# **J2Kad02D1「ピカチュウクラス！」（入門編P.134「クラスの宣言」）**

J2Kad02D1にJ2Kad01C完成版相当のプログラムが準備されている。ピカチュウに関する処理をPikaクラス（新規ファイル「Pika.java」）として独立させよ。

**リスト1：「ピカチュウ現る！」（ファイル「J2Kad02D1.java」）**

Pikaクラスへ持って行く

public class J2Kad02D1 {

public static String name = "ピカチュウ"; // 名前

public static int hp = 20; // 体力

public static void showData() {…}

public static void walk() {…}

public static void sleep() {…}

public static void main(String[] args) {

showData();

System.out.println(name + "を散歩させます！");

walk();

walk();

walk();

showData();

System.out.println(name + "を眠らせます！");

sleep();

sleep();

sleep();

showData();

}

}

Pikaクラスのメンバを

使うように変更する

**課題完成時の画面**

ぼくの名前はピカチュウ、HPは20だよ！

ピカチュウを散歩させます！

てくてく・・・

てくてく・・・

てくてく・・・

ぼくの名前はピカチュウ、HPは17だよ！

ピカチュウを眠らせます！

ぐうぐう・・・

ぐうぐう・・・

ぐうぐう・・・

ぼくの名前はピカチュウ、HPは20だよ！

# **J2Kad02D2「ヤドンクラス！」**

ヤドン（Yadonクラス）を新規作成し、散歩させて眠らせる処理を作成せよ。

**Yadonクラスの仕様（新規ファイル「Yadon.java」） 課題完成時の画面**

フィールド、メソッドともにPikaクラスと同じ。ただしnameの初期値は「ヤドン」、hpの初期値は30とする。

ぼくの名前はヤドン、HPは30だよ！

ヤドンを散歩させます！

てくてく・・・

てくてく・・・

てくてく・・・

ぼくの名前はヤドン、HPは27だよ！

ヤドンを眠らせます！

ぐうぐう・・・

ぐうぐう・・・

ぐうぐう・・・

ぼくの名前はヤドン、HPは30だよ！

**mainメソッドの仕様（J2Kad02D2クラスに作成）**

J2Kad02D1と同じ。ただしPikaクラスの代わりにYadonクラスを使う。

# **J2Kad02C「モンスタークラス！」 （P.138「インスタンスの生成」、P.139「インスタンス変数」、P.171「インスタンスメソッドとは」）**

どのモンスターでも使えるMonsterクラスを作成し、ディアルガ（体力：1000）とコイキング（体力：1）を散歩させる処理を作成せよ。

**Monsterクラスの仕様（新規ファイル「Monster.java」）**

|  |  |  |
| --- | --- | --- |
|  | 書式（staticはつけないこと） | 説明 |
| フィールド | public String name | 名前（初期値：設定しない） |
| public int hp | 体力（初期値：設定しない） |
| メソッド | public void setData(String n, int h) | 名前にnを、体力にhを代入する。 |
| public void showData() | 名前と体力を表示する。「ぼくの名前は～！HPはxxだよ！」 |
| public void walk() | 体力が0以下のとき「つかれて歩けないよ～」と表示する。  そうでないとき「てくてく・・・」と表示し、体力を1減らす。 |
| public void sleep() | 眠らせる。「ぐうぐう・・・」と表示したのち、体力を1増やす。 |

**mainメソッドの仕様（J2Kad02Cクラスに作成）**

（課題完成時の画面を参考に作成すること）

**課題完成時の画面**

ぼくの名前はディアルガ、HPは1000だよ！

ぼくの名前はコイキング、HPは1だよ！

ディアルガを散歩させます！

てくてく・・・

てくてく・・・

てくてく・・・

ぼくの名前はディアルガ、HPは997だよ！

コイキングを散歩させます！

てくてく・・・

つかれて歩けないよ～

つかれて歩けないよ～

ぼくの名前はコイキング、HPは0だよ！

# **J2Kad02B「基本型と参照型」（入門編P.142「参照型」）**

以下の処理を作成し、動作確認せよ。

**作成するメソッド（J2Kad02Bクラス）**

|  |  |
| --- | --- |
| 書式 | 仕様 |
| public static void addInt(int x) | 引数xに5を加算し、「xに5を加算しました！」と表示する。 |
| public static void addArray(int[] b) | 配列bの全要素に5を加算し、「b[#]に5を加算しました！」と表示する。  （#は要素番号） |

