

- ・暗記事項を参考にして予習ノートの作成
- ・予習範囲の熟読
- ・暗記事項の暗記
- ・教科書の演習問題を理解

上記を済ませてから練習問題に取り組んでください。
最終ページに解答を載せています。

予習範囲

6章です。

暗記事項

【暗記事項1】から【暗記事項13】までは、すべて、「A: 正しい」が正答となります。予習確認テストでは表現を変えて「B: 間違っている」が答えになることもあります。

ただ覚えるだけでなく、教科書中の当該事項を説明した文章を読んで理解につとめてください。

【暗記事項1】継承はクラス間が「上位概念-下位概念」にあるときに用いる

【暗記事項2】継承のことを IS-A 関係という

【暗記事項3】上位概念にあたるクラスのことを親クラスまたはスーパークラスという

【暗記事項4】下位概念にあたるクラスのことを子クラスまたはサブクラスという

【暗記事項5】子クラスでは、親クラスの属性(フィールド)や機能(メソッド)を継承する(引継ぐ)

【暗記事項6】子クラスのクラス定義の1行目では「アクセス修飾子 class (子クラスの)クラス名 extends (親クラスの)クラス名」を記述する

【暗記事項7】親クラスや「親クラスの親クラス(“おじいちゃんクラス”)」など祖先にあたるクラスで定義された、アクセス修飾子が private ではないメソッド・フィールドを子クラスから呼出し・参照することができる

【暗記事項8】暗記事項7に関連して、例え親クラスのメソッドが呼出し可能であっても、子クラスのクラス図では、子クラスのクラス定義で定義されたメソッド・フィールド・コンストラクタを記述する

【暗記事項9】祖先クラスで定義されたメソッドを定義しなおすことを「オーバーライド」という

【暗記事項10】暗記事項8、9に関連して、オーバーライドしたメソッドは子クラスのクラス図に記述する

【暗記事項11】クラス図では継承関係を表すために子クラスから親クラスに向けて矢印——▷ を引く

【暗記事項12】親クラスの参照型変数は子クラス(孫クラスなどの子孫クラスを含む)のオブジェクトを参照できる

【暗記事項13】各クラスは、一つの親クラスしか直接的に継承できない(おじいちゃんクラスは親クラスによって継承されている。子クラスは親クラスの子であって、おじいちゃんの子供ではない)

6.1 節 レストランの例

継承

クラス間に「概念- 概念」の関係があるときに用いる 1
この関係を 関係という 2

6.2 節 継承

継承

- ・ クラスの定義方法：
アクセス修飾子 class クラス名 親クラス名 { 6
 クラスの中身
 }
- ・ クラス, クラス: 3
 上位概念を表すクラス
- ・ クラス, クラス: 4
 下位概念を表すクラス．親クラスの属性（項目），機能を
 する（引き継ぐ） 5

(注) 各クラスは, の親クラスしか直接的に継承できない 13

オーバーライド

祖先クラスで定義されたメソッドを こと 9


オーバーライドしたメソッドは子クラスの に記述する 10

コンストラクタ（講義 No.11 の演習で習います，講義 No.11 が終わってから記入しても構いません）

- ・ デフォルトコンストラクタ
 コンストラクタが「 」定義されていない時、
 引数なしで何もしないコンストラクタが とされる。
- ・ 親クラスを参照する特別な参照型変数 **super**
 メソッドがオーバーライドされているが、子クラスで敢えて、親クラスのメソッドを
 呼びたい時がある。このとき、**super** を使ってメソッドを呼び出すことができる。
- ・ 親クラスのコンストラクタ呼出し
 コンストラクタの処理を記述する部分の一行目に限り、親クラスの参照である を用いて
 親クラスのコンストラクタを呼び出すことができる。
 子クラスのコンストラクタの一行目に、親クラスのコンストラクタ呼出し **super()** がない場合、
 Java 言語のコンパイラは の **super()** が として、解釈してくれる。

6.3 節 継承のクラス図

クラス図において継承を表すための矢印

- ・ クラスから クラスに向けて矢印を引く 11
 
- ・ 各クラスで定義されているフィールド・メソッドを記述する

6.4 節 ポリモーフィズムとメソッド定義の探し方

継承におけるポリモーフィズム

- ・ 親クラスの `method` は子クラスの `method` を参照できる [12](#)
- ・ 実行されるメソッドはオブジェクトの型によって変わる

メソッド定義の検索

1. 参照型変数が参照している `Class` の型 (クラス) を調べる
2. `(クラス名).java` というファイル調べる
3. シグネチャが合致するメソッドがあれば、それが呼び出しを行っているメソッドである
4. シグネチャが合致するメソッドがない場合、`Class` からメソッドを探す

