オブジェクト指向設計　予習範囲・暗記事項・予習ノート・練習問題　for 講義 No.9

・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
4 章です.

暗記事項
【暗記事項 1】から【暗記事項 20】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】共通のシグネチャとなるメソッドをもつクラスを関連付けるために
新たなクラス/インタフェースを作ることを抽象化という
【暗記事項 2】インタフェースの定義では、キーワード interface を用いる
【暗記事項 3】インタフェースの定義の 1 行目の書式は、
「アクセス修飾子　interface　インタフェース名　{ 」である
【暗記事項 4】インタフェースの定義では、メソッドに処理内容を記述しない
【暗記事項 5】インタフェース名は，通常，大文字から始める
【暗記事項 6】インタフェース名は，クラス名と識別しやすくするために，able で終わらしたり，
I(大文字のアイ)で始めることがある
【暗記事項 7】インタフェースの参照型変数を宣言できる
【暗記事項 8】インタフェースの参照型変数の宣言の書式は,「インタフェース名 変数名」である
【暗記事項 9】インタフェースは、new 演算子によりオブジェクトを生成することができない
【暗記事項 10】インタフェースを実装したクラスのことを実装クラスという
【暗記事項 11】クラスは複数のインタフェースを実装することができる
【暗記事項 12】インタフェースの実装では、キーワード implements を用いる
【暗記事項 13】インタフェースを実装したクラス(実装クラス)を定義するための書式(1 行目)は
「アクセス修飾子　class　クラス名　implements　インタフェース名　{ 」である
【暗記事項 14】実装クラスは、実装するインタフェースで定義されたメソッドを実装しないといけない
【暗記事項 15】インタフェースの参照型変数は、実装クラスのオブジェクトを参照できる
【暗記事項 16】インタフェースのクラス図では，クラス名を書く枠にインタフェース名を記述する
【暗記事項 17】インタフェースのクラス図では，インタフェース名の上にギュメ「《》」で囲んだ
ステレオタイプに interface というキーワードを入れる
【暗記事項 18】クラス図では，実装クラスからインタフェースに向けて矢印を書く
【暗記事項 19】同一の参照型変数に対して、メソッド呼出しをしても、参照しているオブジェクトに
応じて呼び出されるメソッドが変わる。これをポリモーフィズム(多態性)という
【暗記事項 20】インタフェースの参照型変数が実装クラスのオブジェクトを参照する場合，
「参照型変数の型(インタフェース名)」と「オブジェクトの型(実装クラスのクラス名)」が異なる

4.1節　抽象化

抽象化

・共通の　　となるメソッドをもつクラスを関連付けるために
　新たな　　/　　を作ること [1]
・シグネチャ：メソッド定義で，メソッド名と引数リストの引数の型のこと

抽象化の方策とメリット

・抽象的なクラスでは，（出来るだけ）変更が予想されないものを記述する
・具象クラスでは，変更が予想されるコードを記述するようにする
・以上によりコードのメンテナンスが容易になる

クラスとインタフェースの違い

・メソッドに　　を記述しない [4]
・オブジェクトを　　することができない [9]

インタフェースはロール（役割）をまとめたもの
　インタフェーズを実装しているクラスが「すべきこと／出来ること」をまとめたもの

4.2節　インタフェースの定義

インタフェースの定義

```
アクセス修飾子　　インタフェース名　{ [3]
    メソッドの宣言;
}
```

インタフェース名

・　　から始める [5]
・クラス名と　　しやすくするために，　　で終わらせたり，
　I（　文字の　　）で始めることがある [6]

4.3節　インタフェースの参照型変数とオブジェクト

インタフェースを定義することにより，　　型変数を　　できる [7]

インタフェースの参照型変数の宣言

```
　　変数名; [8]
```

インタフェースで定義したメソッドは処理内容が記述されていないので，オブジェクトを生成できたしても動作させることができない．
インタフェースは，　　演算子によりオブジェクトを生成できない [9]

4.4節　インタフェースの実装

実装クラス

インタフェースを　　したクラスのこと [10]
　　のインタフェースを実装することができる [11]

インタフェースを実装したクラスの定義

```
アクセス修飾子　class　クラス名　　インタフェース名　{ [13]
    クラスの中身
}
```

実装クラスでは，　　するインタフェースで定義されたメソッドを実装しないといけない[14]