**mainメソッドの仕様①**

1. int型変数x（初期値：10）を宣言する。
2. xを引数にしてaddIntメソッドを呼び出す。
3. xの値を表示する。

**mainメソッドの仕様②**

1. int配列a（要素数3）を生成する。
2. a[0]に0、a[1]に1、a[2]に2を代入する。
3. 配列aを引数にしてaddArrayメソッドを呼び出す。
4. 配列aの各要素の値を表示する。

**課題完成時の画面**

int型変数xの値はaddIntメソッドの前後で変わらないが、配列aの値はaddArrayメソッドの前後で変化している。

x：10

xに5を加算しました！

x：10

a：10 20 30

b[0]に5を加算しました！

b[1]に5を加算しました！

b[2]に5を加算しました！

a：15 25 35

仕様①

仕様②

# **J2Kad02A「参照渡し」**

Monsterクラスを使ってピカチュウ（体力：10～19）とイワンコ（体力：10～19）を生成、useMonsterメソッド（新規作成）を使って動かせ（**課題完成時の画面**を参照）。

**作成するメソッド（J2Kad02Aクラス）**

|  |  |
| --- | --- |
| 書式 | 仕様 |
| public static void useMonster(Monster m) | モンスターmに対して以下の処理を行う。   1. 3回散歩させて、データを表示する。 2. 3回眠らせて、データを表示する。 |

**mainメソッドの仕様**

1. ピカチュウとイワンコを生成し、データを表示する。
2. ピカチュウを動かす（useMonsterメソッドを呼び出す）。
3. イワンコを動かす（useMonsterメソッドを呼び出す）。

**課題完成時の画面**

ぼくの名前はピカチュウ、HPは18だよ！

ぼくの名前はイワンコ、HPは18だよ！

ピカチュウを散歩させます！

てくてく・・・

てくてく・・・

てくてく・・・

ぼくの名前はピカチュウ、HPは15だよ！

ピカチュウを眠らせます！

ぐうぐう・・・

ぐうぐう・・・

ぐうぐう・・・

ぼくの名前はピカチュウ、HPは18だよ！

イワンコを散歩させます！

てくてく・・・

てくてく・・・

てくてく・・・

ぼくの名前はイワンコ、HPは15だよ！

イワンコを眠らせます！

ぐうぐう・・・

ぐうぐう・・・

ぐうぐう・・・

ぼくの名前はイワンコ、HPは18だよ！

# **J2Kad02S「そうだ！ECC銀行へ行こう！！②」**

銀行口座を表すAccountクラスを作成し、のび太とスネ夫が信頼と実績のECC銀行へ行く処理を作成せよ。必要なフィールドは各自で考えること。

**Accountクラスの仕様（新規作成）**

|  |  |  |
| --- | --- | --- |
|  | 書式 | 説明 |
| フィールド | public String name | 口座名義 |
| public int accountNumber | 口座番号（7桁） |
| public int money | 預金残高 |
| public int secretNumber | 暗証番号（4桁） |
| メソッド | public void setData(String n,  int a, int m, int s) | n：口座名義、a：口座番号、m：預金残高、s：暗証番号を  対応するフィールドに設定する。 |
| public void showData() | 口座情報（口座名義、口座番号、口座残高）を表示する。 |

**作成するメソッド（J2Kad02Sクラス）**

|  |  |
| --- | --- |
| 書式 | 仕様 |
| public static void gotoECCBank(Account account) | 処理の対象となる銀行口座を引数accountとして受け取る。  （処理内容はJ2Kad01Sと同じ） |
| public static void deposit(Account account) |
| public static void withdraw(Account account) |