練習問題

確認テストも、この練習問題と同じ方法で、解答してください。最終ページに答えを載せています。

解答が複数ある場合は、ハイフン で繋いで答えること。

例: A と B と C を解答したい場合、A-B-C と解答欄に記入する。答えがない場合は-1 と解答すること。

行番号を解答するとき、左詰め0 は取ること

例: 001 行目を解答するときは 1 を解答すること

`System.out.println()` (改行あり) は `main` メソッドにしかありません。

他の場所では、`System.out.print()` (改行なし) を使っています

行番号を振っていない空行に続いて、行番号を振りなおしているソースコードが続く場合は、別の `Java` ファイル (クラス) であることを示しています。

本練習問題 (前半) の答え方

以下のソースコードについて【練習 01】から【練習 12】を解答しなさい

```
001 public class A_InheritanceTest1 {
002     public void method() {
003         System.out.print("0");
004     }
005     public void swim() {
006         System.out.print("1");
007     }
008 }

011 public class B_InheritanceTest2 extends A_InheritanceTest1 {
012     public void swim() {
013         System.out.print("2");
014     }
015     public void run() {
016         System.out.print("3");
017     }
018 }

021 public class C_InheritanceTest3 extends B_InheritanceTest2 {
022     public void method() {
023         System.out.print("4");
024     }
025 }

031 public class TestDrive {
032     public static void main(String[] args) {
033         A_InheritanceTest1 it1 = new B_InheritanceTest2();
034         B_InheritanceTest2 it2 = new C_InheritanceTest3();
035
036         it1.method();
037         System.out.println("A");
038
039         it2.run();
040         it2.method();
041         System.out.println("B");
042     }
043 }
```

【練習 01】出力の一行目を答えなさい **0A**

【練習 02】出力の二行目を答えなさい **34B**

フィールド・コンストラクタ・メソッドの順に記述してください。

該当する行がなければ、-1 を記入してください。

- 【練習 03】 A_InheritanceTest1 のクラス図の一行目を答えなさい A_InheritanceTest1
- 【練習 04】 A_InheritanceTest1 のクラス図の二行目を答えなさい + method(): void
- 【練習 05】 A_InheritanceTest1 のクラス図の三行目を答えなさい + swim(): void
- 【練習 06】 B_InheritanceTest2 のクラス図の一行目を答えなさい B_InheritanceTest2
- 【練習 07】 B_InheritanceTest2 のクラス図の二行目を答えなさい + swim(): void
- 【練習 08】 B_InheritanceTest2 のクラス図の三行目を答えなさい + run(): void
- 【練習 09】 C_InheritanceTest3 のクラス図の一行目を答えなさい C_InheritanceTest3
- 【練習 10】 C_InheritanceTest3 のクラス図の二行目を答えなさい + method(): void
- 【練習 11】 C_InheritanceTest3 のクラス図の三行目を答えなさい -1



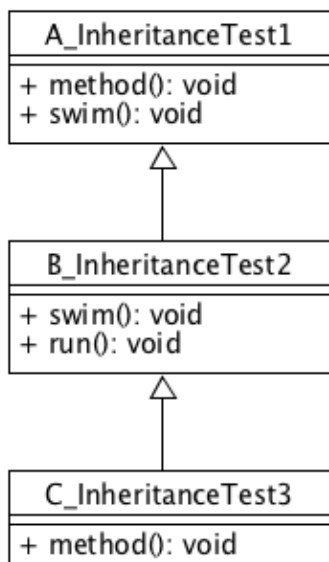
クラスの先頭の文字で、上の矢印を表す記号のどれかを囲みます。
 矢印の先が後ろに来るように答えること。

【練習 12】 クラス間の関係を表す矢印を答えなさい

B>A-C>B

TestDrive 以外のクラス間の関係を記述してください。複数の矢印があればハイフンでつないでください。
 並びは記号を除くアルファベットが辞書順になるようにしてください
 B>A と C>A だったら辞書順では B>A が先なので【練習 12】の解答例のように解答してください。

練習問題の解答のクラス図



次ページに続く

以下のソースコードについて【練習 13】から【練習 22】を解答しなさい

```
001 public class CalcScore {
002     public CalcScore() {
003         System.out.print("0");
004     }
005     public int calcScore(int midTestScore, int finalTestScore){
006         int score;
007         System.out.print("1");
008         if(midTestScore < 60 || finalTestScore < 60){
009             score = 0;
010         }else{
011             score = (midTestScore + finalTestScore)/2;
012         }
013         return score;
014     }
015 }

021 public class Score extends CalcScore {
022     public Score(){
023         super(); //super()は親クラス CalcScore のコンストラクタを呼び出す
024         System.out.print("2");
025     }
026     public String judgement(int midTestScore,int finalTestScore){
027         System.out.print("3");
028         int score = calcScore(midTestScore, finalTestScore);
029         if(score>=60){
030             return "この科目は合格です";
031         }else{
032             return "この科目は落単です";
033         }
034     }
035 }

041 public class ScoreTestDrive {
042     public static void main(String[] args) {
043         Score score = new Score();
044         String str = score.judgement(80, 60);
045         System.out.println("A");
046         System.out.println(str);
047         str = score.judgement(70, 50);
048         System.out.println(str);
049         str = score.judgement(50, 80);
050         System.out.println(str);
051     }
052 }
```