実装クラスのオブジェクトの参照

インタフェースは新たな型を作るので，インタフェースの参照型変数を宣言することは出来る．
しかし，インタフェース自身は new 演算子によりオブジェクトを生成することはできない．
インタフェースの　　型変数は　　クラスのオブジェクトを参照することができる[15]

4.5 節　インタフェースのクラス図

インタフェースのクラス図

・　　をクラス名を書く枠に記述する[17]
・インタフェース名の上にギュメ「≪ ≫」で囲んだ
ステレオタイプに　　というキーワードを入れる[17]
・メソッドの書式は通常のクラス図と同じ

インタフェースとその実装クラスの関係を表す矢印

インタフェースの実装を表す矢印

・　　から　　に向けて矢印を書く[18]
- - - -▷

4.6 節　インタフェースとポリモーフィズム

ポリモーフィズム（多態性）

同一の参照型変数に対して、メソッド呼出しをしても、参照している　　に応じて
呼び出されるメソッドが変わること[19]

インタフェースにおけるポリモーフィズムの留意点

・インタフェースの参照型変数に対してメソッドを呼び出した場合，
参照しているオブジェクトの型（クラス）で定義されたメソッドが呼び出される
・インタフェースの参照型変数に対して呼び出せるメソッドは
インタフェースに定義されたメソッドだけである
（キャストすれば実装クラスで定義されたメソッドも呼び出せる）

参照型変数の型

参照型変数を宣言するときに用いられた型

オブジェクトの型

new 演算子を用いてオブジェクトを生成した時に指定した型

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、左詰めの 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 14】を解答しなさい

```
001   public interface IConfirmTest {
002         public void run() ;
003   }

011   public class AClass implements IConfirmTest {
012         public AClass() {
013               System.out.print("0");
014         }
015         public void run() {
016               System.out.print("1");
017         }
018   }

021   public class BClass implements IConfirmTest {
022         public BClass() {
023               System.out.print("2");
024         }
025         public void run() {
026               System.out.print("3");
027         }
028   }

031   public class CTTestDrive {
032         public static void main(String[] args) {
033               System.out.print("A");
034               IConfirmTest ict;
035               System.out.println("B");
036
037               ict = new BClass();
038               System.out.print("C");
039               ict.run();
040               System.out.println("D");
041
042               IConfirmTest ict2 = new AClass();
043               System.out.print("E");
044               ict2.run();
045               System.out.println("F");
046
047               ict = ict2;
048               System.out.print("G");
```

```
049                 ict.run();
050                 System.out.print("H");
051                 ict2.run();
052                 System.out.println("J");
053         }
054   }
```

【練習 01】出力の一行目を答えなさい　　AB
【練習 02】出力の二行目を答えなさい　　2C3D
【練習 03】出力の三行目を答えなさい　　0E1F
【練習 04】出力の四行目を答えなさい　　G1H1J
【練習 05】出力の五行目を答えなさい　　-1

フィールド・コントラクタ・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 06】IConfirmTest のクラス図の一行目を答えなさい　　<<interface>>
【練習 07】IConfirmTest のクラス図の二行目を答えなさい　　IConfirmTest
【練習 08】IConfirmTest のクラス図の三行目を答えなさい　　+ run(): void
【練習 09】IConfirmTest のクラス図の四行目を答えなさい　　-1
【練習 10】AClass のクラス図の一行目を答えなさい　　AClass
【練習 11】AClass のクラス図の二行目を答えなさい　　+ AClass()
【練習 12】AClass のクラス図の三行目を答えなさい　　+ run(): void
【練習 13】AClass のクラス図の四行目を答えなさい　　-1

矢印	記号
	>
	.
	^

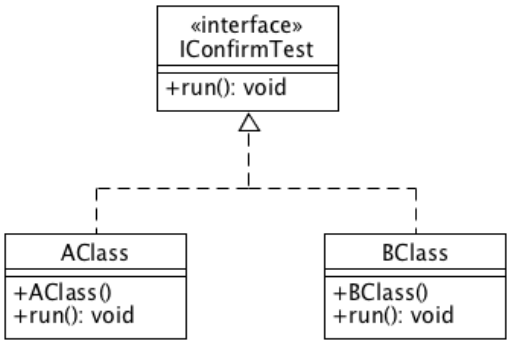
クラスの先頭の文字で，上の矢印を表す記号のどれかを囲みます.
矢印の先が後ろに来るように答えること.

