# **J2Kad08D「抽象クラスとインターフェイス」（入門編P.244「インタフェースの使い方」）**

J2Kad08Dに妖精を召喚して自己紹介させる処理が作成されている。妖精は光の妖精（Light）、闇の妖精（Darkness）、炎の妖精（Fire）の3種類だ。

1. 抽象クラス（Fairy）を追加して、自己紹介処理を簡略化せよ。
2. インターフェイス（IFairy）を追加して、Fairyと置き換えよ。

**①のクラス図 ②のクラス図**

|  |
| --- |
| **Darkness** |
|  |
| + intro()  + heal() |

|  |
| --- |
| **Darkness** |
|  |
| + intro()  + heal() |

|  |
| --- |
| **Darkness** |
|  |
| + intro()  + heal() |

|  |
| --- |
| <<interface>>  ***IFairy*** |
|  |
| *intro()*  *heal()* |

|  |
| --- |
| ***Fairy*** |
|  |
| *+ intro()*  *+ heal()* |

|  |
| --- |
| **Darkness** |
|  |
| + intro()  + heal() |

|  |
| --- |
| **Darkness** |
|  |
| + intro()  + heal() |

|  |
| --- |
| **Darkness** |
|  |
| + intro()  + heal() |

**課題完成時の画面（①、②どちらも同じ）**

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！！」・・・体力が回復した！

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！！」・・・体力を奪われた！

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！！」・・・どこかへ行った！

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**

# **J2Kad08C「多重継承に挑戦！」（入門編P.243「継承の限界」「多重継承をしたくなる場合とは」）**

Monsterクラスが準備されている。J2Kad08Dで作成したFairyもしくはIFairyを再利用してFairyMonsterを作りたい。

A：MonsterとFairyを多重継承してFairyMonsterを作れるかどうか確認せよ。

B：Monsterを継承、IFariyを実装してFairyMonsterを作れるかどうか確認せよ。

AまたはBでFairyMonsterを作ることができた場合、FairyMonsterのインスタンスを生成して、①FairyMonsterの全メソッドを順次呼び出す処理を作成せよ。また同じインスタンスを②Monster型の変数と③Fairy型もしくはIFairy型の変数で参照した場合に呼び出し可能な全メソッドを順次呼び出す処理も作成せよ（**リスト1**）。

A：Fairyを継承

|  |
| --- |
| ***Fairy*** |
|  |
| *+ intro()*  *+ heal()* |

|  |
| --- |
| **Monster** |
| # name |
| + Monster(name : String)  + intro()  + walk()  + sleep() |

|  |
| --- |
| <<interface>>  ***IFairy*** |
|  |
| *+ intro()*  *+ heal()* |

|  |
| --- |
| **FairyMonster** |
|  |
| + FairyMonster(name : String)  + heal() |

B：IFairyを実装

**FairyMonsterクラスで実装するメソッド**

|  |  |
| --- | --- |
| メソッド | 仕様 |
| public FairyMonster(String name) | nameをそのまま引数にして、Monsterのコンストラクタを呼ぶ。 |
| public void heal() | 「誰かのけがを治した！」と表示する。 |

**リスト1：メソッドの呼び出し（ファイル「J2Kad08C.java」） 課題完成時の画面**

public class J2Kad08C {

public static void main(String[] args) {

// ①FairyMonsterとして参照

FairyMonster fm = new FairyMonster("ピクシー");

**fmのFairyMonsterに属するメソッドを実行**

System.out.println();

// ②Monsterとして参照

Monster m = fm;

**fmのMonsterに属するメソッドを順次実行**

System.out.println();

// ③FairyまたはIFairyとして参照

Fairy f = fm; // Bの場合はIFairy

**fmのFairyまたはIFairyに属するメソッドを順次実行**

f.heal();

}

}

ピクシーがやってきた！

おいらの名前はピクシー。よろしくね！

てくてく・・・

ぐうぐう・・・

誰かのけがを治した！！

おいらの名前はピクシー。よろしくね！

ぐうぐう・・・

てくてく・・・

おいらの名前はピクシー。よろしくね！

誰かのけがを治した！！

# **J2Kad08B「発明！電話のできるカメラ！！」（入門編P.249「複数のインタフェースの実装」）**

世界にはばたくECCホームエレクトロニクスが電話のできるカメラを発明した！その名もスマートフォン！じゃなくてカメラフォン！さらに何かのアプリもゲットできるらしい（スマホじゃないよ）。