**mainメソッドの仕様**

1. のび太とスネ夫の銀行口座を作成する（値は各自で設定する）。
2. のび太またはスネ夫を選択し、gotoECCBankメソッドを呼び出す。

**課題完成時の画面**

基本的にJ2Kad01Sと同じ処理。銀行へ行く人（口座）を選択する処理が追加されている。

なお、左ではスネ夫のデータは以下の通り。

・口座名義：スネ夫

・口座番号8901234

・預金残高：10000000

・暗証番号：5678

そうだ！銀行へ行こう！！

誰が行きますか？（1：のび太、2：スネ夫、-1：誰もいかない）＞**2**

信頼と実績のECC銀行へようこそ！

口座名義：スネ夫

口座番号：8901234

預金残高：10000000円

どうしますか？（1：預ける、2：引き出す、-1：帰る）＞**2**

暗証番号を入力してください＞**5678**

いくら引き出しますか？＞**5000000**

口座名義：スネ夫

口座番号：8901234

預金残高：5000000円

どうしますか？（1：預ける、2：引き出す、-1：帰る）＞**-1**

ありがとうございました！

誰が行きますか？（1：のび太、2：スネ夫、-1：誰もいかない）＞**-1**

# **J2Kad02X「スタック！②」**

J2Kad01Xのスタック操作に関する処理をStackクラスとして取り出せ。格納できるデータ数は10個とする。なお、スタックオーバーフローなどの不具合が発生しないように、スタックのデータ数をチェックする処理も追加すること。

**Stackクラスのメソッド（フィールドは各自で考えること）**

|  |  |
| --- | --- |
| 書式 | 説明 |
| public void push(int data) | スタックにデータ（data）を格納する。 |
| public int pop() | スタックからデータを取り出し値を返す。 |

**mainメソッドの仕様（課題完成時の画面を参考）**

1. スタックを生成し、「-1：終了」が選択されるまで以下の処理を行う。
2. 「1：push」を選択するとスタックに0～99までの値（乱数で決定）を格納する。ただしこれ以上データを格納できないときは「スタックがいっぱいです！」と表示する。
3. 「2：pop」を選択するとスタックからデータを取り出す。ただしデータがないときは「データがありません！」と表示する。

**課題完成時の画面 （続き）**

スタック操作をします！

どうしますか？（1：push、2：pop、-1：終了）＞**1**

stack：4

どうしますか？（1：push、2：pop、-1：終了）＞**1**

stack：4 51

どうしますか？（1：push、2：pop、-1：終了）＞**1**

stack：4 51 26

どうしますか？（1：push、2：pop、-1：終了）＞**1**

stack：4 51 26 34

どうしますか？（1：push、2：pop、-1：終了）＞**1**

stack：4 51 26 34 71

どうしますか？（1：push、2：pop、-1：終了）＞**1**

stack：4 51 26 34 71 87

…

　（中略）

…

どうしますか？（1：push、2：pop、-1：終了）＞**1**

stack：4 51 26 34 71 87 18 5 17 94

どうしますか？（1：push、2：pop、-1：終了）＞**1**

スタックがいっぱいです！

stack：4 51 26 34 71 87 18 5 17 94

どうしますか？（1：push、2：pop、-1：終了）＞**2**

94を取り出しました！

stack：4 51 26 34 71 87 18 5 17

どうしますか？（1：push、2：pop、-1：終了）＞**2**

17を取り出しました！

stack：4 51 26 34 71 87 18 5

どうしますか？（1：push、2：pop、-1：終了）＞**2**

5を取り出しました！

stack：4 51 26 34 71 87 18

…

　（中略）

…

どうしますか？（1：push、2：pop、-1：終了）＞**2**

51を取り出しました！

stack：4

どうしますか？（1：push、2：pop、-1：終了）＞**2**

4を取り出しました！

stack：

どうしますか？（1：push、2：pop、-1：終了）＞**2**

データがありません！

stack：

どうしますか？（1：push、2：pop、-1：終了）＞**-1**