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.10

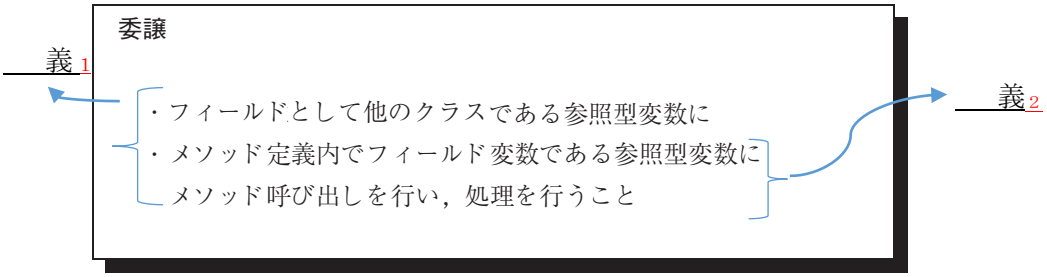
・暗記事項を参考にして予習ノートの作成
・予習範囲の熟読
・暗記事項の暗記
・教科書の演習問題を理解
上記を済ませてから練習問題に取り組んでください。
最終ページに解答を載せています。

予習範囲
5 章です.

暗記事項
【暗記事項 1】から【暗記事項 12】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】広義の委譲では、「フィールドとして他のクラスの参照型変数を持ち、メソッド定義内でフィールド の参照型変数にメソッド呼び出しを行い、処理を行うこと」を意味している.
【暗記事項2】狭義の委譲は、「メソッド定義内でフィールドの参照型変数にメソッド呼び出しを行い、処理を行うこと」を指し、処理を委譲することを表している.
【暗記事項3】「フィールドとして他のクラスの参照型変数を持つこと」をコンポジションという.
【暗記事項4】UML では広義の委譲を、コンポジション(問題3と同じ意味ではない)と集約に分けている.
【暗記事項5】UML でのコンポジションでは、フィールドの参照型変数が参照しているオブジェクトと、フィールド を持っているオブジェクトのライフタイム(オブジェクトの生成・消滅の時期)がほぼ同じであることを意味している.
【暗記事項6】集約では、フィールドの参照型変数が参照しているオブジェクトと、フィールドを持っているオブジェクトのライフタイムに関連がない.
【暗記事項7】委譲のことを HAS-A 関係という
【暗記事項9】処理を委譲するメソッドのメソッド名は、委譲されるメソッドのメソッド名と同じでなくてもよい
【暗記事項10】UMLの集約を表すクラス図では矢印◇── を用いる.
【暗記事項11】UMLのコンポジションを表すクラス図では矢印 ◆── を用いる.
【暗記事項12】クラス図において，集約とコンポジションでは，委譲している他クラスの参照型変数をフィールドに持っているクラスにひし形を描く.
クラス図で集約とコンポジションの違いを問うような問題はだしません．つまり，委譲の矢印は◇──か ◆── のどちらかを使ってください.

5.1節　委譲



（注）処理を委譲するメソッドは，委譲されるメソッドと　　である必要はない9

委譲とコンポジション

・コンポジション：フィールドとして　　クラスの参照型変数を持つこと9
↑UMLでのコンポジションではなく，一般用語としてのコンポジションの定義です
・委譲　　：メソッド定義において，フィールドに対してメソッド呼出しを行うことで処理を実現すること

