

● J2Kad02D1 「ピカチュウクラス！」

J2Kad01C で出現したピカチュウを Pika クラス（新規作成）として実装せよ。

Pika クラスの仕様（新規作成）

	書式	説明
フィールド	public static String name	名前（初期値：ピカチュウ）
	public static int hp	体力（初期値：20）
メソッド	public static void showData()	名前と体力を表示する。「ぼくの名前は～！HP は xx だよ！」
	public static void walk()	散歩させる。「てくてく・・・」と表示したのち、体力を 1 減らす。
	public static 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（J2Kad02D1 クラスに作成）

- ① 「～が現れた！」（～は Pika クラスの name）と表示し、データ表示する。
- ② 3 回散歩させて、データ表示する。
- ③ 3 回眠らせて、データ表示する。

● 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（新規作成）

（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）

main メソッドの仕様（J2Kad02D2 クラスに作成）

（Pika クラスの代わりに Yadon クラスを使う。他は J2Kad02D1 と同じ）

課題完成時の画面（J2Kad02D1）

ピカチュウが現れた！
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！

課題完成時の画面（J2Kad02D2）

ヤドンが現れた！
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！

● J2Kad02C 「モンスタークラス！」

複数のモンスターで使える 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（新規作成）

	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

```
ディアルガが現れた！
ぼくの名前はディアルガ、HP は 1000 だよ！
コイキングが現れた！
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」

以下の処理を作成し、動作確認せよ（Monster クラスは J2Kad02C で作成したものを使用）。

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）
public static void addHp(Monster m)	モンスター m の hp に 5 を加算し、「～の hp に 5 を加算しました！」と表示する。 （～はモンスターの名前）

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

- ④ int 配列 a（要素数 3）を生成する。
- ⑤ a[0]に 0、a[1]に 1、a[2]に 2 を代入する。
- ⑥ 配列 a を引数にして addArray メソッドを呼び出す。
- ⑦ 配列 a の各要素の値を表示する。

main メソッドの仕様③

- ⑧ Monster クラスのインスタンスを生成する。
- ⑨ 名前に「ピカチュウ」、体力に 20 を設定する。
- ⑩ ピカチュウを引数にして addHp メソッドを呼び出す。
- ⑪ ピカチュウのデータを表示する。

課題完成時の画面

x に 5 を加算しました！
x : 10
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 : 5 6 7
ピカチュウの HP に 5 を加算しました！
ぼくの名前はピカチュウ、HP は 25 だよ！

仕様①

仕様②

仕様③

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 と Monster クラスの値は addArray メソッド、addHp メソッドの前後で変化している。

● 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、どちらかを選択して散歩させる・眠らせる処理を作成せよ（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を散歩させたり眠らせたりする。 「1：散歩する、2：眠る、-1：やめる」と表示し、選択した処理を行う。 (マイナスの値が入力されるまで繰り返す)

課題完成時の画面

```
ピカチュウが現れた！
ぼくの名前はピカチュウ、HP は 18 だよ！
イワンコが現れた！
ぼくの名前はイワンコ、HP は 19 だよ！
どのモンスターを使いますか？（1：ピカチュウ、2：イワンコ、-1：やめる）>1
ぼくの名前はピカチュウ、HP は 18 だよ！
どうしますか？（1：散歩する、2：眠る、-1：やめる）>1
てくてく・・・
ぼくの名前はピカチュウ、HP は 17 だよ！
どうしますか？（1：散歩する、2：眠る、-1：やめる）>2
ぐうぐう・・・
ぼくの名前はピカチュウ、HP は 18 だよ！
どうしますか？（1：散歩する、2：眠る、-1：やめる）>-1

どのモンスターを使いますか？（1：ピカチュウ、2：イワンコ、-1：やめる）>2
ぼくの名前はイワンコ、HP は 19 だよ！
どうしますか？（1：散歩する、2：眠る、-1：やめる）>1
てくてく・・・
ぼくの名前はイワンコ、HP は 18 だよ！
どうしますか？（1：散歩する、2：眠る、-1：やめる）>2
ぐうぐう・・・
ぼくの名前はイワンコ、HP は 19 だよ！
どうしますか？（1：散歩する、2：眠る、-1：やめる）>-1

どのモンスターを使いますか？（1：ピカチュウ、2：イワンコ、-1：やめる）>-1
```

● 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

```

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

```

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

● 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。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 createStack(int size)	size 個までデータを格納できるスタックを作る
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。
public int getData(int i)	スタックの i 番目のデータを返す。
public int size()	スタックに格納されているデータ数を返す。
public boolean isEmpty()	スタックが空なら true、データがあるなら false を返す。
public boolean isFull()	スタックが一杯なら true、まだデータを格納できるのなら false を返す。

作成するメソッド (J2Kad02X クラス)

書式	仕様
public static void showData(Stack s)	スタックに格納されているデータを表示する。「stack : データ 1 データ 2 …」
public static void pushData(Stack s)	スタックに 0～99 までの値 (乱数で決定) を 3 つ格納する。 ただしスタックが一杯のときは「スタックがいっぱいです！」と表示して終了。
public static void popData(Stack s)	スタックからデータを 1 つ取り出して「xx を取り出しました！」と表示する。 ただしスタックにデータがないときは「データがありません！」と表示して終了。

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、要素数 10 を設定する。
- ② 以下、課題完成時の画面を参考に作成する。

課題完成時の画面

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 : 56 12 27 どうしますか？ (1 : push、2 : pop、-1 : 終了) >1 stack : 56 12 27 91 99 22 どうしますか？ (1 : push、2 : pop、-1 : 終了) >1 stack : 56 12 27 91 99 22 41 51 14 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 : 56 12 27 91 99 22 41 51 14 91 どうしますか？ (1 : push、2 : pop、-1 : 終了) >2 91 を取り出しました！ stack : 56 12 27 91 99 22 41 51 14
--

(続き)

: (中略) : どうしますか？ (1 : push、2 : pop、-1 : 終了) >2 12 を取り出しました！ stack : 56 どうしますか？ (1 : push、2 : pop、-1 : 終了) >2 56 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