【練習 13】出力の一行目を答えなさい

【練習 14】出力の二行目を答えなさい

【練習 15】出力の三行目を答えなさい

【練習 16】CalcScore のクラス図の一行目を答えなさい

【練習 17】CalcScore のクラス図の二行目を答えなさい

【練習 18】CalcScore のクラス図の三行目を答えなさい

【練習 19】Score のクラス図の一行目を答えなさい

【練習 20】Score のクラス図の二行目を答えなさい

【練習 21】Score のクラス図の三行目を答えなさい

クラスの先頭の文字で、上の矢印を表す記号のどれかを囲みます。

矢印の先にあたるクラスが後ろに来るように答えること。

【練習 22】クラス間の関係を表す矢印を答えなさい



以下のソースコードについて【練習 23】から【練習 35】を解答しなさい

```
001 public class Vehicle { //Vehicle は抽象クラスの方が良い
002     protected String name;
003     public Vehicle(String name) {
004         this.name = name;
005         System.out.print("0");
006     }
007     public void emergencyStop() {
008         System.out.print("緊急停止します");
009     }
010     public void run() {
011         System.out.print(name+"は走ります");
012     }
013 }

021 public class Bus extends Vehicle {
022     public Bus(String name) {
023         super(name); //親クラスのコンストラクタ呼び出し
024         System.out.print("1");
025     }
026     public void run() {
027         System.out.print(name+"はぶーんと走ります");
028     }
029 }

041 public class Train extends Vehicle {
042     public Train(String name) {
043         super(name); //親クラスのコンストラクタ呼び出し
044         System.out.print("2");
045     }
046     public void run() {
047         System.out.print(name+"はがたんごとんと走ります");
048     }
049 }

061 public class VehicleTestDrive {
062     public static void main(String[] args) {
063         Vehicle vehicle;
064         for (int i = 0; i < 3; i++) {
065             if (i==0) vehicle = new Vehicle("乗り物");
066             else if (i==1) vehicle = new Bus("バス");
067             else vehicle = new Train("電車");
068
069             System.out.println("A");
070             vehicle.run();
071             vehicle.emergencyStop();
072             System.out.println("B");
073         }
074     }
075 }
```

【練習 23】出力の一行目を答えなさい

【練習 24】出力の二行目を答えなさい

【練習 25】出力の三行目を答えなさい

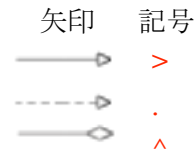
【練習 26】出力の四行目を答えなさい

【練習 27】出力の五行目を答えなさい

【練習 28】出力の六行目を答えなさい

【練習 29】Vehicle のクラス図の一行目を答えなさい

- 【練習 30】 Vehicle のクラス図の二行目を答えなさい
- 【練習 31】 Vehicle のクラス図の三行目を答えなさい
- 【練習 32】 Bus のクラス図の一行目を答えなさい
- 【練習 33】 Bus のクラス図の二行目を答えなさい
- 【練習 34】 Bus のクラス図の三行目を答えなさい
- 【練習 35】 クラス間の関係を表す矢印を答えなさい



以下のソースコードについて 【練習 36】 から 【練習 46】 を解答しなさい

```

001 public class Instrument {
002     private String name;
003     public Instrument(String name) {
004         this.name = name;
005         System.out.print("0");
006     }
007     public String getName() {
008         System.out.print("1");
009         return name;
010     }
011 }




021 public class Piano extends Instrument {
022     public Piano(String name) {
023         super(name);
024         System.out.print("2");
025     }
026     public void play() {
027         System.out.print("ポロロン");
028     }
029     public void tuning() {
030         System.out.print("調律します");
031     }
032 }

041 public class ElectricPiano extends Piano {
042     public ElectricPiano(String name) {
043         super(name);
044         System.out.print("3");
045     }
046     public void tuning() {
047         System.out.print("調律は要りません");
048     }
049 }

061 public class PianoTestDrive {
062     public static void main(String[] args) {
063         Piano piano = new Piano("スタインウェイ");
064         System.out.println("A");
065         System.out.print(piano.getName());
066         piano.play();
067         piano.tuning();
068         System.out.println("B");
069         piano = new ElectricPiano("ヤマハ");
070         System.out.println("C");
071         System.out.print(piano.getName());
072         piano.play();
073         piano.tuning();
074         System.out.println("D");
075     }
076 }