HAS-A関係

・　　の関係にあるクラスのこと7

5.2節　集約とコンポジション

　　では広義の委譲を　　と　　に分けている4

(UMLでの)コンポジション

フィールドの参照型変数が　　いるオブジェクトと
フィールドを　　いるオブジェクトのライフタイムが
　　であることを意味している5

ライフタイム：オブジェクトが生成してから消滅するまでの期間

集約

フィールドの参照型変数が　　いるオブジェクトと
フィールドを　　いるオブジェクトのライフタイムに
　がないことを意味している6

5.3節　委譲のクラス図

UMLの　　を表すクラス図では矢印◇――を用いる10
UMLの　　を表すクラス図では矢印◆――を用いる11

クラス図において，集約とコンポジションでは，
委譲している他クラスの参照型変数を　　に持っている　　にひし形を描く12

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 17】を解答しなさい

```
001  public interface IVehicle {
002    public void move();
003  }

011  public class Car implements IVehicle {
012    public Car() {
013      System.out.print(“0");
014    }
015    public void move() {
016      System.out.print("1");
017    }
018  }

021  public class Bus implements IVehicle {
022    public Bus() {
023      System.out.print("2");
024    }
022    public void move() {
023      System.out.print("3");
024    }
025  }

031  public class TransportSystem {
032    private IVehicle vehicle;
033    public TransportSystem() {
034      vehicle = new Car();
035      System.out.print("4");
036    }
037    public TransportSystem(IVehicle v) {
038      vehicle = v;
039      System.out.print("5");
040    }
041
042    public void move() {
043      vehicle.move();
044      System.out.print("6");
045    }
046  }

051  public class VehicleTestdrive {
052   public static void main(String[] args) {
053     TransportSystem trans =
                          new TransportSystem();
054     System.out.println("A");
055
056     trans.move();
057     System.out.println("B");
058
059     IVehicle v = new Bus();
060     System.out.print("C");
061     trans = new TransportationSystem(v);
062     System.out.println("D");
063
064     trans.move();
065     System.out.println("E");
066   }
067  }
```

【練習 01】出力の一行目を答えなさい　　04A
【練習 02】出力の二行目を答えなさい　　16B
【練習 03】出力の三行目を答えなさい　　2C5D
【練習 04】出力の四行目を答えなさい　　36E
【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】IVehicle のクラス図の一行目を答えなさい　　<<interface>>
【練習 07】IVehicle のクラス図の二行目を答えなさい　　IVehicle
【練習 08】IVehicle のクラス図の三行目を答えなさい　　+ move(): void
【練習 09】Car のクラス図の一行目を答えなさい　　Car
【練習 10】Car のクラス図の二行目を答えなさい　　+ Car()

【練習 11】Car のクラス図の三行目を答えなさい　　+ move(): void
【練習 12】TransportSystem のクラス図の一行目を答えなさい　　TransportSystem
【練習 13】TransportSystem のクラス図の二行目を答えなさい　　- vehicle: IVehicle
【練習 14】TransportSystem のクラス図の三行目を答えなさい　　+ TransportSystem()
【練習 15】TransportSystem のクラス図の四行目を答えなさい　　+ TransportSystem(v: IVehicle)
【練習 16】TransportSystem のクラス図の五行目を答えなさい　　+ move(): void

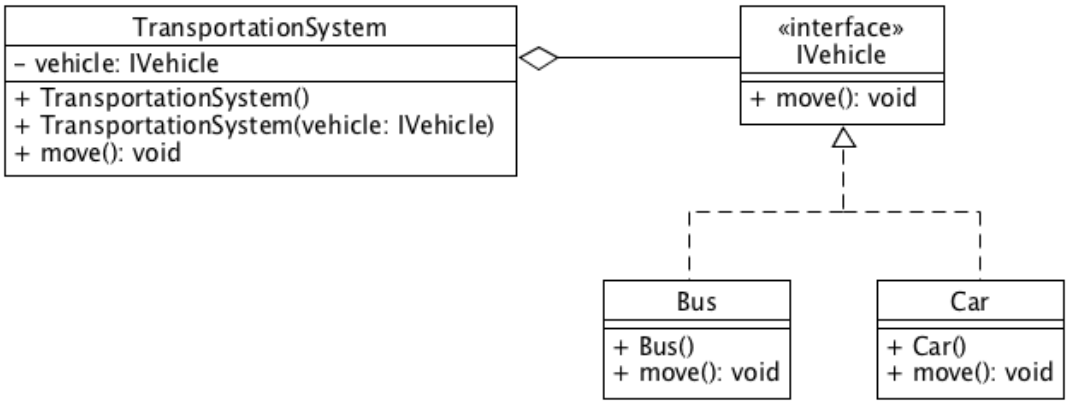
クラスの先頭の文字で，上の矢印を表す記号のどれかを囲みます．
矢印の先が後ろに来るように答えること．