1. カメラ機能のインターフェイス（ICamera）と電話機能のインターフェイス（IPhone）を定義し、CameraPhoneクラスを作成せよ。
2. CameraPhoneのインスタンスを生成し、「0：全部」「1：カメラ」「2：電話」として操作する（メソッドを順次呼び出す）処理を作成せよ。

|  |
| --- |
| <<interface>>  ***ICamera*** |
|  |
| *takePicture()*  *displayImage()* |

|  |
| --- |
| <<interface>>  ***IPhone*** |
|  |
| *call()* |

|  |
| --- |
| **CameraPhone** |
|  |
| + CameraPhone()  + getApp()  + takePicture()  + displayImage()  + call() |

**CameraPhoneクラスのメソッド（ICameraとIPhoneを実装）**

|  |  |  |
| --- | --- | --- |
| メソッド | 仕様 | 備考 |
| コンストラクタ | 「カメラフォンを作った！」と表示する。 | CameraPhoneオリジナル |
| public void getApp() | 「何かのアプリを手に入れました！」と表示する。 |
| public void takePicture() | 「何かの写真を撮りました！」と表示する。 | ICameraのオーバーライド |
| public void displayImage() | 「何かの写真を表示しました！」と表示する。 |
| public void call() | 「誰かに電話をかけました！」と表示する。 | IPhoneのオーバーライド |

**課題完成時の画面**

カメラフォンを作った！

どの機能を使いますか？（0：全部、1：カメラ、2：電話、-1：もうあきた）＞**0**

何かの写真を撮りました！

何かの写真を表示しました！

誰かに電話をかけました！

何かのアプリを手に入れました！

どの機能を使いますか？（0：全部、1：カメラ、2：電話、-1：もうあきた）＞**1**

何かの写真を撮りました！

何かの写真を表示しました！

どの機能を使いますか？（0：全部、1：カメラ、2：電話、-1：もうあきた）＞**2**

誰かに電話をかけました！

どの機能を使いますか？（0：全部、1：カメラ、2：電話、-1：もうあきた）＞**-1**

# **J2Kad08A「図形を描こう！」**

図形の描画と消去を行う処理が準備されている。Triangleクラスで二等辺三角形、Rectangleクラスで長方形の描画ができる。またEraserクラスで図形の消去ができる。

1. インターフェイスIShapeを定義して、図形描画処理を簡略化せよ。必要であれば各描画クラス（Triangle、Rectangle、Eraser）を修正してもよい。
2. 正方形を描画するSquareクラスを追加せよ。

|  |
| --- |
| **Eraser** |
| ･･･ |
| Eraser(･･･)  erase(c : Canvas) |

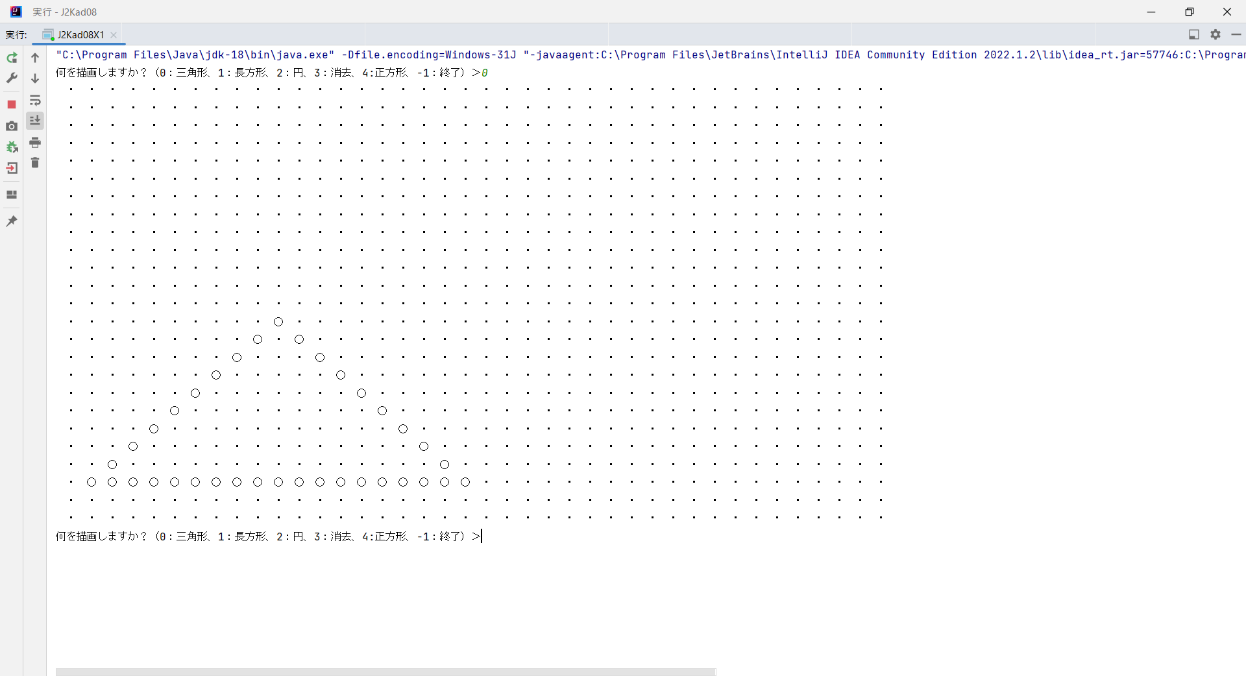
|  |
| --- |
| **Rectangle** |
| ･･･ |
| Rectangle(･･･)  drawRectangle(c : Canvas) |