【練習 14】クラス間の関係を表す矢印を答えなさい　　A.I-B.I
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
A.I と B.I だったら辞書順では A.I が先なので【練習 14】の解答例のように解答してください.

本講義の後半は，「参照型変数の型とオブジェクトの型が異なる」ことに注意しましょう！

・上のソースコードで言えば，34 行目にあるように，
　参照型変数 ict はインタフェース IConfirmTest の参照型変数として宣言されています.
・この参照型変数 ict に対して，実装クラス BClass のオブジェクトを生成しています（37 行目）.

参照型変数の型とオブジェクトの型を変えるのは，ポリモーフィズムを用いるためです.
39 行目と 49 行目は，同じ参照型変数に，同じメソッドを呼び出していますが，その時に
参照しているオブジェクトの型（クラス）が異なるので呼び出されるメソッドが変わります！



練習問題の解答のクラス図（ただし，BClass 中の記述については出題されていない）
次ページに続く

以下のソースコードについて【練習 15】から【練習 23】を解答しなさい

```
001   public interface Driver {
002         public void drive();
003   }

011   public class Car implements Driver{
012         public Car() {
013               System.out.print("0");
014         }
015         public void drive(){
016               System.out.print("車を運転します");
017         }
018   }

021   public class Train implements Driver {
022         public Train() {
023               System.out.print("1");
024         }
025         public void drive(){
026               System.out.print("電車を運転します");
027         }
028   }

031   public class DriverTestDrive {
032         public static void main(String[] args) {
033               Driver driver = new Car();
034               System.out.print("A");
035               driver.drive();
036               System.out.println("B");
037
038               driver = new Train();
039               System.out.print("C");
040               driver.drive();
041               System.out.println("D");
042
043         }
044   }
```

【練習 15】出力の一行目を答えなさい
【練習 16】出力の二行目を答えなさい
【練習 17】Driver のクラス図の一行目を答えなさい
【練習 18】Driver のクラス図の二行目を答えなさい
【練習 19】Driver のクラス図の三行目を答えなさい
【練習 20】Car のクラス図の一行目を答えなさい
【練習 21】Car のクラス図の二行目を答えなさい
【練習 22】Car のクラス図の三行目を答えなさい

矢印	記号
	>
	.
	^

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 23】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 24】から【練習 33】を解答しなさい

```
001   public interface Instrument {
002         public void sound();
003   }

011   public class Guitar implements Instrument {
012         public Guitar(){
013               System.out.print("a");
014         }
015         public void sound() {
016               System.out.print("ジャカジャーン");
017         }
018   }

021   public class Piano implements Instrument{
022         public Piano(){
023               System.out.print("b");
024         }
025         public void sound() {
026               System.out.print("ポロンポロン");
027         }
028   }

031   public class InstrumentTestDrive {
032         public static void main(String[] args) {
033               Instrument ins1 = new Guitar();
034               System.out.print("z");
035               ins1.sound();
036               System.out.println("y");
037
038               Instrument ins2 = new Piano();
039               ins2.sound();
040               System.out.println("x");
041
042               ins1=ins2;
043               ins1.sound();
044               System.out.println("w");
045         }
046   }
```

【練習 24】出力の一行目を答えなさい
【練習 25】出力の二行目を答えなさい
【練習 26】出力の三行目を答えなさい
【練習 27】Instrument のクラス図の一行目を答えなさい
【練習 28】Instrument のクラス図の二行目を答えなさい
【練習 29】Instrument のクラス図の三行目を答えなさい
【練習 30】Piano のクラス図の一行目を答えなさい
【練習 31】Piano のクラス図の二行目を答えなさい
【練習 32】Piano のクラス図の三行目を答えなさい