【練習 17】クラス間の関係を表す矢印を答えなさい　　B.I-C.I-I^T

矢印	記号
	>
	.
	^

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.I と C.I だったら辞書順では B.I が先なので【練習 17】の解答例のように解答してください．

クラス図中の TranportationSystem は TransportSystem に読み替えてください



練習問題の解答のクラス図（ただし，クラス Bus のクラス図の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 18】から【練習 27】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 {
022      private CalcScore calcScore;
023      public Score(){
024          System.out.print("2");
025          calcScore = new CalcScore();
026      }
027      public String calcScore(int midTestScore,int finalTestScore){
028          System.out.print("3");
029          int score = calcScore.calcScore(midTestScore, finalTestScore);
030          if(score>=60){
031              return "この科目は合格です";
032          }else{
033              return "この科目は落単です";
034          }
035      }
036  }

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

【練習 18】出力の一行目を答えなさい
【練習 19】出力の二行目を答えなさい
【練習 20】出力の三行目を答えなさい
【練習 21】CalcScore のクラス図の一行目を答えなさい
【練習 22】CalcScore のクラス図の二行目を答えなさい
【練習 23】CalcScore のクラス図の三行目を答えなさい
【練習 24】Score のクラス図の一行目を答えなさい
【練習 25】Score のクラス図の二行目を答えなさい
【練習 26】Score のクラス図の三行目を答えなさい
【練習 27】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 28】から【練習 38】を解答しなさい

```
001  public interface Appliance {
002      public String behave();
003      public String getName();
004      public void     setFlag(boolean flag);
005  }

011  public class User {
012      private Appliance ap;
013      public User(Appliance ap){
014          System.out.print("0");
015          this.ap=ap;
016      }
017      public void setAp(Appliance ap){
018          System.out.print("1");
019          this.ap=ap;
020      }
021      public String use(){
022          System.out.print("2");
023          return ap.getName() + "は" + ap.behave();
024      }
025  }

031  public class Oven implements Appliance {
032      private boolean flag;
033      private String name;
034      public Oven(boolean flag){
035          System.out.print("3");
036          this.flag = flag;
037          name = "オーブン";
038      }
039      public String behave() {
040          System.out.print("4");
041          if(flag)
042              return "チーーーンと言いました";
043          return "壊れているようです";
044      }
045      public String getName(){
046          System.out.print("5");
047          return name;
048      }
049      public void setFlag(boolean flag){
050          System.out.print("6");
051          this.flag=flag;
052      }
053  }

061  public class Refrigerator implements Appliance {
062      private boolean flag = true;
063      public String behave() {
064          System.out.print("7");
065          if (flag)
066              return "キンキンに冷えている";
067          return "壊れました";
068      }
069      public String getName(){
070          System.out.print("8");
071          return "冷蔵庫";
```

```
072         }
073         public void setFlag(boolean flag){
074             System.out.print("9");
075             this.flag = flag;
076         }
077     }

081     public class ApplianceTestDrive {
082         public static void main(String[] args) {
083             Appliance appliance = new Oven(true);
084             User user = new User(appliance);
085             System.out.println("A");
086             System.out.println(user.use());
087             appliance.setFlag(false);
088             System.out.println(user.use());
089             user.setAp(new Refrigerator());
090             System.out.println(user.use());
091         }
092     }
```