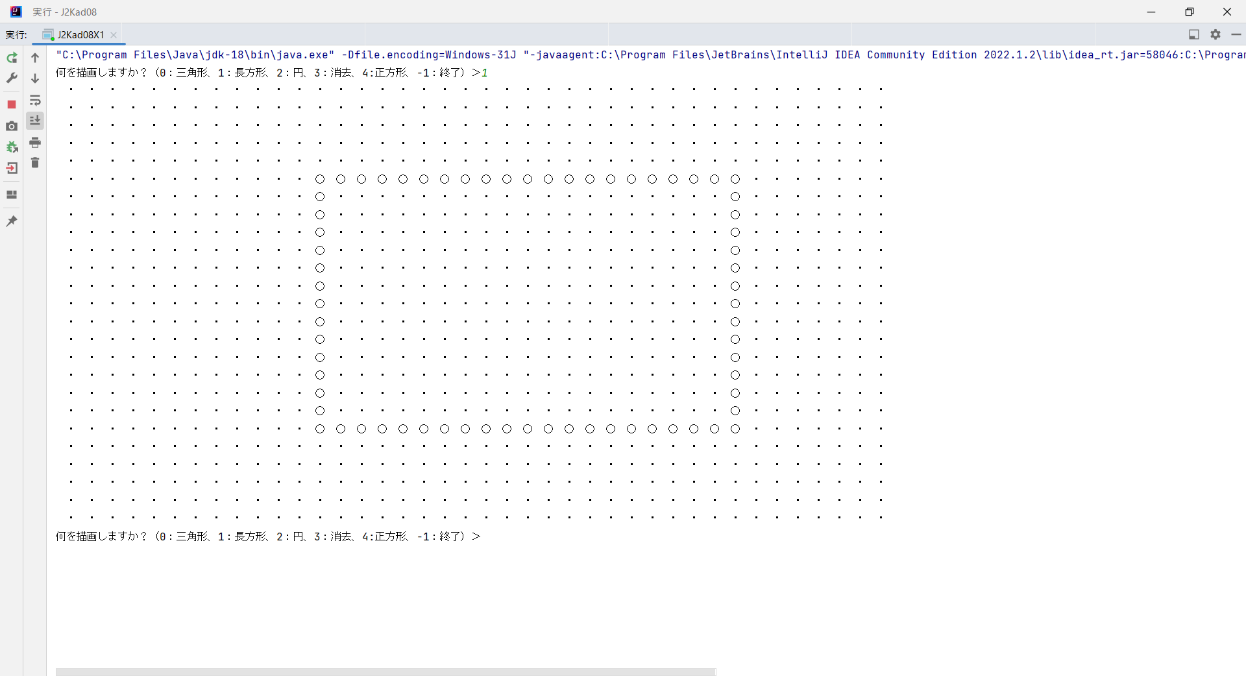
|  |
| --- |
| **Triangle** |
| ･･･ |
| Triangle(･･･)  drawTriangle(c : Canvas) |

|  |  |  |
| --- | --- | --- |
| 入力 | 描画する図形 | 使用するクラス |
| 0 | 二等辺三角形を描画する。左下座標：(1, 22)、底辺の長さ：18、高さ：9 | Triangle |
| 1 | 長方形を描画する。左上座標：(12, 5)、横幅：20、高さ：14 | Rectangle |
| 2 | 図形を消去する | Eraser |
| 3 | 正方形を描画する。左上座標：(1, 1)、辺の長さ：16 | Square |

