# **J2Kad04D「クラス変数」（入門編P.139「インスタンス変数」、P.176「クラス変数」）**

以下の仕様で処理を作成せよ。

**仕様①** Studentクラスを作成し、のび太・しずか・ジャイアンの情報を表示する処理を作成せよ。

**仕様②** コース名（course）にstaticを付けて動作確認せよ。

**Studentクラス（新規作成）の仕様**

|  |  |
| --- | --- |
| 書式 | 説明 |
| private String name | 名前 |
| private String course | コース名（IE、SK、SEなど）※仕様②でstaticを付ける |
| public Student(String name, String course) | コンストラクタ。引数の値を対応するフィールドに設定する。 |
| public void showData() | 名前とコースを表示する。 |

**のび太・しずか・ジャイアンの初期値**

|  |  |
| --- | --- |
| インスタンス名 | 初期値（名前、コース） |
| nobita | のび太、IE |
| sizuka | しずか、SK |
| suneo | スネ夫、SE |
| jaian | ジャイアン、オレさまのコース |

**mainメソッドの仕様**

1. Studentクラスのインスタンス（nobita、sizuka、suneo、jaian）を生成する。
2. 各インスタンスのデータを表示する。

**仕様①まで完成時の画面 仕様②まで完成時の画面**

のび太がやってきた！

しずかがやってきた！

スネ夫がやってきた！

ジャイアンがやってきた！

それぞれのデータを表示します！

名前：のび太 コース：オレさまのコース

名前：しずか コース：オレさまのコース

名前：スネ夫 コース：オレさまのコース

名前：ジャイアン コース：オレさまのコース

のび太がやってきた！

しずかがやってきた！

スネ夫がやってきた！

ジャイアンがやってきた！

それぞれのデータを表示します！

名前：のび太 コース：IE

名前：しずか コース：SK

名前：スネ夫 コース：SE

名前：ジャイアン コース：オレさまのコース

仕様②まで作成すると全員が

「オレさまのコース」になる。

**課題完成時のStudentクラス**

|  |
| --- |
| **Student** |
| - name : String  - course : String |
| + Student(name : String, course : String)  + showData() : void |

# **J2Kad04C「クラスメソッド」（P.171「インスタンスメソッドとは」、P.180「クラスメソッド」）**

羊を表すSheepクラスが準備されている。コンストラクタで名前（name）を設定し、introメソッドで自己紹介する。

**仕様①** 羊を3匹生成し、自己紹介する処理を作成せよ。

**仕様②** Sheepクラスに以下のメンバを追加し、羊の生成前と生成後で羊の数を表示する処理を追加せよ。

**Sheepクラスに追加するメンバ**

|  |  |  |
| --- | --- | --- |
|  | 書式 | 説明 |
| フィールド | private static int n = 0; | 生成した羊の数、初期値は0 |
| メソッド | public static void countSheep() | 生成した羊の数を表示する。「羊は全部でxx匹です！」 |

**リスト1：J2Kad04Cクラス 課題完成時の画面（羊の数の表示は仕様②）**

羊は全部で0匹です！

ホイヘンスがやってきた！

ニュートンがやってきた！

ジュールがやってきた！

羊は全部で3匹です！

「わたしはホイヘンス、よろしくメェ～！」

「わたしはニュートン、よろしくメェ～！」

「わたしはジュール、よろしくメェ～！」

public class J2Kad04C {

public static void main(String[] args) {

**羊の数を表示する**

Sheep s1 = new Sheep();

Sheep s2 = new Sheep();

Sheep s3 = new Sheep();

**羊の数を表示する**

s1.intro();

s2.intro();

s3.intro();

}

}

**課題完成時のSheepクラス**

|  |
| --- |
| **Sheep** |
| - name : String  - n : int |
| + Sheep()  + intro() : void  + countSheep() : void |

staticメンバ（下線）は仕様②で追加する。

# **J2Kad04B「インスタンスの配列」（入門編P.145「インスタンスの配列」）**

J2Kad04Cと同じ処理をSheepクラスの配列で作成せよ。

**課題完成時の画面**

（J2Kad04Cと同じ）

# **J2Kad04A「学籍番号を割り当てろ！」※J2Kad04Dをコピーして作成**

課題完成時の画面と同じようになるように、Studentクラスを修正せよ。mainメソッドはJ2Kad04Dと同じでOK。

1. ジャイアンが勝手に「オレさまのコース」に変更しないようにする。
2. 学籍番号を2230001から連番で割り当てるようにする（showDataメソッドに学籍番号の表示も追加すること）。

**課題完成時の画面**

のび太がやってきた！

しずかがやってきた！

スネ夫がやってきた！

ジャイアンがやってきた！

それぞれのデータを表示します！

名前：のび太 コース：IE 学籍番号：2230001

名前：しずか コース：SK 学籍番号：2230002

名前：スネ夫 コース：SE 学籍番号：2230003

名前：ジャイアン コース：オレさまのコース 学籍番号：2230004

**課題完成時のStudentクラス**

|  |
| --- |
| **Student** |
| - name : String  - course : String  - myNumber : int  - studentNumber : int |
| + Student(name : String, course : String)  + showData() : void |