【練習 28】出力の一行目を答えなさい
【練習 29】出力の二行目を答えなさい
【練習 30】出力の三行目を答えなさい
【練習 31】出力の四行目を答えなさい
【練習 32】Appliance のクラス図の一行目を答えなさい
【練習 33】Appliance のクラス図の二行目を答えなさい
【練習 34】Appliance のクラス図の三行目を答えなさい
【練習 35】User のクラス図の一行目を答えなさい
【練習 36】User のクラス図の二行目を答えなさい
【練習 37】User のクラス図の三行目を答えなさい
【練習 38】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 39】から【練習 49】を解答しなさい

```
001  public interface OperatingSystem {
002      public String usingOS();
003  }

011  public class A_OSmanagement {
012      private OperatingSystem os;
013      public A_OSmanagement() {
014          System.out.print("6");
015          this.os = new MacOS();
016      }
017      public A_OSmanagement(OperatingSystem os) {
018          System.out.print("7");
019          this.os = os;
020      }
021      public void setOS(OperatingSystem os) {
022          System.out.print("8");
023          this.os = os;
024      }
025      public String usingOS() {
026          System.out.print("9");
027          return os.usingOS();
028      }
029  }

041  public class MacOS implements OperatingSystem{
042      public MacOS(){
043          System.out.print("2");
044      }
045      public String usingOS() {
046          System.out.print("3");
047          return "現在 Mac を使用しています";
048      }
049  }

061  public class Windows implements OperatingSystem{
062      public Windows() {
063          System.out.print("4");
064      }
065      public String usingOS() {
066          System.out.print("5");
067          return "Windows を使用しています";
068      }
069  }

081  public class OperatingSystemTestDrive {
082      public static void main(String[] args) {
083          A_OSmanagement osm = new A_OSmanagement();
084          System.out.println("A");
085          String str = osm.usingOS();
086          System.out.println(str);
087          osm.setOS(new Windows());
088          str = osm.usingOS();
089          System.out.println(str);
090      }
091
092  }
```

【練習 39】出力の一行目を答えなさい
【練習 40】出力の二行目を答えなさい
【練習 41】出力の三行目を答えなさい
【練習 42】OperatingSystem のクラス図の一行目を答えなさい
【練習 43】OperatingSystem のクラス図の二行目を答えなさい

【練習 44】OperatingSystem のクラス図の三行目を答えなさい

【練習 45】A_OSmanagement のクラス図の一行目を答えなさい
【練習 46】A_OSmanagement のクラス図の二行目を答えなさい
【練習 47】A_OSmanagement のクラス図の三行目を答えなさい
【練習 48】A_OSmanagement のクラス図の四行目を答えなさい
【練習 49】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 50】から【練習 60】を解答しなさい

```
001  public interface KamenRider {
002      public void henshin();
003      public void finalMode();
004      public void deadlyBlow();
005  }

011  public class RiderMove {
012      private KamenRider rider;
013      public RiderMove(KamenRider rider){
014          this.rider = rider;
015      }
016      public void henshin(){
017          rider.henshin();
018      }
019      public void finalMode(){
020          rider.finalMode();
021      }
022      public void deadlyBlow(){
023          rider.deadlyBlow();
024      }
025  }

031  public class Drive implements KamenRider{
032      public void henshin() {
033          System.out.println("ドライブ　タイプ：スピード");
034      }
035      public void finalMode() {
036          System.out.println("ドライブ　タイプ：トライドロン");
037      }
038      public void deadlyBlow() {
039          System.out.println("フールスロットル！！");
040      }
041  }

051  public class OOO implements KamenRider{
052      public void henshin() {
053          System.out.println("タカ・トラ・バッタ　タ・ト・バ　タトバ　タ・ト・バ");
054      }
055      public void finalMode() {
056          System.out.println("タカ・クジャク・コンドル　タージャードルー");
057      }
058      public void deadlyBlow() {
059          System.out.println("スキャニングチャージ！");
060      }
061  }

071  public class A_Ryuki implements KamenRider{
072      public void henshin() {
073          System.out.println("（効果音のみ）");
074      }
075      public void finalMode(){
076          System.out.println("サバイブ");
077      }
078      public void deadlyBlow() {
079          System.out.println("ファイナル・ベント");
080      }
081  }
```

```
091  public class RiderMoveTestDrive {
092      public static void main(String args[]){
093          RiderMove rider01 = new RiderMove(new A_Ryuki());
094          System.out.println("変身！");
095          rider01.henshin();
096          rider01.deadlyBlow();
097
098          RiderMove rider02 = new RiderMove(new OOO());
099          System.out.println("変身！");
100          rider02.henshin();
101          rider02.finalMode();
102
103          RiderMove rider03 = new RiderMove(new Drive());
104          System.out.println("変身！");
105          rider03.finalMode();
106          rider03.henshin();
107      }
108  }
```