ヒント1：ポリモーフィズムを使うためにはメソッドの仕様を同じにする必要がある。

ヒント2：正方形（Square）は長方形（Rectangle）を継承すれば作成できる。

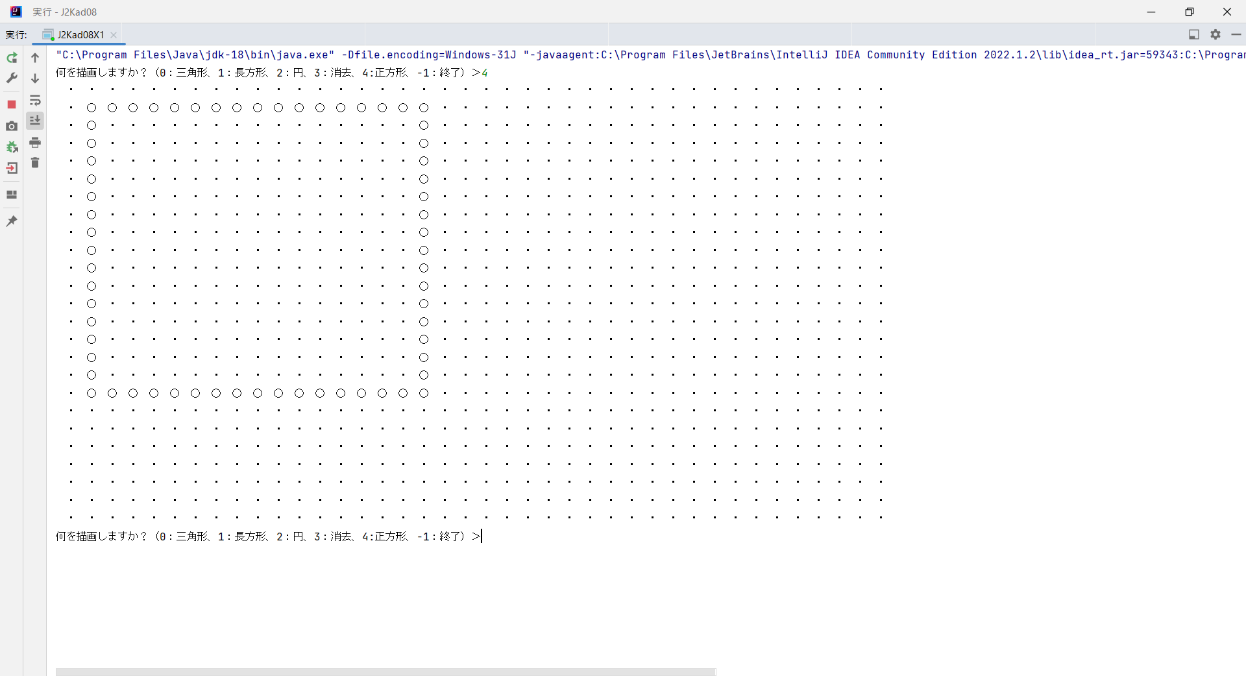
**課題完成時の画面**



長方形

三角形

長方形



正方形

# **J2Kad08S「新型エアコンの開発」**

世界的な家電メーカーECCホームエレクトロニクスがエアコン操作の画期的なインターフェイスIAirconを発表した！ボタンを押すだけで「電源ON/OFF」「冷房」「暖房」「送風」を切り換えられるというスグレモノだ！このインターフェイスで操作するエアコンを開発し（NewAriconクラス）、IAirconを使って動作確認を行え。

**NewAriconクラスでの実装（IAirconに基づいて実装する）**

|  |  |
| --- | --- |
| メンバ | 仕様 |
| int mode | 運転モード（0：冷房、1：暖房、2：送風）、初期値：冷房 |
| boolean power | 電源（true：ON、false：OFF）、初期値：OFF |
| コンストラクタ | modeとpowerに初期値を設定する |
| void showData() | 電源ONのとき「電源：ON、運転モード～」（～は現在のモード）と表示する。  電源OFFのとき「電源：OFF」と表示する。 |
| void powerOnOff() | 電源ON/OFFを切り換えて「電源を入れました！」または「電源を切りました！」と表示する。 |
| void toCool() | 電源OFFのとき「電源が入っていません！」と表示する。  電源ONのとき運転モードを冷房にして「冷房に切り換えました！」と表示する。 |
| void toHeat() | 電源OFFのとき「電源が入っていません！」と表示する。  電源ONのとき運転モードを暖房にして「暖房に切り換えました！」と表示する。 |
| void toBlow() | 電源OFFのとき「電源が入っていません！」と表示する。  電源ONのとき運転モードを送風にして「送風に切り換えました！」と表示する。 |

