※ 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」）

```
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);
    }
}
```

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
```

「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」）

```
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);
    }
}
```

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：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

```
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
```

（続き）

```
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

```
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
```

データ数がオーバーしたとき（例外発生、今回は無視）

```
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)
```