```


- 【練習 36】出力の一行目を答えなさい
- 【練習 37】出力の二行目を答えなさい
- 【練習 38】出力の三行目を答えなさい
- 【練習 39】出力の四行目を答えなさい
- 【練習 40】Instrument のクラス図の一行目を答えなさい
- 【練習 41】Instrument のクラス図の二行目を答えなさい
- 【練習 42】Instrument のクラス図の三行目を答えなさい
- 【練習 43】ElectricPiano のクラス図の一行目を答えなさい
- 【練習 44】ElectricPiano のクラス図の二行目を答えなさい
- 【練習 45】ElectricPiano のクラス図の三行目を答えなさい
- 【練習 46】クラス間の関係を表す矢印を答えなさい

矢印	記号
	>
	.
	^

以下のソースコードについて【練習 47】 から【練習 57】 を解答しなさい

```
001 public class CPU {
002     private String speed;
003     public CPU(String speed) {
004         this.speed = speed;
005         System.out.print("0");
006     }
007     public String run() {
008         System.out.print("1");
009         return speed + "実行します";
010     }
011 }




021 public class OperatingSystem {
022     private CPU cpu;
023     private String task;
024     public OperatingSystem(CPU cpu, String task) {
025         this.cpu = cpu;
026         this.task = task;
027     }
028     public void execute() {
029         System.out.print(task + "を" + cpu.run());
030     }
031 }

041 public class AMDProcessor extends CPU {
042     public AMDProcessor(String speed) {
043         super(speed);
044         System.out.print("2");
045     }
046     public String run() {
047         System.out.print("3");
048         return "AMD の CPU を使って" + super.run();
049     }
050     // super.run()は親クラスのメソッド run()を呼び出す
051 }

061 public class IntelProcessor extends CPU {
062     public IntelProcessor(String speed) {
063         super(speed);
064         System.out.print("4");
065     }
066     public String run() {
067         System.out.print("5");
068         return "Intel の CPU を使って" + super.run();
069     }
070 }

081 public class OperatingSystemTestDrive {
082     public static void main(String[] args) {
083         OperatingSystem os;
084         os = new OperatingSystem(new AMDProcessor("控えめに"), "計算処理");
085         System.out.println("A");
086         os.execute();
087         System.out.println("B");
088         os = new OperatingSystem(new IntelProcessor("速く"), "ゲーム");
089         System.out.println("C");
090         os.execute();
091         System.out.println("D");
092     }
093 }
```

- 【練習 47】出力の一行目を答えなさい
- 【練習 48】出力の二行目を答えなさい
- 【練習 49】出力の三行目を答えなさい
- 【練習 50】出力の四行目を答えなさい
- 【練習 51】CPU のクラス図の一行目を答えなさい
- 【練習 52】CPU のクラス図の二行目を答えなさい
- 【練習 53】CPU のクラス図の三行目を答えなさい
- 【練習 54】OperatingSystem のクラス図の一行目を答えなさい
- 【練習 55】OperatingSystem のクラス図の二行目を答えなさい
- 【練習 56】OperatingSystem のクラス図の三行目を答えなさい
- 【練習 57】クラス間の関係を表す矢印を答えなさい

矢印	記号
	>
	·
	^

以下のソースコードについて【練習 58】から【練習 73】を解答しなさい

```
001 public interface Technology {
002     public String getName();
003 }

011 public class CPU {
012     private String name;
013     protected Technology technology;
014     public CPU(String name, Technology technology) {
015         this.name = name;
016         this.technology = technology;
017         System.out.print("0");
018     }
019     public String run() {
020         System.out.print("1");
021         return name + "は" + technology.getName() + "で高速に実行します";
022     }
023 }

031 public class SMT implements Technology {
032     public SMT() {
033         System.out.print("2");
034     }
035     public String getName() {
036         System.out.print("3");
037         return "Simultaneous Multi-Threading";
038     }
039 }




051 public class V_TBMT implements Technology {
052     public V_TBMT() {
053         System.out.print("4");
054     }
055     public String getName() {
056         System.out.print("5");
057         return "Turbo Boost Max Technology";
058     }
059 }

071 public class AMDProcessor extends CPU {
072     public AMDProcessor() {
073         super("AMD Zen", new SMT());
074         System.out.print("6");
075     }
076 }

081 public class IntelProcessor extends CPU {
082     public IntelProcessor() {
083         super("Intel i7", new V_TBMT());
084         System.out.print("7");
085     }
086 }

091 public class CPUtestDrive {
092     public static void main(String[] args) {
093         CPU cpu = new AMDProcessor();
094         System.out.println("A");
095         System.out.println(cpu.run());
096         cpu = new IntelProcessor();
097         System.out.println("B");
098         System.out.println(cpu.run());
099     }
}
```

