P32

オーバーロードとオーバーライドの違いに関する問題です。オーバーロードとオーバーライドは混同しやすいので、しっかりと違いを覚えておきましょう。

オーバーロードは、メソッドの多重定義で、引数の異なる同じ名前のメソッドを複数定義することです。一方、**オーバーライド**は、サブクラスでスーパークラスのメソッドを再定義することです。メソッドの再定義であるため、メソッドのシグニチャを変更することはできません。

以上のことから、選択肢**A**と**D**が正解です。

52. D

→ P32

コンストラクタの特徴に関する問題です。

コンストラクタには次のような3つのルールがあります。

- ・ インスタンス生成時にしか呼び出せない
- ・ 戻り値型を記述できない
- ・ コンストラクタ名はクラス名と同じであること

この3つのルールを除けば、コンストラクタは通常のメソッドと変わりません。

コンストラクタは、インスタンスがなければ動作しないインスタンスメソッドの一種です。staticメソッドからインスタンスメソッドへはアクセスできませんが、その逆は可能です。そのため、引数を受け取るコンストラクタからstaticなnumフィールドにアクセスすることは可能です（選択肢A）。

前述のとおり、コンストラクタには戻り値型を記述できません。設問のように戻り値型をvoidと記述した場合には、コンストラクタではなく、通常のメソッドとして扱われます。このように戻り値型を記述するとコンストラクタ

第9章

として扱われないだけで、文法上の誤りではありません。そのため、コンパイルエラーは発生しません（選択肢B）。

ただし、通常のメソッドとして扱われている以上、オーバーロードした別のコンストラクタを`this()`で呼び出すことはできません。コンストラクタはインスタンス生成時にしか使えないため、通常のメソッドから呼び出すことができません（選択肢D）。

コンストラクタは`private`で修飾することができます。コンストラクタをオーバーロードして、外部から使えるコンストラクタと内部からしか使えないコンストラクタを定義することが可能です（選択肢C）。

53. B、E、F

→ P33

配列の宣言方法についての問題です。

3行目は、初期化子「`{}`」を使い、配列インスタンスの生成と要素の初期化を同時に行っているコードです。よって、コンパイルエラーは発生しません（選択肢A）。

4行目は、`int`型の変数に`int`型で扱う配列型変数の値を代入しています。配列型変数の中には、配列インスタンスへの参照が入っています。`int`型の変数には、参照を代入できません。よって、4行目でコンパイルエラーが発生します（選択肢B）。

5行目は、配列型変数を用意して3つの要素を持つ配列インスタンスを生成し、その参照を代入しているコードです。このコードでコンパイルエラーは起きません（選択肢C）。

6行目は、新しい配列インスタンスを生成し、宣言済みの配列型変数にその参照を代入しています。変数の参照先の配列インスタンスを変更しただけなので、コンパイルエラーは起きません（選択肢D）。

7行目は、配列インスタンスの生成をするためにカッコ「`()`」を使っていますが、角カッコ「`[]`」を使わなければいけません。よって、このコードはコンパイルエラーになります（選択肢E）。

8行目は、配列型変数を宣言しています。配列型変数は、配列インスタンスへの参照を持つだけで、配列そのものは作りません。そのため、変数宣言時に要素数を指定することはできません。よって、このコードはコンパイルエラーになります（選択肢F）。

54. C

→ P34

インスタンスの生成に関する問題です。

変数を宣言するだけではインスタンスは生成されません。**new**キーワードを使いコンストラクタを呼び出すことで、インスタンスが生成され、そのインスタンスへの参照が戻されます。

設問では、3行目と4行目でItemクラスのインスタンスを生成し、変数item1とitem2にそれぞれインスタンスへの参照を代入しています。5行目のitem3は変数を宣言し、**null**で初期化しているだけです。そのため、生成されるインスタンスは2つです。以上のことから、選択肢**C**が正解です。

55. A、D

→ P34

継承に関する問題です。

フィールドは、サブクラスでも同じ名前のものを定義することができます。よって、選択肢**A**は正解です。

コンストラクタは、継承で引き継がれることはありません。また、サブクラスでスーパークラスのコンストラクタを定義することもできません。サブクラスでスーパークラスのコンストラクタを定義すると、コンストラクタではなく、戻り値型を定義していないメソッドとして扱われ、コンパイルエラーとなります。よって、選択肢**B**と**C**は誤りです。

finalで修飾されたメソッドは、サブクラスでオーバーライドすることはできません。よって、選択肢**D**が正解で、選択肢**E**は誤りです。

56. B、D、F

→ P35

命名規則に関する問題です。

クラス名の1文字目以降に使えるのはUnicode文字、ドル記号「\$」とアンダースコア「_」です。また、2文字目以降は数字も使うことができます。選択肢**A**、**C**、**E**のようにシャープ記号「#」やパーセント記号「%」、ハイフン「-」をクラス名に使うことはできません。

以上のことから、選択肢**B**、**D**、**F**が正解です。

第9章

57. A、F、G

→ P35

メソッドのシグニチャに関する問題です。

メソッドの**シグニチャ**は、メソッド名、引数の型、数、順序によって構成されています（選択肢B、C、D、E）。修飾子や引数の名前、戻り値型は関係ありません（選択肢**A、F、G**）。

58. B

→ P35

ポリモーフィズムに関する問題です。

ポリモーフィズムでは、異なるインスタンスを同じ型で扱います。同じ型で扱っていても、実際に動作するインスタンスの型は異なるため、処理結果が異なるというのがポリモーフィズムの特徴です。

ポリモーフィズムが成り立つには、継承関係にあるか、あるいはインタフェースの実現関係にあることが条件です。ただし、継承関係にあるだけでは、前述の「処理結果が異なる」ことが実現できません。このように処理結果が異なることを実現するためには、継承した上で、スーパークラスで定義したメソッドをサブクラスでオーバーライドします。そのため、選択肢Aの「継承」よりも選択肢**B**の「オーバーライド」のほうが、ポリモーフィズムに関係のある用語として適切です。

オーバーロードは、メソッドの多重定義であり、ポリモーフィズムとは関係ありません（選択肢C）。また、インタフェースを継承したインタフェースを定義することもポリモーフィズムとは関係ありません（選択肢D）。

59. C

→ P36

インクリメント演算子に関する問題です。

設問のコードでは、変数*i*を宣言して2で初期化しています。その後、変数の値に2を加算代入しているため、この時点で変数*i*の値は4になります。この加算代入が終わった段階での式は「 $4 + (i++)$ 」になります。

インクリメント演算子を**後置**した場合、元の値（4）をコピーしたあとに変数の値を1増やし（5）、その後、先にコピーしておいた値（4）を戻します。そのため、上記の式は「 $4 + (4)$ 」に変更され、コンソールには8が表示されます。よって、選択肢**C**が正解です。

オーバーロードに関する問題です。

オーバーロードの条件は、メソッドの**シグニチャ**が異なることです。メソッドのシグニチャは、メソッド名、引数の型、数、順序で構成されています。引数の変数名はシグニチャに含まれないことに注意しましょう。

設問のSampleクラスには、testというメソッドが2つ定義されています。この2つのメソッドは名前が同じだけでなく、引数の型、数、順序も同じです。異なるのは戻り値型だけです。前述のとおり、引数の変数名はシグニチャに含まれないため、オーバーロードは成り立ちません。そのため、このコードは同じシグニチャのメソッドを2つ定義しているとして、コンパイルエラーが発生します。以上のことから、選択肢**C**が正解です。