注）main メソッド以外にも println()があります！

【練習 50】出力の一行目を答えなさい
【練習 51】出力の二行目を答えなさい
【練習 52】出力の三行目を答えなさい
【練習 53】出力の四行目を答えなさい
【練習 54】出力の五行目を答えなさい
【練習 55】RiderMove のクラス図の一行目を答えなさい
【練習 56】RiderMove のクラス図の二行目を答えなさい
【練習 57】RiderMove のクラス図の三行目を答えなさい
【練習 58】RiderMove のクラス図の四行目を答えなさい
【練習 59】RiderMove のクラス図の五行目を答えなさい
【練習 60】クラス間の関係を表す矢印を答えなさい

矢印	記号
	>
	.
	^

本練習問題（後半）の答え方

以下について【練習 61】から【練習 87】を解答しなさい

四足獣[Quadruped]は，種類[type String 型]と走り方[run Runnable 型]という属性を持つ．
四足獣を生成するときに，具体的な種類と走り方を与える
　　← コンストラクタに引数が必要なこと．また，引数をフィールドに代入させることを暗示しています．
四足獣は説明するメソッド[explain void 型]を有している．
(このメソッドは引数を持たない．)
type+”は”+run.run()
メソッドの中身です．println()を使ってください

走り方[Runnable]には走るメソッド[run String 型]がある．
(ただし，Runnable 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．)

走り方の種類(実装クラス)には速く走る[FastRun],ゆっくり走る[SlowRun]の 2 つがある．
・それぞれのメソッド run()では以下の文字列を返す．.
　速く走ります
　ゆっくり走ります

TestDrive を以下のように作成する
01. 四足獣（オオカミ，速く走る）を生成する．
　　名前は，quadruped とする．
02. 四足獣（オオカミ，速く走る）を説明する．
03. 四足獣（ゾウ，ゆっくり走る）を生成する．
04. 四足獣（オオカミ，速く走る）を説明する．

プロ実 1 の UML 作成テストでは，強く要請されていないと思いますが，本講義では，インタフェース・親クラス の参照型変数を用いるようにしてください
← 可能な限り参照型変数を使いまわしてください．

クラス図はフィールド・コンストラクタ・メソッドの順で解答してください．フィールド／メソッドが複数個出てきた場合は問題文で出てきた順番に解答してください．

【練習 61】Quadruped のクラス図の一行目を答えなさい　Quadruped
【練習 62】Quadruped のクラス図の二行目を答えなさい　- type: String
【練習 63】Quadruped のクラス図の三行目を答えなさい　- run: Runnable
【練習 64】Quadruped のクラス図の四行目を答えなさい　+ Quadruped(type: String, run: Runnable)
【練習 65】Quadruped のクラス図の五行目を答えなさい　+ explain(): void
【練習 66】Runnable のクラス図の一行目を答えなさい　<<interface>>
【練習 67】Runnable のクラス図の二行目を答えなさい　Runnable
【練習 68】Runnable のクラス図の三行目を答えなさい　+ run(): String
【練習 69】FastRun のクラス図の一行目を答えなさい　FastRun
【練習 70】FastRun のクラス図の二行目を答えなさい　+ run(): String

【練習 71】クラス間の関係を表す矢印を答えなさい　　F.R-R^Q-S.R

【練習 71】Quadruped のソースコードの一行目を答えなさい public class Quadruped {
【練習 72】Quadruped のソースコードの二行目を答えなさい private String type;
【練習 73】Quadruped のソースコードの三行目を答えなさい private Runnable run;
【練習 74】Quadruped のソースコードの四行目を答えなさい public Quadruped(String type, Runnable run) {
【練習 75】Quadruped のソースコードの五行目を答えなさい this.type = type;
【練習 76】Quadruped のソースコードの六行目を答えなさい this.run = run;
【練習 77】Quadruped のソースコードの八行目を答えなさい public void explain() {
【練習 78】Quadruped のソースコードの九行目を答えなさい System.out.println(type+"は"+run.run());
【練習 79】Runnable のソースコードの一行目を答えなさい　public interface Runnable{
【練習 80】Runnable のソースコードの二行目を答えなさい　public String run();
【練習 81】SlowRun のソースコードの一行目を答えなさい　public class SlowRun implements Runnable{
【練習 82】SlowRun のソースコードの二行目を答えなさい　public String run() {
【練習 83】SlowRun のソースコードの三行目を答えなさい　return “ゆっくり走ります”;
【練習 84】TestDrive のメインメソッドの一行目を答えなさい
Quadruped quadruped = new Quadruped(“オオカミ”, new FastRun());
【練習 85】TestDrive のメインメソッドの二行目を答えなさい　　quadruped.explain();
【練習 86】TestDrive のメインメソッドの三行目を答えなさい
quadruped= new Quadruped(“ゾウ”, new SlowRun());
【練習 87】TestDrive のメインメソッドの四行目を答えなさい　　quadruped.explain();