- 【練習 58】出力の一行目を答えなさい
- 【練習 59】出力の二行目を答えなさい
- 【練習 60】出力の三行目を答えなさい
- 【練習 61】出力の四行目を答えなさい
- 【練習 62】Technology のクラス図の一行目を答えなさい
- 【練習 63】Technology のクラス図の二行目を答えなさい
- 【練習 64】Technology のクラス図の三行目を答えなさい
- 【練習 65】CPU のクラス図の一行目を答えなさい
- 【練習 66】CPU のクラス図の二行目を答えなさい
- 【練習 67】CPU のクラス図の三行目を答えなさい
- 【練習 68】SMT のクラス図の一行目を答えなさい
- 【練習 69】SMT のクラス図の二行目を答えなさい
- 【練習 70】SMT のクラス図の三行目を答えなさい
- 【練習 71】AMDProcessor のクラス図の一行目を答えなさい
- 【練習 72】AMDProcessor のクラス図の二行目を答えなさい
- 【練習 73】クラス間の関係を表す矢印を答えなさい

矢印	記号
	>
	.
	^

以下のソースコードについて【練習 74】から【練習 84】を解答しなさい

```
001 public interface AttackWay {
002     public String attack();
003 }

011 public interface ThrowWay {
012     public String thrown();
013 }

021 public interface U_RunWay {
022     public String run();
023 }

031 public class InsufficientAttack implements AttackWay {
032     public InsufficientAttack() {
033         System.out.print("J");
034     }
035     public String attack() {
036         System.out.print("K");
037         return "蚊に刺されたのかと";
038     }
039 }

051 public class NormalAttack implements AttackWay {
052     public NormalAttack() {
053         System.out.print("L");
054     }
055     public String attack() {
056         System.out.print("M");
057         return "普通だな";
058     }
059 }

071 public class StrongAttack implements AttackWay {
072     public StrongAttack() {
073         System.out.print("N");
074     }
075     public String attack() {
076         System.out.print("O");
077         return "強い、圧倒的に強い";
078     }
079 }

091 public class HeavyThrow implements ThrowWay {
092     public HeavyThrow() {
093         System.out.print("F");
094     }
095     public String thrown() {
096         System.out.print("G");
097         return "おっも . . . ";
098     }
099 }
```

```

111 public class Q_NormalThrow implements ThrowWay {
112     public Q_NormalThrow() {
113         System.out.print("H");
114     }
115     public String thrown() {
116         System.out.print("I");
117         return "普通だな";
118     }
119 }

131 public class FastRun implements U_RunWay {
132     public FastRun() {
133         System.out.print("P");
134     }
135     public String run() {
136         System.out.print("Q");
137         return "は・・・はやい・・・！！";
138     }
139 }

151 public class J_InsufficientRun implements U_RunWay {
152     public J_InsufficientRun() {
153         System.out.print("J");
154     }
155     public String run() {
156         return "速さが足りない";
157     }
158 }

171 public class O_NormalRun implements U_RunWay {
172     public O_NormalRun() {
173         System.out.print("R");
174     }
175     public String run() {
176         System.out.print("S");
177         return "普通だな";
178     }
179 }

191 public class Pikmin {
192     private ThrowWay thrown;
193     private AttackWay attack;
194     private U_RunWay run;
195
196     public Pikmin() { // is of Purple.
197         System.out.print("A");
198         thrown = new HeavyThrow();
199         attack = new NormalAttack();
200         run = new J_InsufficientRun();
201     }
202     public Pikmin(ThrowWay thrown, AttackWay attack, U_RunWay run) {
203         System.out.print("B");
204         this.thrown = thrown;
205         this.attack = attack;
206         this.run = run;
207     }

```



```

208
209     public String thrown() {
210         System.out.print("C");
211         return thrown.thrown();
212     }
213     public String attack() {
214         System.out.print("D");
215         return attack.attack();
216     }
217     public String run() {
218         System.out.print("E");
219         return run.run();
220     }
221 }

231 public class Red extends Pikmin{
232     public Red(){
233         super(new Q_NormalThrow(), new StrongAttack(), new O_NormalRun());
234         System.out.print("T");
235     }
236 }

241 public class White extends Pikmin {
242     public White(){
243         super(new Q_NormalThrow(), new InsufficientAttack(), new FastRun());
244         System.out.print("U");
245     }
246 }

251 public class PikminTestDrive {
252     public static void main(String args[]){
253         System.out.print("私はオリマー");
254         Pikmin pikmin = new White();
255         Pikmin pikmin2 = pikmin;
256         System.out.print("まずはこいつを投げよう , ");
257         System.out.println(pikmin2.thrown());
258
259         System.out.print("攻撃はこいつに任せよう , ");
260         pikmin = new Red();
261         System.out.println(pikmin.attack());
262
263         System.out.print("よしこつちについてこい , ");
264         pikmin = new Pikmin();
265         System.out.println(pikmin.run());
266
267         System.out.print("こいつを投げておこう , ");
268         System.out.println(pikmin2.thrown());
269     }
270 }

