# **問題１：Java言語の基礎知識**

Java言語について以下の設問に答えよ。

**＜設問1-1＞**

以下の文章はJava言語について述べたものである。空欄に入れるべき適切な語句を**解答欄1-1**に記入せよ。

Ａ．int型の数値をdouble型の変数に代入すると自動的に　　①　　型に変換される。

double型の数値をint型の変数に代入するには　　②　　が必要で数値の前に　　③　　を付ける。

このとき小数点以下の数値は　　④　　。

Ｂ．long型もdouble型もデータサイズは　　⑤　　ビットであるが、

扱える数値の範囲は　　⑥　　型の方が大きい。

ただし数値が大きくなると扱える値が飛び飛びになる。

Ｃ．配列の要素数は　　⑦　　で取得できる。

インデックスは　　⑧　　から始まるため、

　　⑨　　番目の要素は存在しない。

Ｄ．if文では　　⑩　　の値によって処理が分岐する。

　　⑩　　の値が　　⑪　　のとき処理Aを実行し、

　　⑫　　のとき処理Bを実行する。

if (　　⑩　　) {

（処理A）

} else {

（処理B）

}

Ｅ．while文では　　⑬　　の値が　　⑭　　の間、処理Aを繰り返す。

　　⑮　　を使うとwhile文を抜け出すことができる。

　　⑯　　を使うとループ処理の先頭に戻る。

while(　　⑬　　) {

（処理A）

}

Ｆ．for文では最初に1回だけ　　⑰　　の処理を実行する。

その後、　　⑱　　に書かれた条件式の値が　　⑲　　のとき、処理Dを実行したのち、

　　⑳　　の処理を実行してからループの先頭に戻る。

for ( A; B; C) {

（処理D）

}

**解答欄1-1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. Int 型 | 1. 型変換（キャスト） | 1. ( int ) | 1. 切り捨てられる | 1. 64 |
| 1. Double | 1. 配列名.length | 1. 0 | 1. 配列名.length | 1. 条件式 |
| 1. true | 1. False | 1. 条件式 | 1. true | ⑮　break |
| ⑯ continue | 1. A | ⑱　B | ⑲　true | ⑳　C |

