● J2Kad03D1「コンストラクタ」（入門編 P.165「コンストラクタとは」）

モンスターを表す Monster クラスが準備されている。**リスト 1** を入力して実行したところ、最初の showData メソッドの呼び出しで、名前が「null」、HP が「0」と表示される。名前と HP にデフォルト値（「まだないよ～」と「1」）が設定されるように Monster クラスにコンストラクタを追加せよ。

追加するメソッド（Monster クラス）

書式	仕様
public Monster()	コンストラクタ。setData メソッドを使って name に「まだないよ～」、hp に 1 を設定する。

リスト 1：J2Kad03D1 クラス

```
public class J2Kad03D1 {
    public static void main(String[] args) {
        Monster m1 = new Monster();
        m1.showData();
        m1.setData("ピカチュウ", 20);
        m1.showData();
    }
}
```

リスト 1 を入力して実行したときの画面

```
ぼくの名前は null、HP は 0 だよ！
ぼくの名前はピカチュウ、HP は 20 だよ！
```

課題完成時の画面

```
ぼくの名前はまだないよ～、HP は 1 だよ！
ぼくの名前はピカチュウ、HP は 20 だよ！
```

● J2Kad03D2「引数付きコンストラクタ」（入門編 P.168「コンストラクタのオーバーロード」）

ヤドン（HP：30）を生成してデータ表示する処理を作成せよ。なおヤドンのデータ（名前と HP）の設定は、引数付きコンストラクタで行うこと。

追加するメソッド（Monster クラス）

書式	仕様
public Monster(String n, int h)	引数付きコンストラクタ。setData メソッドを使って name に n、hp に h を設定する。

課題完成時の画面

```
ぼくの名前はヤドン、HP は 30 だよ！
```

● J2Kad03C「カプセル化」（入門編 P.231「アクセス修飾子」、P.166「コンストラクタの例」）

リスト1 を入力して実行すると、ピカチュウの名前と HP が「ゲレゲレ」と「-1」になる。これではピカチュウがかわいそうだ！勝手にピカチュウの名前や HP を変更できないようにせよ。

追加するメソッド（Monster クラス）

書式	仕様
public void setName(String name)	name のセッター。引数 name が「ゲレゲレ」のときは「ゲレゲレなんていやだ～」と表示して、名前を設定しない。
public void setHp(int hp)	hp のセッター。引数 hp が 0 以下のときは「せめて HP、1 はちょうだい！」と表示して、1 を設定する。

リスト1：J2Kad03C クラス

```
public class J2Kad03C {
    public static void main(String[] args) {
        Monster m1 = new Monster("ピカチュウ", 20);
        m1.showData();
        m1.name = "ゲレゲレ";
        m1.hp = -1;
        m1.showData();
    }
}
```

リスト1まで入力したときの画面

```
ぼくの名前はピカチュウ、HP は 20 だよ！
ぼくの名前はゲレゲレ、HP は-1 だよ！
```

課題完成時の画面

```
ぼくの名前はピカチュウ、HP は 20 だよ！
ゲレゲレなんて名前はいやだ～
せめて HP、1 はちょうだい！
ぼくの名前はピカチュウ、HP は 1 だよ！
```

● J2Kad03B「アクセサ」（入門編 P.72「3 項演算子」）

Profile クラスを作成し、ドラえもんとドラミちゃんのプロフィールを表示する処理を作成せよ。

プロフィール

	身長	体重
ドラえもん	129.3	129.3
ドラミちゃん	100.0	91.0

仕様①

J2Kad03B クラスの main メソッドでドラえもんとドラミちゃんの Profile を引数付きコンストラクタで生成し、それぞれのフィールドを直接参照して表示する処理を作成せよ。

Profile クラス（新規作成）の仕様①

	書式	説明
フィールド	public String name	名前
	public double height	身長
	public double weight	体重
メソッド	public Profile(String name, double height, double weight)	コンストラクタ。 引数を対応するフィールドに設定する。