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 Runnable {
002         public String run();
003   }

011   public class Quadruped {
012         private String type;
013         private Runnable run;
014         public Quadruped(String type, Runnable run) {
015               this.type = type;
016               this.run = run;
017         }
018         public void explain() {
019               System.out.println(type + "は" + run.run());
020         }
021   }

031   public class FastRun implements Runnable {
032         public String run() {
033               return "はやく走ります";
034         }
035   }
```

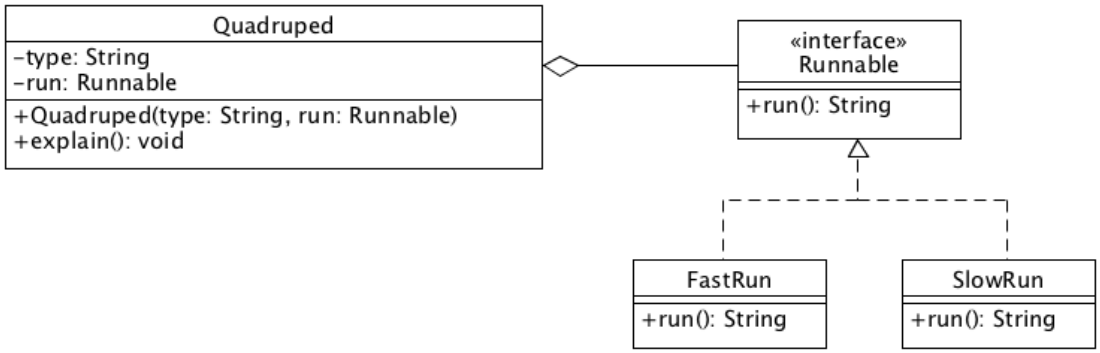
```
041  public class SlowRun implements Runnable {
042      public String run() {
043          return "ゆっくり走ります";
044      }
045  }

