* **課題作成用ファイルは課題フォルダ内の「J2Kad01」フォルダ（プロジェクトフォルダ）に準備しているので、このフォルダをIntelliJのアイコンにドラッグして作成すること。**

# **J2Kad01D1「ローカル変数」（入門編P.81「変数のスコープ」）**

引数xに5を加算して表示するメソッドadd5を作成し、mainメソッドから3回呼び出して動作を確認せよ。

**リスト1：「ローカル変数」（ファイル「J2Kad01D1.java」） 課題完成時の画面**

xに5を足しました！

xに5を足しました！

xに5を足しました！

x：10

public class J2Kad01D1 {

public static void add5(**int x**) {

**xに5を加算する**

System.out.println("xに5を足しました！");

}

public static void main(String[] args) {

**int x = 10;**

**add5を3回呼び出す**

System.out.println("x：" + x);

}

}

「5を足しました！」と表示されるが、

xの値は変わらない。

# **J2Kad01D2「フィールド」（P.134「クラスの宣言」、P.176「クラス変数」）※J2Kad01D1をコピー**

J2Kad01D1のmainメソッドの変数xをフィールドとして宣言し、値が更新されるようにadd5メソッドを修正せよ。

**リスト1：「フィールド」（ファイル「J2Kad01D2.java」） 課題完成時の画面**

public class J2Kad01D2 {

**ここで変数xを宣言する**

public static void add5() {

**xに5を加算する**

System.out.println("xに5を足しました！");

}

public static void main(String[] args) {

**//**int x = 10;

**add5を3回呼び出す**

System.out.println("x：" + x);

}

}

xに5を足しました！

xに5を足しました！

xに5を足しました！

x：25

フィールドとして宣言するとxの値が更新される

# **J2Kad01C「ピカチュウ現る！」**

ピカチュウを散歩させて眠らせる処理を作成せよ。仕様は以下の通り。

**フィールドの宣言**

|  |  |  |
| --- | --- | --- |
| データ型 | 名前 | 説明 |
| String | name | モンスターの名前、初期値：ピカチュウ |
| int | hp | モンスターの体力、初期値：20 |

**メソッド**

|  |  |
| --- | --- |
| 書式 | 仕様 |
| public static void showData() | モンスターのデータ（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() | モンスターを散歩させる。「てくてく・・・」と表示したのち、体力を1減らす。 |
| public static void sleep() | モンスターを眠らせる。「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**

1. 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。

**課題完成時の画面**

ぼくの名前はピカチュウ！HPは20だよ！

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！

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！

# **J2Kad01B「ライチュウ現る！」（入門編P.180「クラスメソッド」）**

ライチュウを散歩させて眠らせる処理を作成せよ。なおフィールドおよびメソッドはJ2Kad01Cのものを使うこと。仕様は以下の通り。

**mainメソッドの仕様**

1. J2Kad01Cのデータを表示する。
2. 「～が進化した！」（～はJ2Kad01Cのname）と表示し、J2Kad01Cのフィールドnameに「ライチュウ」、hpに40を代入する。
3. J2Kad01CのshowDataメソッドを使って、データを表示する。
4. 行動（1：散歩する、2：眠る、-1：終了）を選択する。
5. -1（マイナスの値）が入力されたら、終了。
6. 1が選択されたらJ2Kad01Cのwalkメソッド、2が選択されたらJ2Kad01Cのsleepメソッドを呼び出し、③へ戻る。

**課題完成時の画面**

ぼくの名前はピカチュウ！HPは20だよ！

ピカチュウが進化した！！

ぼくの名前はライチュウ！HPは40だよ！

どうしますか？（1：散歩する、2：眠る、-1：終了）＞**1**

てくてく・・・

ぼくの名前はライチュウ！HPは39だよ！

どうしますか？（1：散歩する、2：眠る、-1：終了）＞**2**

ぐうぐう・・・

ぼくの名前はライチュウ！HPは40だよ！

どうしますか？（1：散歩する、2：眠る、-1：終了）＞**-1**

# **J2Kad01A「2回目のおつかい」**

のび太がハンバーガー（200円）とドーナツ（120円）とコーヒー（350円）を買いに行く処理を作成せよ。なお所持金の初期値は1000円とし、必要なフィールドは各自で考えること。

**メソッド**

|  |  |
| --- | --- |
| 書式 | 仕様 |
| public static void showMoney() | 現在の所持金を表示する。「所持金はxx円です。」 |
| public static void gotoECCBurger() | ECCバーガーでハンバーガーを買う。  「ECCバーガーに着きました」「ハンバーガー200円を買いました」と表示して  所持金を200減らす。 |
| public static void gotoECCDonut() | ECCドーナツでドーナツを買う。  「ECCドーナツに着きました」「ドーナツ120円を買いました」と表示して  所持金を120減らす。 |
| public static void gotoECCCoffee() | ECCコーヒーでコーヒーを買う。  「ECCコーヒーに着きました」「コーヒー350円を買いました」と表示して  所持金を350減らす。 |