矢印	記号
	>
	.
	^

【練習 33】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 34】から【練習 41】を解答しなさい

```
001  public interface OperatingSystem {
002      public void run();
003  }

011  public class Linux implements OperatingSystem{
012      public Linux() {
013          System.out.print("0");
014      }
015      public void run() {
016          System.out.print("コマンドラインのプログラムも多いです");
017      }
018  }

021  public class MacOS implements
                              OperatingSystem {
022      public MacOS(){
023          System.out.print("2");
024      }
025      public void run() {
026          System.out.print("操作が直感的です");
027      }
028  }

031  public class Windows implements
                                  OperatingSystem{
032      public Windows() {
033          System.out.print("1");
034      }
035      public void run() {
036          System.out.print("GUI アプリが多いです");
037      }
038  }

041  public class OperatingSystemTestDrive {
042      public static void main(String[] args) {
043          System.out.print("A");
044          OperatingSystem os;
045          System.out.println("B");
046
047          os = new Windows();
048          os.run();
049          System.out.println("C");
050
051          os = new Linux();
052          os.run();
053          System.out.println("D");
054
055          os = new MacOS();
056          os.run();
057          System.out.println("E");
058      }
059  }
```

【練習 34】出力の一行目を答えなさい
【練習 35】出力の二行目を答えなさい
【練習 36】出力の三行目を答えなさい
【練習 37】出力の四行目を答えなさい
【練習 38】OperatingSystem のクラス図の一行目を答えなさい
【練習 39】OperatingSystem のクラス図の二行目を答えなさい
【練習 40】OperatingSystem のクラス図の三行目を答えなさい

矢印	記号
	>
	.
	^

【練習 41】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 42】から【練習 52】を解答しなさい

```
001  public interface Pikmin {
002      public String thrown();
003      public String attack();
004      public String run();
005  }

011  public class APurple implements Pikmin{
012      public APurple(){
013          System.out.print("0");
014      }
015      public String thrown() {
016          System.out.print("1");
017          return "おっも・・・";
018      }
019      public String attack() {
020          System.out.print("2");
021          return "普通だな";
022      }
023      public String run() {
024          System.out.print("3");
025          return "速さが足りない";
026      }
027  }

031  public class Red implements Pikmin{
032      public Red(){
033          System.out.print("4");
034      }
035      public String thrown() {
036          System.out.print("5");
037          return "普通だな";
038      }
039      public String attack() {
040          System.out.print("6");
041          return “強い，圧倒的に強い";
042      }
043      public String run() {
044          System.out.print("7");
045          return "普通だな";
046      }
047  }
```

```
051  public class White implements Pikmin{
052      public White(){
053          System.out.print("8");
054      }
055      public String thrown() {
056          System.out.print("9");
057          return "普通だな";
058      }
059      public String attack() {
060          System.out.print("a");
061          return "蚊に刺されたのかと";
062      }
063      public String run() {
064          System.out.print("b");
065          return "は・・・はやい・・・！！";
066      }
067  }

071  public class PikminTestDrive {
072      public static void main(String args[]){
073          System.out.print("私はオリマー");
074          Pikmin pikmin, pikmin2;
075          System.out.println("z");
076
077      System.out.print("まずはこいつを投げよう，");
078          pikmin = pikmin2 = new White();
079          System.out.println(pikmin.thrown());
080
081      System.out.print("攻撃はこいつに任せよう，");
082          pikmin = new Red();
083          System.out.println(pikmin.attack());
084
085      System.out.print("よしこっちについてこい，");
086          pikmin = new APurple();
087          System.out.println(pikmin.run());
088
089       System.out.print("こいつを投げておこう，");
090          pikmin = pikmin2;
091          System.out.println(pikmin.thrown());
092      }
093  }
```

【練習 42】出力の一行目を答えなさい
【練習 43】出力の二行目を答えなさい
【練習 44】出力の三行目を答えなさい
【練習 45】出力の四行目を答えなさい
【練習 46】出力の五行目を答えなさい
【練習 47】Pikmin のクラス図の一行目を答えなさい
【練習 48】Pikmin のクラス図の二行目を答えなさい
【練習 49】Pikmin のクラス図の三行目を答えなさい
【練習 50】Pikmin のクラス図の四行目を答えなさい
【練習 51】Pikmin のクラス図の五行目を答えなさい
【練習 52】クラス間の関係を表す矢印を答えなさい

本練習問題（後半）の答え方

以下について【練習 53】から【練習 68】を解答しなさい

全ての動物[Animal]は，飛ぶ能力[fly void 型],と掘る能力[dig void 型]を持つ.
(ただし，Animal はインタフェースを用いる．また,これらのメソッドは引数を持たない．)

動物の種類(実装クラス)はオオカミ[Wolf],ツバメ[Swallow],モグラ[Mole]の 3 つがある.