**＜設問1-2＞**

以下の文章はクラスについて述べたものである。空欄に入れるべき適切な語句を**解答欄1-2**に記入せよ。

Ａ．メンバ（フィールドやメソッド）は外部に対して公開するのか非公開にするのか設定することができる。

　　①　　を指定すると外部に対して公開するためクラス外からでも使える。

　　②　　を指定すると外部に対し非公開となるためクラス内でしか使えない。

　　③　　を指定するとサブクラスに対しては公開となるためサブクラスからは使うことができる。

Ｂ．クラスを使う側に必要なメンバだけを公開しクラスの中でしか使わないメンバを隠すことを　　④　　という。

Ｃ．Javaではメソッド名が同じでも　　⑤　　の仕様が異なればメソッドを複数定義できる。

これをメソッドの　　⑥　　と呼ぶ。

Ｄ．フィールドやメソッドの宣言に　　⑦　　を付けると、

インスタンスを生成していなくても使える　　⑧　　メンバとなる。

クラス全体でひとつだけ作られ同じクラスであればどのインスタンスからでも参照できる。

クラス外から　　⑧　　メンバを使うには　　⑨　　と記述する。

Ｅ．フィールドやメソッドの宣言に　　⑩　　を付けないと、　　⑪　　メンバとなる。

　　⑪　　メンバはインスタンスごとに作られるのでインスタンスを生成しないと使えない。

クラス外から　　⑪　　メンバを使うには　　⑫　　と記述する。

Ｅ．クラスにはインスタンス生成時に自動的に呼び出される　　⑬　　と呼ばれるメソッドがある。

　　⑬　　に　　⑭　　は記述せず、　　⑮　　はクラス名と同じにする。

Ｆ．Javaでは　　⑯　　でインスタンスの生成を行う。

Ｇ．Javaのデータ型は　　⑰　　（int型やdouble型など）と　　⑱　　（配列やクラスなど）に分けられる。

　　⑰　　の変数にはそのデータ型の　　⑲　　が入る。

　　⑱　　の変数にはそのデータ型の　　⑳　　が入る。

**解答欄1-2**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. Public | 1. Private | 1. Protected | 1. カプセル化 | 1. 引数 |
| 1. オーバーロード | 1. Static | 1. クラス | 1. クラス名.メンバ名 | 1. Static |
| 1. インスタンス | 1. インスタンス名.メンバ名 | 1. コンストラクタ | 1. 戻り値 | 1. メソッド名 |
| 1. New | 1. 基本型 | 1. 参照型 | 1. 値 | 1. 参照 |

**＜設問1-3＞**

以下の文章はクラスの継承について述べたものである。空欄に入れるべき適切な語句を**解答欄1-3**に記入せよ。

Ａ．Javaではクラスを継承して新しいクラスを作ることができる。このとき継承元のクラスを　　①　　、

継承先のクラスを　　　②　　と呼ぶ。　　②　　には新たにフィールドやメソッドを追加することができる。

Ｂ．クラスを継承するにはクラス宣言で「　　③　　 スーパークラス名」と記述する。

Ｃ．スーパークラスと同じ仕様のメソッドをサブクラスに定義することができる。

これをメソッドの　　④　　と呼ぶ。

Ｄ．スーパークラス型の変数でサブクラスのインスタンスを指すことができる。

ただし使えるのは　　⑤　　のメンバのみとなる。

Ｅ．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すると、スーパークラス型の変数を使ってサブクラス側のメソッドを呼び出すことができるようになる。スーパークラス型の変数が指すサブクラスによって異なる動作をさせることができる。このように呼び出し方が同じでも異なる動作をすることを　　⑥　　と呼ぶ。

Ｆ．サブクラスでオーバーライドしたスーパークラスのメソッドを（サブクラス内から）呼び出すには　　⑦　　と記述する。

Ｇ．　　⑧　　とはプログラムコードを実装しないメソッドのことで、メソッドの宣言に　　⑨　　を、

プログラムコードを定義する中カッコの代わりに　　⑩　　を付ける。

Ｈ．　　⑪　　を持つクラスを　　⑫　　と呼び、classの前に　　⑬　　を付ける。

　　⑫　　型の　　⑭　　を作ることはできるが　　⑮　　を作ることはできない。

Ｉ．クラス宣言のclassを　　⑯　　と記述するとインターフェイスになる。

インターフェイスにするとすべてのメソッドが　　⑰　　になる。

インターフェイスを実装するにはクラス宣言で「　　⑱　　 インターフェイス名」と記述する。

Ｊ．継承できるクラスは　　⑲　　であるが、実装できるインターフェイスは　　⑳　　である。

**解答欄1-3**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1. スーパークラス | 1. サブククラス | 1. extends | 1. オーバーライド | 1. スーパークラス |
| 1. ポリモーフィズム | 1. Super.メソッド名 | 1. 抽象メソッド | 1. Abstract | 1. セミコロン（；） |
| 1. 抽象メソッド | 1. 抽象クラス | 1. Abstract | 1. 変数 | 1. インスタンス |
| 1. interface | 1. 抽象メソッド | 1. Implements | 1. ひとつ | 1. 複数 |

# **問題２：Java基本文法**

以下で指示された処理について**実行結果**を参考に**リスト2**の　　A　　の部分のプログラムコードを作成せよ。

**リスト2**

import java.util.Scanner;

public class J2Test13\_2 {

public static void main(String[] args) {

Scanner in = new Scanner(System.in);

System.out.print("0より大きい整数を入力してください＞");

int n = in.nextInt();

　　A

in.close();