051  public class QuadrupedTestDrive {
052      public static void main(String[] args) {
053          Quadruped quadruped = new Quadruped("オオカミ", new FastRun());
054          quadruped.explain();
055          quadruped = new Quadruped("ゾウ", new SlowRun());
056          quadruped.explain();
057      }
058  }
```

実行例

オオカミははやく走ります

ゾウはゆっくり走ります



以下について【練習 88】から【練習 113】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.
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. 高速に運転する乗り物を生成する．名前は，vehicle とする.
02. 乗り物を走らす.
03. 低速に運転する乗り物を生成する.
04. 乗り物を走らす.
【練習 88】Vehicle のクラス図の一行目を答えなさい
【練習 89】Vehicle のクラス図の二行目を答えなさい
【練習 90】Vehicle のクラス図の三行目を答えなさい
【練習 91】Vehicle のクラス図の四行目を答えなさい
【練習 92】Drive のクラス図の一行目を答えなさい
【練習 93】Drive のクラス図の二行目を答えなさい
【練習 94】Drive のクラス図の三行目を答えなさい
【練習 95】ExpressDrive のクラス図の一行目を答えなさい
【練習 96】ExpressDrive のクラス図の二行目を答えなさい
【練習 97】ExpressDrive のクラス図の三行目を答えなさい
【練習 98】クラス間の関係を表す矢印を答えなさい
【練習 99】Vehicle のソースコードの一行目を答えなさい
【練習 100】Vehicle のソースコードの二行目を答えなさい
【練習 101】Vehicle のソースコードの三行目を答えなさい
【練習 102】Vehicle のソースコードの四行目を答えなさい
【練習 103】Vehicle のソースコードの六行目を答えなさい
【練習 104】Drive のソースコードの一行目を答えなさい
【練習 105】Drive のソースコードの二行目を答えなさい
【練習 106】Drive のソースコードの三行目を答えなさい
【練習 107】NormalDrive のソースコードの一行目を答えなさい
【練習 108】NormalDrive のソースコードの二行目を答えなさい
【練習 109】NormalDrive のソースコードの三行目を答えなさい
【練習 110】TestDrive のメインメソッドの一行目を答えなさい
【練習 111】TestDrive のメインメソッドの二行目を答えなさい
【練習 112】TestDrive のメインメソッドの三行目を答えなさい
【練習 113】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 114】から【練習 136】を解答しなさい

全ての乗客[Passenger]は乗り物の種類[type VehicleType 型]、目的地[boundFor String 型]という属性を持つ.
乗客は new するときに乗り物の種類、目的地を決める.
（Passenger のコンストラクタの引数は type、boundFor の 2 つとする）
乗客は乗る機能[getIn 引数なし void 型]、降りる機能[getOut 引数なし void 型] 並びに,
目的地に向かう機能[go 引数なし void 型]がある.
乗る機能と降りる機能は処理を委譲する.
目的地に向かう機能は「boundFor+"へ向かいます"」と表示する.

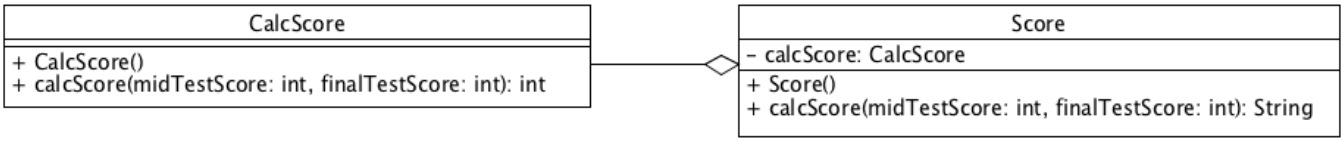
乗り物の種類[VehicleType]はインタフェースを用いる.
乗り物の種類[VehicleType]は乗る機能[getIn 引数なし void 型]、降りる機能[getOut 引数なし void 型] がある.
乗り物の種類には，具体的に，飛行機[Airplane]と電車[Train]の 2 種類がある.
飛行機に乗るときは「飛行機に乗ります」、降りるときは「飛行機から降ります」と表示する.
電車に乗るときは「電車に乗ります」、降りるときは「電車から降ります」と表示する.
（Airplane、Train は VehicleType の実装クラスとする）

TestDrive を以下のように作成する

01. 乗客を生成（飛行機を使う．目的地は大阪空港）する.
02. 飛行機に乗る.
03. 目的地に向かう.
04. 飛行機から降りる.
05. 乗客を生成（電車を使う．目的地は大阪なんば駅）とする.
06. 電車に乗る.
07. 目的地に向かう.
08. 電車から降りる.

【練習 114】Passenger のクラス図の一行目を答えなさい
【練習 115】Passenger のクラス図の二行目を答えなさい
【練習 116】Passenger のクラス図の三行目を答えなさい
【練習 117】Passenger のクラス図の四行目を答えなさい
【練習 118】Passenger のクラス図の五行目を答えなさい
【練習 119】VehicleType のクラス図の一行目を答えなさい
【練習 120】VehicleType のクラス図の二行目を答えなさい
【練習 121】VehicleType のクラス図の三行目を答えなさい
【練習 122】Airplane のクラス図の一行目を答えなさい
【練習 123】Airplane のクラス図の二行目を答えなさい
【練習 124】クラス間の関係を表す矢印を答えなさい
【練習 125】Passenger のソースコードの一行目を答えなさい
【練習 126】Passenger のソースコードの二行目を答えなさい