仕様②

Profile クラスのフィールドをすべて private にし、ゲッターを使って取得するように変更せよ。

Profile クラスの仕様②

	書式	説明
メソッド	public String getName()	name のゲッター。name を返す。
	public double getHeight()	height のゲッター。height が 120 より小さいときは 170 を返す（身長をごまかす）※3 項演算子を使うこと
	public double getWeight()	weight のゲッター。weight が 100 より大きいときは 50 を返す（体重をごまかす）※3 項演算子を使うこと

仕様①まで完成したときの画面

```
ドラえもんのプロフィールです！
身長：129.3
体重：129.3
ドラミちゃんのプロフィールです！
身長：100.0
体重：91.0
```

J2Kad03B クラスから直接、身長や体重を参照するので設定した値がそのまま表示される。

仕様②まで完成したときの画面

```
ドラえもんのプロフィールです！
身長：129.3
体重：50.0
ドラミちゃんのプロフィールです！
身長：170.0
体重：91.0
```

Profile クラスからゲッターを使って取得した値を表示する。
ドラえもんは体重を、ドラミちゃんは身長をごまかしている。

● J2Kad03A「スタック！③」

J2Kad03A クラスと Stack クラスに J2Kad02X の完成版相当のプログラムが準備されている。ただし Stack クラスのフィールドは他のクラスから直接アクセスできる状態になっている。Stack クラスのフィールドを private に変更し、プログラムが動作するように修正せよ。

Stack クラスに追加するメソッド

書式	説明
public Stack(int size)	要素数 size の配列 stack を生成し、スタックポインタ（sp）を 0 にする。
public boolean isFull()	これ以上データを格納できないとき true、格納できるとき false を返す。
public boolean isEmpty()	データがないとき true、データがあるとき false 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 int get(int i)	スタックの i 番目のデータを返す。

課題完成時の画面（J2Kad02X と同じ）

```
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
```

（続き）

```
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
```

● J2Kad03S「そうだ！ECC 銀行へ行こう！！③」

J2Kad03S クラスと Account クラスに J2Kad02S の完成版相当のプログラムが準備されている。ただし Account クラスのフィールド（口座名義・口座番号・預金残高・暗証番号）は他のクラス（ここでは J2Kad03S クラス）から直接アクセスできる状態になっている（暗証番号もまる見えだ！）。Account クラスのフィールドを private に変更し、プログラムが動作するように修正せよ。

追加するメソッド（Account クラス）

書式	説明
public Account(String name, int accountNumber, int money, int secretNumber)	コンストラクタ。引数を対応するフィールドに設定する。
public String getName()	口座名義のゲッター。
public int getAccountNumber()	口座番号のゲッター。
public int getMoney()	預金残高のゲッター。
public addMoney(int money)	預金残高に引数 money を加算する。
public boolean checkSecretNumber(int secretNumber)	暗証番号と引数 secretNumber が一致していたら true を返す。 一致していなかったら false を返す。 ※暗証番号についてはゲッターを作らない。
public void subMoney(int money)	預金残高から引数 money を減らす。

課題完成時の画面（J2Kad02S と同じ）

```
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
ECC 銀行梅田本店へようこそ！
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
```

● J2Kad03X「そうだ！ECC 銀行へ行こう！！④」※JKad03S をコピーして作成

信頼と実績の ECC 銀行に難波支店ができた！これで梅田本店と難波支店の 2 店舗体制になった。のび太とスネ夫が銀行へときにどちらへ行くのか選択できる処理を追加せよ。ECC 銀行に関しては ECCBank クラスを新規作成すること。

ECCBank クラスの仕様（新規ファイル「ECCBank.java」）

	書式	説明
フィールド	private String name	店名（初期値：設定しない）
メソッド	引数付きコンストラクタ	店名（梅田本店、難波支店）を設定する
	public void gotoECCBank(Account account)	J2Kad03S の gotoBank メソッドに該当する処理
	⋮	（その他必要なメソッドあれば追加する）

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）＞1
どの支店へ行きますか？（1：梅田本店、2：難波支店）＞2
ECC 銀行難波支店へようこそ！
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？＞100

口座名義：のび太
口座番号：1234567
預金残高：100 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）＞2
どの支店へ行きますか？（1：梅田本店、2：難波支店）＞1
ECC 銀行梅田本店へようこそ！
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？＞1000000

口座名義：スネ夫
口座番号：8901234
預金残高：9000000 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
```

店名を表示すること