・それぞれのメソッド fly()では以下のように表示する．　メソッドの中身です．println()を使ってください.
　オオカミは飛べません.
　ツバメは飛べます.
　モグラは飛べません.

・一方，メソッド dig()では以下のように表示する.
　オオカミは少しなら掘れます.
　ツバメは掘れません.
　モグラはめちゃくちゃ掘れます.

TestDrive を以下のように作成する　　プロ実 1 の UML 作成テストでは，強く要請されていないと
01. オオカミを生成する．名前は，animal とする．　思いますが，本講義では，インタフェース・親クラスの
02. オオカミの飛ぶ能力を表示させる.　　参照型変数を用いるようにしてください
03. ツバメを生成する.　←　可能な限り参照型変数を使いまわしてください.
04. ツバメの飛ぶ能力を表示させる.
05. ツバメの掘る能力を表示させる.
06. モグラを生成する.
07. モグラの掘る能力を表示させる

【練習 53】Animal のクラス図の一行目を答えなさい　　<<interface>>
【練習 54】Animal のクラス図の二行目を答えなさい　　Animal
【練習 55】Animal のクラス図の三行目を答えなさい　　+ fly(): void　クラス図のメソッドは問題文で
【練習 56】Wolf のクラス図の一行目を答えなさい　　Wolf　　出てきた順番にしてください.
【練習 57】Wolf のクラス図の二行目を答えなさい　　+ fly(): void　動物インタフェースの dig()が四行
【練習 58】Wolf のクラス図の三行目を答えなさい　　+ dig(): void　目にくるはずです.
【練習 59】クラス間の関係を表す矢印を答えなさい　　M.A-S.A-W.A
【練習 60】Animal のソースコードの一行目を答えなさい　　public interface Animal {
【練習 61】Animal のソースコードの二行目を答えなさい　　public void fly();
【練習 62】Swallow のソースコードの一行目を答えなさい　　public class Swallow implements Animal {
【練習 63】Swallow のソースコードの二行目を答えなさい　　public void fly() {
【練習 64】Swallow のソースコードの三行目を答えなさい　　System.out.println(“ツバメは飛べます．”);
【練習 65】TestDrive のメインメソッドの一行目を答えなさい　　Animal animal = new Wolf();
【練習 66】TestDrive のメインメソッドの二行目を答えなさい　　animal.fly();

【練習 67】TestDrive のメインメソッドの三行目を答えなさい　　animal = new Swallow();

【練習 68】TestDrive のメインメソッドの四行目を答えなさい　　animal.fly();

参考（前ページの問題で本来作成すべきソースコードとクラス図）

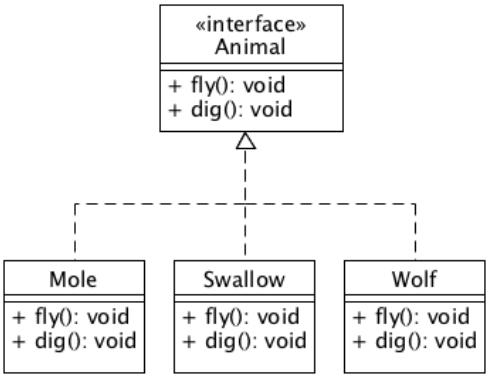
```
001  public interface Animal {
002      public void fly();
003      public void dig();
004  }

011  public class Mole implements Animal {
012      public void fly() {
013          System.out.println("モグラは飛べません．");
014      }
015      public void dig() {
016          System.out.println("モグラはめちゃくちゃ掘れます．");
017      }
018  }

021  public class Swallow implements Animal {
022      public void fly() {
023          System.out.println("ツバメは飛べます．");
024      }
025      public void dig() {
026          System.out.println("ツバメは掘れません．");
027      }
028  }

031  public class Wolf implements Animal {
032      public void fly() {
033          System.out.println("オオカミは飛べません．");
034      }
035      public void dig() {
036          System.out.println("オオカミは少しなら掘れます．");
037      }
038  }

041  public class AnimalTestDrive {
042      public static void main(String[] args) {
043          Animal animal = new Wolf();
044          animal.fly();
045          animal = new Swallow();
046          animal.fly();
047          animal.dig();
048          animal = new Mole();
049          animal.dig();
050      }
051  }
```