}

}

1. 入力された数値の桁数を表示する処理を、while文を使って作成せよ。

**実行結果①（太字は入力箇所）「1234」を入力したとき**

0より大きい整数を入力してください＞**1234**

4桁です！

while(n / 1000 < 10){  
 System.*out*.println("4です");  
 break;  
}

1. 1から入力された数値までを加算して表示する処理を、for文を使って作成せよ。

**実行結果②（太字は入力箇所）「10」を入力したとき**

0より大きい整数を入力してください＞**10**

1から10まで加算すると55です！

int x = 0;  
for(int i = 1 ; i <= n; i++){  
 x+=i;  
}  
System.*out*.println("1から加数" + x );

1. 入力された数値を5で割ったとき、割り切れるかどうか表示する処理を、if～else文を使って作成せよ。

**実行結果③（太字は入力箇所）「100」を入力したとき 実行結果③（太字は入力箇所）「99」を入力したとき**

0より大きい整数を入力してください＞**99**

5で割り切れません！

0より大きい整数を入力してください＞**100**

5で割り切れます！

if( n % 5 == 0 ){  
 System.*out*.println("5割り切れます");  
}else{  
 System.*out*.println("5割り切れません");  
}

1. ③と同じ処理を、switch文を使って作成せよ。

**実行結果④（太字は入力箇所）「100」を入力したとき 実行結果④（太字は入力箇所）「99」を入力したとき**

0より大きい整数を入力してください＞**99**

5で割り切れません！

0より大きい整数を入力してください＞**100**

5で割り切れます！

switch (n % 5){  
 case 0 :  
 System.*out*.println("5で割り切れます");  
 break;  
 default:  
 System.*out*.println("5で割り切れません");  
 break;  
}

# **問題３：クラスメンバとインスタンスメンバ**

**リスト3-1**および**リスト3-2**は生成したSheepクラスのインスタンスの数を表示する処理である。**実行結果3**となるように**リスト3-1**および**リスト3-2**のコードを作成せよ。

**リスト3-1：Sheepクラス**

public class Sheep {

private 　　①　　 counter; // 生成した羊の数

private 　　②　　 num; // 羊番号（生成した順に1、2、3、・・・）

public 　　③　　 showCounter() {

System.out.println("羊は全部で" + 　　④　　 + "匹です！");

}

public 　　⑤　　 { // コンストラクタ

　　⑥　　 = ++　　⑦　　;

System.out.println(num + "番の羊がやってきた！");

}

}

**リスト3-2：Sheepの生成と生成した数の表示**

public class J2Test13\_3 {

public static void main(String[] args) {

　　⑧　　; // 羊の数を表示

　　⑨　　 s1 = 　　⑩　　; // s1に羊を1匹生成

　　⑧　　; // 羊の数を表示

　　⑪　　 sn = 　　⑫　　; // snに羊の配列（3匹分）を生成

for (int i = 0; 　　⑬　　; i++) {

　　⑭　　 = 　　⑮　　; // 配列に羊を生成

}

　　⑧　　; // 羊の数を表示

}

}

**実行結果3**

羊は全部で0匹です！

1番の羊がやってきた！

羊は全部で1匹です！

2番の羊がやってきた！

3番の羊がやってきた！

4番の羊がやってきた！

羊は全部で4匹です！

|  |  |  |
| --- | --- | --- |
| 1. Static Int | 1. Int | 1. Static void |
| 1. Counter | 1. Sheep | 1. Counter |
| 1. num | 1. Sheep.showCounter(); | 1. Sheep |
| 1. New Sheep( ); | 1. Sheep[ ] | 1. New Sheep[3]; |
| 1. I < Counter | 1. Sheep [ i ] | 1. New Sheep(); |

# **問題４：継承**

Monsterクラスを継承してFlyMonsterクラスとSwimMonsterクラスを定義する。

**＜設問4-1＞**

クラス図をもとに**リスト4-1、4-2、4-3**のコードを作成せよ。

|  |
| --- |
| **Monster** |
| # name |
| + Monster(name)  + intro() |

|  |
| --- |
| **FlyMonster** |
|  |
| + FlyMonster(name)  + intro() |

|  |
| --- |
| **SwimMonster** |
|  |
| + SwimMonster(name)  + intro() |

**リスト4-1：Monsterクラス**

public class Monster {

　　①　　 String name; // 名前

　　②　　 Monster(String name) {

　　③　　 = name;