```

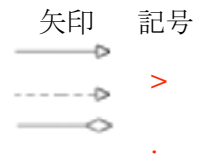
【練習 74】 出力の一行目を答えなさい

【練習 75】 出力の二行目を答えなさい

【練習 76】 出力の三行目を答えなさい

【練習 77】 出力の四行目を答えなさい

- 【練習 78】 AttackWay のクラス図の一行目を答えなさい
- 【練習 79】 AttackWay のクラス図の二行目を答えなさい
- 【練習 80】 AttackWay のクラス図の三行目を答えなさい
- 【練習 81】 Pikmin のクラス図の一行目を答えなさい
- 【練習 82】 Pikmin のクラス図の二行目を答えなさい
- 【練習 83】 Pikmin のクラス図の三行目を答えなさい
- 【練習 84】 クラス間の関係を表す矢印を答えなさい



以下について【練習 85】 から【練習 105】 を解答しなさい

全ての乗り物[Vehicle]は、運転の仕方の属性[drive Drive 型]と走る能力[run void 型]を持つ。
(メソッド run() は引数を持たない。)

走るときは、処理を運転の仕方に委譲する。運転の仕方は生成時に与えるものとする。

乗り物には、バス[Bus]と電車[Train]の 2 種類のサブクラスが存在する。

バスは「通常運転」を行う。電車は「高速運転」を行う。これらはスーパークラスのコンストラクタを介して指定する。

バスや電車には、その他に属性や機能はない。

運転の仕方[Drive]には走るメソッド[driving void 型]がある。

(ただし、Drive はインタフェースを用いる。このメソッドは引数を持たない。)

具体的な運転の仕方の種類(実装クラス)は高速運転[ExpressDrive]と通常運転[NormalDrive]の 2 つがある。

- ・それぞれのメソッド driving() では以下のように表示する。

ビューーン

ブーン

TestDrive を以下のように作成する

01. バスを生成する。名前は、vehicle とする。
02. バスを走らす。
03. 電車を生成する。
04. 電車を走らす。

【練習 85】 Vehicle のクラス図の一行目を答えなさい

【練習 86】 Vehicle のクラス図の二行目を答えなさい

【練習 87】 Vehicle のクラス図の三行目を答えなさい

【練習 88】 Drive のクラス図の一行目を答えなさい

【練習 89】 Drive のクラス図の二行目を答えなさい

【練習 90】 Drive のクラス図の三行目を答えなさい

【練習 91】 Train のクラス図の一行目を答えなさい

【練習 92】 Train のクラス図の二行目を答えなさい

【練習 93】 クラス間の関係を表す矢印を答えなさい

【練習 94】 Vehicle のソースコードの一行目を答えなさい

【練習 95】 Vehicle のソースコードの二行目を答えなさい

【練習 96】 Vehicle のソースコードの三行目を答えなさい

【練習 97】 Drive のソースコードの一行目を答えなさい

【練習 98】 Drive のソースコードの二行目を答えなさい

【練習 99】 Bus のソースコードの一行目を答えなさい

【練習 100】 Bus のソースコードの二行目を答えなさい

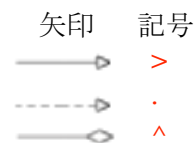
【練習 101】 Bus のソースコードの三行目を答えなさい

【練習 102】 TestDrive のメインメソッドの一行目を答えなさい

【練習 103】 TestDrive のメインメソッドの二行目を答えなさい

【練習 104】 TestDrive のメインメソッドの三行目を答えなさい

【練習 105】 TestDrive のメインメソッドの四行目を答えなさい



以下について【練習 106】 から【練習 137】 を解答しなさい

全ての動物[Animal]は体温維持の方法[regulateTemp RegulateTemp 型]、繁殖方法[breeding Breeding 型]、名前[name String 型]という属性を持つ。

動物は new するときに体温維持の方法、繁殖方法、名前を決める。

(Animal のコンストラクタの引数は regulateTemp、breeding、name の 3 つとする)