実行例

```
オオカミは飛べません．
ツバメは飛べます．
ツバメは掘れません．
モグラはめちゃくちゃ掘れます．
```

以下について【練習 69】から【練習 84】を解答しなさい

全ての乗り物[Vehicle]は，走る能力[drive void 型]を持つ.
(ただし，Vehicle はインタフェースを用いる．また,これらのメソッドは引数を持たない．)

乗り物の種類(実装クラス)は電車[Train]とバス[Bus]の 2 つがある.
・それぞれのメソッド drive()では以下のように表示する.
　ガタンゴトン
　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 69】Vehicle のクラス図の一行目を答えなさい
【練習 70】Vehicle のクラス図の二行目を答えなさい
【練習 71】Vehicle のクラス図の三行目を答えなさい
【練習 72】Train のクラス図の一行目を答えなさい
【練習 73】Train のクラス図の二行目を答えなさい
【練習 74】Train のクラス図の三行目を答えなさい
【練習 75】クラス間の関係を表す矢印を答えなさい
【練習 76】Vehicle のソースコードの一行目を答えなさい
【練習 77】Vehicle のソースコードの二行目を答えなさい
【練習 78】Bus のソースコードの一行目を答えなさい
【練習 79】Bus のソースコードの二行目を答えなさい
【練習 80】Bus のソースコードの三行目を答えなさい
【練習 81】TestDrive のメインメソッドの一行目を答えなさい
【練習 82】TestDrive のメインメソッドの二行目を答えなさい
【練習 83】TestDrive のメインメソッドの三行目を答えなさい
【練習 84】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 85】から【練習 100】を解答しなさい

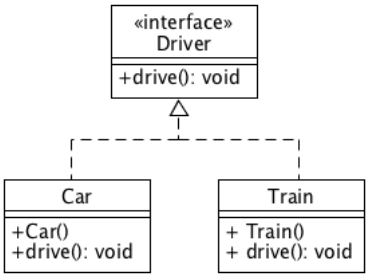
全ての OS[OperatingSystem]は，実行する能力[run void 型]を持つ．
(ただし，OperatingSystem はインタフェースを用いる．メソッドは引数を持たない．)

OS の種類(実装クラス)はリナックス[Linux]，マック[MacOS]，ウインドウズ[Windows]の 3 つがある．
・それぞれのメソッド run()では以下のように表示する．
　コマンドラインのプログラムも多いです
　操作が直感的です
　GUI アプリが多いです

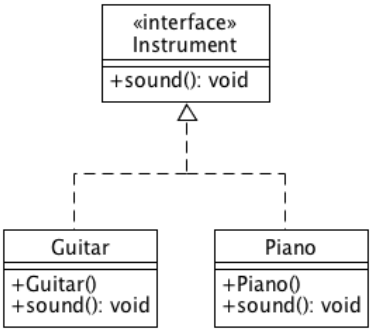
TestDrive を以下のように作成する
01. ウインドウズを生成する．名前は，os とする．
02. ウインドウズの実行する能力を表示させる．
03. マックを生成する．
04. マックの実行する能力を表示させる．
05. リナックスを生成する．
06. リナックスの実行する能力を表示させる．

【練習 85】OperatingSystem のクラス図の一行目を答えなさい
【練習 86】OperatingSystem のクラス図の二行目を答えなさい
【練習 87】OperatingSystem のクラス図の三行目を答えなさい
【練習 88】Linux のクラス図の一行目を答えなさい
【練習 89】Linux のクラス図の二行目を答えなさい
【練習 90】Linux のクラス図の三行目を答えなさい
【練習 91】クラス間の関係を表す矢印を答えなさい
【練習 92】OperatingSystem のソースコードの一行目を答えなさい
【練習 93】OperatingSystem のソースコードの二行目を答えなさい
【練習 94】MacOS のソースコードの一行目を答えなさい
【練習 95】MacOS のソースコードの二行目を答えなさい
【練習 96】MacOS のソースコードの三行目を答えなさい
【練習 97】TestDrive のメインメソッドの一行目を答えなさい
【練習 98】TestDrive のメインメソッドの二行目を答えなさい
【練習 99】TestDrive のメインメソッドの三行目を答えなさい
【練習 100】TestDrive のメインメソッドの四行目を答えなさい