【練習 127】Passenger のソースコードの三行目を答えなさい
【練習 128】Passenger のソースコードの四行目を答えなさい
【練習 129】VehicleType のソースコードの一行目を答えなさい
【練習 130】VehicleType のソースコードの二行目を答えなさい
【練習 131】Train のソースコードの一行目を答えなさい
【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 18】2031 この科目は合格です
【練習 19】31 この科目は落単です
【練習 20】31 この科目は落単です
【練習 21】CalcScore
【練習 22】+ CalcScore()
【練習 23】+ calcScore(midTestScore: int, finalTestScore: int): int
【練習 24】Score
【練習 25】- calcScore: CalcScore
【練習 26】+ Score()
【練習 27】C^S



【練習 28】30A
【練習 29】254 オーブンはチーーーンと言いました
【練習 30】6254 オーブンは壊れているようで
【練習 31】1287 冷蔵庫はキンキンに冷えている
【練習 32】<<interface>>
【練習 33】Appliance
【練習 34】+ behave(): String
【練習 35】User
【練習 36】- ap: Appliance
【練習 37】+ User(ap: Appliance)
【練習 38】A^U-O.A-R.A

【練習 39】62A
【練習 40】93 現在 Mac を使用しています
【練習 41】4895Windows を使用しています

【練習 42】<<interface>>
【練習 43】OperatingSystem
【練習 44】+ usingOS(): String
【練習 45】A_OSmanagement
【練習 46】- os: OperatingSystem
【練習 47】+ A_OSmanagement()
【練習 48】+ A_OSmanagement(os: OperatingSystem)
【練習 49】M.O-O^A-W.O

【練習 50】変身！
【練習 51】（効果音のみ）
【練習 52】ファイナル・ベント
【練習 53】変身！
【練習 54】タカ・トラ・バッタ　タ・ト・バ　タトバ　タ・ト・バ
【練習 55】RiderMove
【練習 56】- rider: KamenRider
【練習 57】+ RiderMove(rider: KamenRider)
【練習 58】+ henshin(): void
【練習 59】+ finalMode(): void
【練習 60】A.K-D.K-K^R-O.K

【練習 88】Vehicle
【練習 89】- drive: Drive
【練習 90】+ Vehicle(drive: Drive)
【練習 91】+ run(): void
【練習 92】<<interface>>
【練習 93】Drive
【練習 94】+ driving(): void
【練習 95】ExpressDrive
【練習 96】+ driving(): void
【練習 97】-1
【練習 98】D^V-E.D-N.D
【練習 99】public class Vehicle {
【練習 100】private Drive drive;
【練習 101】public Vehicle(Drive drive) {
【練習 102】this.drive = drive;
【練習 103】public void run() {
【練習 104】public interface Drive {
【練習 105】public void driving();
【練習 106】}

【練習 107】public class NormalDrive implements Drive {
【練習 108】public void driving() {
【練習 109】System.out.println(“ブーン”);
【練習 110】Vehicle vehicle = new Vehicle(new ExpressDrive());
【練習 111】vehicle.run();
【練習 112】vehicle = new Vehicle(new NormalDrive());
【練習 113】vehicle.run();

【練習 114】Passenger
【練習 115】- type: VehicleType
【練習 116】- boundFor: String
【練習 117】+ Passenger(type: VehicleType, boundFor: String)
【練習 118】+ getIn(): void
【練習 119】<<interface>>
【練習 120】VehicleType
【練習 121】+ getIn(): void
【練習 122】Airplane
【練習 123】+ getIn(): void
【練習 124】A.V-T.V-V^P
【練習 125】public class Passenger{
【練習 126】private VehicleType type;
【練習 127】private String boundFor;
【練習 128】public Passenger(VehicleType type, String boundFor) {
【練習 129】public interface VehicleType{
【練習 130】public void getIn();
【練習 131】public class Train implements VehicleType {
【練習 132】Passenger passenger = new Passenger(new Airplaine(), “大阪空港”);
【練習 133】passenger.getIn();
【練習 134】passenger.go();
【練習 135】passenger.getOut();
【練習 136】passenger = new Passenger(new Train(), “大阪なんば駅”);