System.out.println("Monster！");

}

　　④　　 void intro() {

System.out.println("ぼくの名前は" + name + "。");

}

}

**リスト4-2：FlyMonsterクラス リスト4-3：SwimMonsterクラス**

public class SwimMonster 　　⑤　　 {

public SwimMonster(String name) {

　　⑥　　;

System.out.println("SwimMonster！");

}

public void intro() {

　　⑦　　; // Monsterのintro

System.out.println("泳ぎが得意さ！");

}

}

public class FlyMonster 　　⑤　　 {

public FlyMonster(String name) {

　　⑥　　;

System.out.println("FlyMonster！");

}

public void intro() {

　　⑦　　; // Monsterのinro

System.out.println("空をとべるよ！");

}

}

|  |  |  |
| --- | --- | --- |
| 1. Protected | 1. Public | 1. This.name |
| 1. Public | 1. Extends Monster | 1. Super(name); |
| 1. Monster.Intro(); |  |  |

**＜設問4-2＞**

**リスト4-4**はFlyMonsterとSwimMonsterに自己紹介させる処理である。**リスト4-4**を入力し実行結果を確認せよ。

**リスト4-4：FlyMonsterとSwimMonsterの自己紹介**

public class J2Test13\_4 {

public static void main(String[] args) {

Monster fly = new FlyMonster("ムックル");

fly.intro();

Monster swim = new SwimMonster("コイキング");

swim.intro();

}

}

# **問題５：ポリモーフィズム**

|  |
| --- |
| <<interface>>  ***IFairy*** |
|  |
| *intro() : void* |

|  |
| --- |
| **Darkness** |
|  |
| intro() : void |

|  |
| --- |
| **Fire** |
|  |
| intro() : void |

|  |
| --- |
| **Light** |
|  |
| intro() : void |

Light・Darkness・Fireの各クラスはIFairyを実装（継承）している。

**＜設問5-1＞**

IFairyインターフェイスを作成せよ。

**＜設問5-2＞**

**実行結果5**のようになるようにLight・Darkness・Fireの各妖精クラスとJ2Test13\_5クラス（mainメソッド）を作成せよ。

**実行結果5（J2Test13\_5クラス）**

妖精を召喚して自己紹介させます！

誰を召喚しますか？（0：光の妖精、1：闇の妖精、2：炎の妖精、-1：やめる）＞**0**

わたしは光の妖精！この者に祝福を！！

誰を召喚しますか？（0：光の妖精、1：闇の妖精、2：炎の妖精、-1：やめる）＞**1**

わたしは闇の妖精だ！闇の力を思い知れ！！

誰を召喚しますか？（0：光の妖精、1：闇の妖精、2：炎の妖精、-1：やめる）＞**2**

わたしは炎の妖精さ！炎の力は気まぐれなのさ！！

誰を召喚しますか？（0：光の妖精、1：闇の妖精、2：炎の妖精、-1：やめる）＞**-1**

public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 System.*out*.println("妖精を召喚して自己紹介させます！");  
 while(true){  
 System.*out*.print("誰を召喚しますか？（0：光の妖精、1：闇の妖精、2：炎の妖精、-1：やめる）＞");  
 int n = in.nextInt();  
 if (n < 0) break;  
 IFairy fairy;  
 switch(n) {  
 case 0: fairy = new Light(); fairy.intro(); break;  
 case 1: fairy = new Darkness(); fairy.intro(); break;  
 case 2: fairy = new Fire(); fairy.intro(); break;  
 }  
 System.*out*.println();  
 }  
 }  
}

public class Light implements IFairy {  
 public void intro() {System.*out*.println("わたしは光の妖精！この者に祝福を！！");}  
}

public interface IFairy{

void intro();

}