動物は自己紹介する機能を有している[explain 引数なし void 型]がある。

自己紹介では、例えば、「ライオンは恒温動物で、子を産む生き物です」のように、名前、体温維持の方法、繁殖方法を出力する。

体温維持の方法[RegulateTemp]はインタフェースを用いる。

体温維持の方法[RegulateTemp]は体温維持の方法を返す機能[get 引数なし String 型]がある。

体温維持の方法には、具体的に、恒温[Homoiotherm]と変温[Poikilotherm]の 2 種類がある。

恒温の場合はメソッド get で「恒温動物」と、変温の場合はメソッド get で「変温動物」を返す。

繁殖方法[Breeding]はインタフェースを用いる。

繁殖方法[Breeding]は繁殖方法を返す機能[get 引数なし String 型]がある。

繁殖方法には、具体的に、卵生[Oviparity]と胎生[Viviparity]の 2 種類がある。

卵生の場合はメソッド get で「卵を産む生き物」と、胎生の場合はメソッド get で「子を産む生き物」を返す。

動物にはライオン[Lion]、カメ[Turtle]、カラス[Crow]がある。

名前 | 体温 | 子の産生

ライオン | 恒温 | 胎生

カメ | 変温 | 卵生

カラス | 恒温 | 卵生

TestDrive を以下のように作成する

01. カラスを生成する。名前は animal
02. カラスに自己紹介させる。
03. ライオンを生成する。
04. ライオンに自己紹介させる。
05. カメを生成する。
06. カメに自己紹介させる。

【練習 106】 Animal のクラス図の一行目を答えなさい

【練習 107】 Animal のクラス図の二行目を答えなさい

【練習 108】 Animal のクラス図の五行目を答えなさい

【練習 109】 Animal のクラス図の六行目を答えなさい

【練習 110】 RegulateTemp のクラス図の一行目を答えなさい

【練習 111】 RegulateTemp のクラス図の二行目を答えなさい

【練習 112】 RegulateTemp のクラス図の三行目を答えなさい

【練習 113】 Lion のクラス図の一行目を答えなさい

【練習 114】 Lion のクラス図の二行目を答えなさい

【練習 115】 クラス間の関係を表す矢印を答えなさい

【練習 116】 Animal のソースコードの一行目を答えなさい

【練習 117】 Animal のソースコードの二行目を答えなさい

【練習 118】 Animal のソースコードの三行目を答えなさい

【練習 119】 Animal のソースコードの四行目を答えなさい

【練習 120】 Animal のソースコードの五行目を答えなさい

【練習 121】 Animal のソースコードの六行目を答えなさい

【練習 122】 Animal のソースコードの七行目を答えなさい

【練習 123】 Animal のソースコードの八行目を答えなさい

【練習 124】 Animal のソースコードの九行目を答えなさい

【練習 125】 Animal のソースコードの 10 行目を答えなさい

【練習 126】 Animal のソースコードの 11 行目を答えなさい

【練習 127】 Breeding のソースコードの一行目を答えなさい

【練習 128】 Breeding のソースコードの二行目を答えなさい

【練習 129】 Turtle のソースコードの一行目を答えなさい

【練習 130】 Turtle のソースコードの二行目を答えなさい

【練習 131】 Turtle のソースコードの三行目を答えなさい

【練習 132】 TestDrive のメインメソッドの一行目を答えなさい

【練習 133】 TestDrive のメインメソッドの二行目を答えなさい

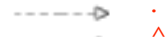
【練習 134】 TestDrive のメインメソッドの三行目を答えなさい

【練習 135】 TestDrive のメインメソッドの四行目を答えなさい

【練習 136】 TestDrive のメインメソッドの五行目を答えなさい

【練習 137】 TestDrive のメインメソッドの六行目を答えなさい

矢印 記号



練習の解答

【練習 13】 0231A

【練習 14】 この科目は合格です

【練習 15】 31 この科目は落単です

【練習 16】 CalcScore

【練習 17】 + CalcScore()

【練習 18】 + calcScore(midTestScore: int, finalTestScore: int): int

【練習 19】 Score

【練習 20】 + Score()

【練習 21】 + judgement(midTestScore: int, finalTestScore: int): String

【練習 22】 S>C

【練習 23】 0A

【練習 24】 乗り物は走ります緊急停止します B

【練習 25】 01A

【練習 26】 バスはぶーんと走ります緊急停止します B

【練習 27】 02A

【練習 28】 電車はがたんごとんと走ります緊急停止します B

【練習 29】 Vehicle

【練習 30】 # name: String

【練習 31】 + Vehicle(name: String)

【練習 32】 Bus

【練習 33】 + Bus(name: String)

【練習 34】 + run():void

【練習 35】 B>V-T>V

【練習 36】 02A

【練習 37】 1 スタインウェイポロロン調律します B

【練習 38】 023C

【練習 39】 1 ヤマハポロロン調律は要りません D

【練習 40】 Instrument

【練習 41】 - name: String

【練習 42】 + Instrument(name: String)