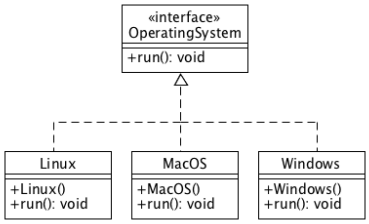
【練習 15】0A 車を運転します B
【練習 16】1C 電車を運転します D
【練習 17】<<interface>>
【練習 18】Driver
【練習 19】+ drive(): void
【練習 20】Car
【練習 21】+ Car()
【練習 22】+ drive(): void
【練習 23】C.D-T.D



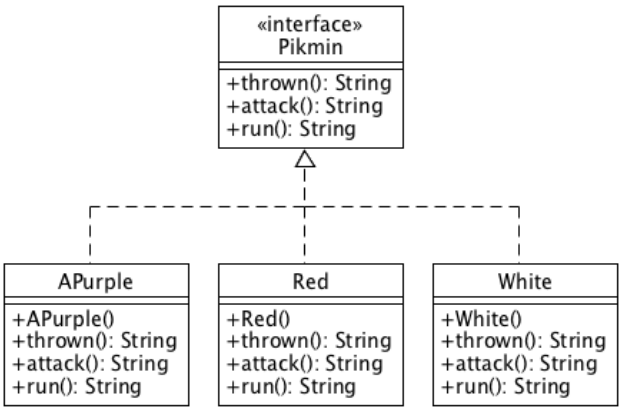
【練習 24】az ジャカジャーン y
【練習 25】b ポロンポロン x
【練習 26】ポロンポロン w
【練習 27】<<interface>>
【練習 28】Instrument
【練習 29】+ sound(): void
【練習 30】Piano
【練習 31】+ Piano()
【練習 32】+ sound(): void
【練習 33】G.I-P.I



【練習 34】AB
【練習 35】1GUI アプリが多いです C
【練習 36】0 コマンドラインのプログラムも多いです D
【練習 37】2 操作が直感的です E
【練習 38】<<interface>>
【練習 39】OperatingSystem
【練習 40】+ run(): void
【練習 41】L.O-M.O-W.O



【練習 42】私はオリマーz
【練習 43】まずはこいつを投げよう，89 普通だな
【練習 44】攻撃はこいつに任せよう，46 強い，圧倒的に強い
【練習 45】よしこっちについてこい，03 速さが足りない
【練習 46】こいつを投げておこう，9 普通だな
【練習 47】<<interface>>
【練習 48】Pikmin
【練習 49】+ thrown(): String
【練習 50】+ attack(): String
【練習 51】+ run(): String
【練習 52】A.P-R.P-W.P



【練習 69】<<interface>>
【練習 70】Vehicle
【練習 71】+ drive(): void
【練習 72】Train
【練習 73】+ drvie(): void
【練習 74】-1
【練習 75】B.V-T.V
【練習 76】public interface Vehicle {
【練習 77】public void drive();
【練習 78】public class Bus implements Vehicle {
【練習 79】public void drive() {
【練習 80】System.out.println(“ブーン”);
【練習 81】Vehicle vehicle = new Bus();
【練習 82】vehicle.drive();
【練習 83】vehicle = new Train();
【練習 84】vehicle.drive();

【練習 85】<<interface>>
【練習 86】OperatingSystem
【練習 87】+ run(): void
【練習 88】Linux
【練習 89】+ run(): void
【練習 90】-1
【練習 91】L.O-M.O-W.O
【練習 92】public interface OperatingSystem {
【練習 93】public void run();
【練習 94】public class MacOS implements OperatingSystem {
【練習 95】public void run() {
【練習 96】System.out.println(“操作が直感的です”);
【練習 97】OperatingSystem os = new Windows();
【練習 98】os.run();
【練習 99】os = new MacOS();
【練習 100】os.run();