# **J2Kad04S「カードを並べよう！」**

0～9までの番号が書かれたカードを表すCardクラス（最大10枚まで）を作成し、カードを10枚引いて表示する処理を作成せよ。

**Cardクラス（新規作成）の仕様　※必要なフィールドは各自で考えること**

|  |
| --- |
| **Card** |
| … |
| + Card()  + getNumber() : int |

|  |  |
| --- | --- |
| 書式 | 説明 |
| コンストラクタ | 「カードを引いた！」と表示し、0～9までの番号を割り当てる。  ただし他のカードと番号が重ならないようにすること。 |
| int getNumber() | 自分の番号を返す。 |

**課題完成時の画面（10枚すべて異なる番号になる）**

0から9までのカードをランダムに並べます！

カードを引いた！

カードを引いた！

カードを引いた！

カードを引いた！

カードを引いた！

カードを引いた！

カードを引いた！

カードを引いた！

カードを引いた！

カードを引いた！

順番にカードを表示します！

カードの番号は4です！

カードの番号は1です！

カードの番号は8です！

カードの番号は3です！

カードの番号は2です！

カードの番号は9です！

カードの番号は5です！

カードの番号は0です！

カードの番号は6です！

カードの番号は7です！

# **J2Kad04X1「レジ待ち行列①」**

ECCが激安スーパーを開業した！その名も「激安スーパーECC」。J2Kad04X1に客の呼び込みとレジ打ちを行う処理が準備されている。ところがレジ待ち行列をスタック（Stack）形式にしたため、後から並んだ人が先に処理されてしまう！ 先に並んだ人から先に処理されるキュー（Queue）クラスを作成し、StackをQueueに置き換えよ。なお、Queueのインターフェイス（publicなフィールド・メソッドの書式）はStackクラスと同じでOK。

**課題作成前の画面（Stackクラスの場合、行列の最後の人から処理される）**

いらっしゃい！激安スーパーECCです！！

何をしますか？（0：客を呼び込む、1：レジを打つ、-1：店をたたむ）＞**0**

ギャラドスがやってきた！

現在のレジ待ち行列です！

0：ギャラドス

何をしますか？（0：客を呼び込む、1：レジを打つ、-1：店をたたむ）＞**0**

ミュウがやってきた！

現在のレジ待ち行列です！

0：ギャラドス

1：ミュウ

…

　（中略）

…

何をしますか？（0：客を呼び込む、1：レジを打つ、-1：店をたたむ）＞**0**

ムックルがやってきた！

もう店に入れない！残念！！

現在のレジ待ち行列です！

0：ギャラドス

1：ミュウ

2：カビゴン

3：スピアー

4：ピジョン

何をしますか？（0：客を呼び込む、1：レジを打つ、-1：店をたたむ）＞**1**

ピジョンは帰っていった！！

現在のレジ待ち行列です！

0：ギャラドス

1：ミュウ

2：カビゴン

3：スピアー

何をしますか？（0：客を呼び込む、1：レジを打つ、-1：店をたたむ）＞**-1**

|  |
| --- |
| **Queue** |
| - container : Monster[]  - sp : int |
| + Queue(size : int)  + push(data : Monster) : void  + pop() : Monster  + get(i : int) : Monster  + size() : int  + empty() : boolean  + full() : boolean |

Queueのインターフェイス（フィールド・メソッドの書式）はStackと同じ

Stackの場合、最後尾のピジョンが処理される。Queueにすると先頭のギャラドスから処理されるようになる。

# **参考：キュー（Queueクラス、データシフト方式）**

キューは、最初に格納したデータが最初に取り出されるというデータ構造です（First In, First Out、略してFIFO）。非同期処理をするプログラム間でデータを受け渡す場合などで使われます。

データ格納（プッシュ）

データ読み出し（ポップ）

3

2

6

1

47

0

23

①

0

3

2

1

sp

①

②

sp

6

47

①

23

データ読み出し時は

①0番からデータを取り出し、

②spを-1すると同時に残っているデータをずらす。

データ格納時は

①spの指す場所にデータを格納し、

②spを+1する。

# **J2Kad04X2「レジ待ち行列②（リングバッファ）」**

ECCが激安スーパーの2号店を開業した！ところがレジ待ち行列をまたもやスタック形式にしてしまった！！ リングバッファ形式のキュー（Queue2）クラスを作成し、Stackと置き換えよ。

|  |
| --- |
| **Queue2** |
| - container : Monster[]  - start : int  - used : int |
| + Queue2(size : int)  + push(data : Monster) : void  + pop() : Monster  + get(i : int) : Monster  + size() : int  + empty() : boolean  + full() : boolean |

# **参考：リングバッファ（Queue2クラス）**

行列の先頭位置を表すstartと並んでいる人数を表すusedを使って作成します。データの書き込みはstart＋usedの場所に行います。書き込みを行うとusedを1増やします。読み出しはstartの場所を読み出し、startを1進めusedを1減らします。読み出す場所や書き込む場所が配列の最後尾を越えたら先頭へ戻ります（最後尾と先頭がリング状につながっているイメージ）。

used（行列の人数）

|  |  |
| --- | --- |
| 4 | データ |
| 3 | データ |
| 2 | データ |
| 1 | データ |
| 0 | データ |

メソッドの書式はStackと同じ

start（行列の先頭）

最後尾まで行ったら先頭へ戻る