**課題完成時の画面**

|  |
| --- |
| <<interface>>  ***IAircon*** |
|  |
| *showData()*  *powerOnOff()*  *toCool()*  *toHeat()*  *toBlow()* |

|  |
| --- |
| **NewAircon** |
| mode  power |
| NewAircon()  showData()  powerOnOff()  toCool()  toHeat()  toBlow() |

ECCホームエレクトロニクスが新型エアコンを開発した！

電源：OFF

どうしますか？（0：電源ON/OFF、1：冷房、2：暖房、3：送風、-1：終了）＞**0**

電源を入れました！

電源：ON、モード：冷房

どうしますか？（0：電源ON/OFF、1：冷房、2：暖房、3：送風、-1：終了）＞**1**

冷房に切り換えました！

電源：ON、モード：冷房

どうしますか？（0：電源ON/OFF、1：冷房、2：暖房、3：送風、-1：終了）＞**2**

暖房に切り換えました！

電源：ON、モード：暖房

どうしますか？（0：電源ON/OFF、1：冷房、2：暖房、3：送風、-1：終了）＞**3**

送風に切り換えました！

電源：ON、モード：送風

どうしますか？（0：電源ON/OFF、1：冷房、2：暖房、3：送風、-1：終了）＞**0**

電源を切りました！

電源：OFF

どうしますか？（0：電源ON/OFF、1：冷房、2：暖房、3：送風、-1：終了）＞**1**

電源が入っていません！

# **J2Kad08X「旧式エアコン（Adapterパターン）」**

ECCホームエレクトロニクスには「つまみ」を回して操作する旧式エアコン（OldAircon）の在庫が山ほどあった！これをIAirconインターフェイスで操作できるようにして売りさばきたい！！OldAirconをIAirconで操作できるように処理を作成せよ。ただしOldAirconはすでに製品として出来上がってしまっているのでOldAirconに手を加えることはできないものとする（もちろんIAirconも変更不可）。

切

冷房

暖房

送風

**OldAriconクラスの仕様（変更不可）**

「つまみ」を回して切り換える。

[切]⇔[冷房]⇔[暖房]⇔[送風]

|  |  |
| --- | --- |
| メンバ | 仕様 |
| コンストラクタ | つまみを[切]（OFF）にする。 |
| int getKnob() | 現在のつまみ（ノブ）の位置を返す。  （[切]：OFF、[冷房]：COOL、[暖房]：HEAT、[送風]：BLOW） |
| void showData() | 現在のつまみの位置（運転モード）を表示する。 |
| void turnRight() | つまみを右（時計回り）へひとつ回す。 |
| void turnLeft() | つまみを左（反時計回り）へひとつ回す。 |

ヒント1：インターフェイスは多重継承（←検索）OK。

|  |
| --- |
| <<interface>>  ***IAircon*** |

|  |
| --- |
| **OldAircon** |

|  |
| --- |
| **OldAirconAdapter** |

ヒント2：Adapterパターン（継承によるアダプター）を検索。

これを考えること

変更不可

**課題完成時の画面**

新型エアコンと同じ操作で旧式エアコンを動かします！

ただいま[切]です。

どうしますか？（0：電源ON/OFF、1：冷房、2：暖房、3：送風、-1：終了）＞**0**

つまみを右に回した！

ただいま[冷房]です。

どうしますか？（0：電源ON/OFF、1：冷房、2：暖房、3：送風、-1：終了）＞**2**

つまみを右に回した！

ただいま[暖房]です。

どうしますか？（0：電源ON/OFF、1：冷房、2：暖房、3：送風、-1：終了）＞**3**

つまみを右に回した！

ただいま[送風]です。

どうしますか？（0：電源ON/OFF、1：冷房、2：暖房、3：送風、-1：終了）＞**0**

つまみを左に回した！

つまみを左に回した！

つまみを左に回した！

ただいま[切]です。

どうしますか？（0：電源ON/OFF、1：冷房、2：暖房、3：送風、-1：終了）＞**-1**

つまみが[切]のときに

電源ON/OFFをすると

[冷房]で動くものとする。