**課題完成時の画面**

2回目のおつかい！

のび太がハンバーガーとドーナツとコーヒーを買いに行きます！

所持金は1000円です。

ECCバーガーに着きました！

ハンバーガー200円を買いました！

所持金は800円です。

ECCドーナツに着きました！

ドーナツ120円を買いました！

所持金は680円です。

ECCコーヒーに着きました！

コーヒー350円を買いました！

所持金は330円です。

# **J2Kad01S「そうだ！ECC銀行へ行こう！！」**

のび太が信頼と実績の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。

**口座情報**

|  |  |
| --- | --- |
| 項目 | 初期値 |
| 口座名義 | のび太 |
| 口座番号 | 7桁の整数（各自で決める） |
| 口座残高 | 0円 |
| 暗証番号 | 4桁の整数（各自で決める） |

**メソッド**

|  |  |
| --- | --- |
| 書式 | 仕様 |
| public static void gotoECCBank() | 銀行の処理。口座情報を表示し、1：預け入れ、2：引き出し、-1：帰るの処理をする。 |
| public static void showAccount() | 口座情報（口座名義、口座番号、口座残高）を表示する。 |
| public static void deposit() | 預け入れ。  預け入れ金額を入力させ、口座残高に加算する。 |
| public static void withdraw() | 引き出し。   1. 暗証番号を入力させ、間違っていたら「番号が違います！」と表示して終了。 2. 引き出し金額を入力させ、預金残高より大きかったら「残高不足です！」と表示して終了。 3. 口座残高から引き出し金額を減らす。 |

**mainメソッドの仕様**

1. 「そうだ！銀行へ行こう！！」と表示し、gotoECCBankメソッドを呼び出す。

**課題完成時の画面 （続き）**

そうだ！銀行へ行こう！！

信頼と実績のECC銀行へようこそ！

口座名義：のび太

口座番号：1234567

預金残高：0円

どうしますか（1：預ける、2：引き出す、-1：帰る）＞**1**

いくら預けますか？＞**1500**

口座名義：のび太

口座番号：1234567

預金残高：1500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**

番号が違います！

口座名義：のび太

口座番号：1234567

預金残高：1500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＞**1234**

いくら引き出しますか？＞**2000**

残高不足です！

口座名義：のび太

口座番号：1234567

預金残高：1500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＞**1234**

いくら引き出しますか？＞**300**

口座名義：のび太

口座番号：1234567

預金残高：1200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！

# **J2Kad01X「スタック！」※スタックがわからないときは調べること**

int型データを10個まで格納できるスタックを実装し、データの格納（push）と取り出し（pop）を行う処理を作成せよ。スタックのしくみがわからないときは各自で調べ、必要なフィールドも各自で考えること。また、データ数（10個まで）がオーバーしたときや、データがないのに取り出しを行ったときに例外が発生するが、今回は対処しなくてもOK。

**メソッド（スタック操作）**

|  |  |
| --- | --- |
| 書式 | 仕様 |
| public static void push(int data) | スタックにデータ（data）を格納する。 |
| public static 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. スタック操作（1：push、2：pop、-1：終了）を選択する。
2. -1（マイナスの値）のとき終了。
3. 1のときスタックにデータを格納する（値は乱数で0～99）。2のときスタックからデータを1つ取り出す。
4. スタックのデータを表示して①へ戻る。

**課題完成時の画面 データ数がオーバーしたとき（例外発生、今回は無視）**

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：89

どうしますか？（1：push、2：pop、-1：終了）＞**1**

stack：89 54

どうしますか？（1：push、2：pop、-1：終了）＞**1**

stack：89 54 6

どうしますか？（1：push、2：pop、-1：終了）＞**2**

6を取り出しました！

stack：89 54

どうしますか？（1：push、2：pop、-1：終了）＞**2**

54を取り出しました！

stack：89

どうしますか？（1：push、2：pop、-1：終了）＞**2**

89を取り出しました！

stack：

どうしますか？（1：push、2：pop、-1：終了）＞**-1**

どうしますか？（1：push、2：pop、-1：終了）＞**1**

stack：99 33 76 41 24 71 32 30 81 54

どうしますか？（1：push、2：pop、-1：終了）＞**1**

Exception in thread "main"

java.lang.ArrayIndexOutOfBoundsException: Index 10 out of bounds for length 10

at J2Kad01X.push(J2Kad01X.java:40)

at J2Kad01X.pushData(J2Kad01X.java:32)

at J2Kad01X.main(J2Kad01X.java:16)

**データがないのに取り出したとき（例外発生、今回は無視）**

スタック操作をします！

どうしますか？（1：push、2：pop、-1：終了）＞**2**

Exception in thread "main"

java.lang.ArrayIndexOutOfBoundsException: Index -1 out of bounds for length 10

at J2Kad01X.pop(J2Kad01X.java:43)

at J2Kad01X.popData(J2Kad01X.java:36)

at J2Kad01X.main(J2Kad01X.java:17)