【練習 43】 ElectricPiano

【練習 44】 + ElectricPiano(name: String)

【練習 45】 + tuning(): void

【練習 46】 E>P-P>I

- 【練習 47】 02A
- 【練習 48】 31 計算処理を AMD の CPU を使って控えめに実行します B
- 【練習 49】 04C
- 【練習 50】 51 ゲームを Intel の CPU を使って速く実行します D
- 【練習 51】 CPU
- 【練習 52】 - speed
- 【練習 53】 + CPU(speed: String)
- 【練習 54】 OperatingSystem
- 【練習 55】 - cpu: CPU
- 【練習 56】 - task: String
- 【練習 57】 A>C-C^O-I>C

- 【練習 58】 206A
- 【練習 59】 13AMD Zen は Simultaneous Multi-Threading で高速に実行します
- 【練習 60】 407B
- 【練習 61】 15Intel i7 は Turbo Boost Max Technology で高速に実行します
- 【練習 62】 <<interface>>
- 【練習 63】 Technology
- 【練習 64】 + getName(): void
- 【練習 65】 CPU
- 【練習 66】 - name: String
- 【練習 67】 # technology: Technology
- 【練習 68】 SMT
- 【練習 69】 + SMT()
- 【練習 70】 + getName(): String
- 【練習 71】 AMDProcessor
- 【練習 72】 +AMDProcessor()
- 【練習 73】 A>C-I>C-S.T-T^C-V.T

- 【練習 74】 私はオリマーHJPBU まずはこいつを投げよう, CI 普通だな
- 【練習 75】 攻撃はこいつに任せよう, HNRBTDO 強い、圧倒的に強い
- 【練習 76】 よしこっちについてこい, AFLJE 速さが足りない

【練習 77】 こいつを投げておこう， CI 普通だな

【練習 78】 <<interface>>

【練習 79】 AttackWay

【練習 80】 + attack(): String

【練習 81】 Pikmin

【練習 82】 - thrown: ThrowWay

【練習 83】 - attack: AttackWay

【練習 84】 A^P-F.U-H.T-I.A-J.U-N.A-O.U-Q.T-R>P-S.A-T^P-U^P-W>P

【練習 85】 Vehicle

【練習 86】 - drive: Drive

【練習 87】 + Vehicle(drive: Drive)

クラス図はクラス名，フィールド，コンストラクタ，メソッドの順に解答すること．フィールド，コンストラクタ，メソッドが複数ある場合は，問題文の出現順に解答すること．

【練習 88】 <<interface>>

【練習 89】 Drive

【練習 90】 + driving(): void

【練習 91】 Train

【練習 92】 + Train()

【練習 93】 B>V-D^V-E.D-N.D-T>V

【練習 94】 public class Vehicle{

【練習 95】 private Drive drive;

【練習 96】 public Vehicle(Drive drive) {

【練習 97】 public interface Drive{

【練習 98】 public void driving();

【練習 99】 public class Bus extends Vehicle {

【練習 100】 public Bus() {

【練習 101】 super(new NormalDrive());

【練習 102】 Vehicle vehicle = new Bus();

【練習 103】 vehicle.run();

【練習 104】 vehicle = new Train();

【練習 105】 vehicle.run();

【練習106】 Animal
【練習107】 - regulateTemp: RegulateTemp
【練習108】 + Animal(regulateTemp: RegulateTemp, breeding: Breeding, name: String)
【練習109】 + explain(): void
【練習110】 <<interface>>
【練習111】 RegulateTemp
【練習112】 + get(): String
【練習113】 Lion
【練習114】 + Lion()
【練習115】 B^A-C>A-H.R-L>A-O.B-P.R-R^A-T>A-V.B
【練習116】 public class Animal {
【練習117】 private RegulateTemp regulateTemp;
【練習118】 private Breeding breeding;
【練習119】 private String name;
【練習120】 public Animal(RegulateTemp regulateTemp, Breeding breeding, String name) {
【練習121】 this.regulateTemp = regulateTemp;
【練習122】 this.breeding = breeding;
【練習123】 this.name = name;
【練習124】 }
【練習125】 public void explain() {
【練習126】 System.out.println(name + “は” + regulateTemp.get() + “で, ” + breeding.get() + “です”);
【練習127】 public interface Breeding {
【練習128】 public String get();
【練習129】 public class Turtle extends Animal {
【練習130】 public Turtle() {
【練習131】 super(new Poikilotherm(), new Oviparity(), “カメ”);
【練習132】 Animal animal = new Crow();
【練習133】 animal.explain();
【練習134】 animal = new Lion();
【練習135】 animal.explain();
【練習136】 animal = new Turtle();
【練習137】 animal.explain();