● J2Kad14D「例外処理①（try～catch）」

（実践編 P.37「プログラム実行時のトラブル」P.39「例外の発生する状況」P.40「投げられた例外をキャッチする」）

年齢を入力して表示する処理が準備されている。年齢に整数を入力すると正常に動作するが文字列を入力すると例外が発生する。

① 整数と文字列を入力して、文字列のとき例外が発生することを確認せよ（**課題作成前の画面①、②**）。
② 例外が発生したとき、発生した例外の名前と「例外が発生しました！」（←変更可）という文字列を表示するようにせよ。

課題作成前の画面①（整数を入力したとき）

```
あなたの年齢を教えてください>20
あなたは 20 歳なんですね！
```

課題作成前の画面②（文字列を入力したとき）

```
あなたの年齢を教えてください>ecc
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:943)
    at java.base/java.util.Scanner.next(Scanner.java:1598)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2263)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2217)
    at J2Kad14D.main(J2Kad14D.java:14)
```

← InputMismatchException が発生

課題完成時の画面（文字列を入力したとき）

```
あなたの年齢を教えてください>ecc
java.util.InputMismatchException
何で整数ちゃうねん！年齢は整数に決まっとるやろ！！
```

● J2Kad14C「例外処理②（try～catch～finally）」

（実践編 P.44「catch ブロックの検索」、P.47「例外オブジェクトの種類による場合分け」、P.42「finally の処理」）

2 つの整数を入力すると割り算を行って結果を表示する処理が準備されている。発生する例外を確認し、発生した例外に対応した文字列を表示せよ。また finally ブロックも追加せよ。

・InputMismatchException が発生したとき　…例外の名前と「int 型でない値が入力されました！」(←変更可) を表示する
・ArithmeticException が発生したとき　…例外の名前と「0 除算が発生しました！」(←変更可) を表示する
・finally ブロックの処理　…「finally ブロックの処理です！」と表示する

課題作成前の画面①（整数 1 に文字列を入力したとき）

```
整数1を入力してください>ecc
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:943)
    at java.base/java.util.Scanner.next(Scanner.java:1598)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2263)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2217)
    at J2Kad15D.main(J2Kad15D.java:9)
```

← InputMismatchException が発生

課題作成前の画面②（整数 2 に 0 を入力したとき）

```
整数1を入力してください>10
整数2を入力してください>0
10÷0を計算します！
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at J2Kad15D.main(J2Kad15D.java:14)
```

← ArithmeticException が発生

課題完成時の画面①（整数 1 に文字列を入力したとき）

```
整数1を入力してください>ecc
java.util.InputMismatchException
だから整数を入力せえってゆうてるやろ！
finallyブロックの処理です！
```

課題完成時の画面②（整数 2 に 0 を入力したとき）

```
整数1を入力してください>10
整数2を入力してください>0
10÷0を計算します！
java.lang.ArithmeticException: / by zero
ゼロで割ったらどないなる思ってんねん！
finallyブロックの処理です！
```

課題完成時の画面③（正常終了のとき）

```
整数1を入力してください>10
整数2を入力してください>5
10÷5を計算します！
計算結果は2です！
finallyブロックの処理です！
```

● J2Kad14B「ECC 長屋、恐怖の 3 号室！」

ECC 長屋（部屋数 3）に入居している羊たちに自己紹介させる処理が準備されている。

① 存在しない部屋番号を入力して例外が発生したときに「そんな部屋番号はありません！」（←変更可）と表示せよ。
② 部屋番号に文字列など整数でない値が入力されたとき「int 型でない値が入力されました！」（←変更可）と表示せよ。

課題作成前の画面①（2 より大きい値を指定したとき）

```
ポアンカレがやってきた！
ピタゴラスがやってきた！
ホイヘンスがやってきた！
何号室を見ますか？（-1：興味なし）＞3
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
    at J2Kad15B.main(J2Kad15B.java:12)
```

課題作成前の画面②（文字列を入力したとき）

```
マクスウェルがやってきた！
アルキメデスがやってきた！
ニュートンがやってきた！
何号室を見ますか？（-1：興味なし）＞ecc
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:943)
    at java.base/java.util.Scanner.next(Scanner.java:1598)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2263)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2217)
    at J2Kad14B.main(J2Kad14B.java:15)
```

課題完成時の画面

```
ド・モルガンがやってきた！
フィボナッチがやってきた！
ケプラーがやってきた！
何号室を見ますか？（-1：興味なし）＞3
配列の部屋数は 3 やけど、部屋番号は 0 から始まるから、0 号室・1 号室・2 号室までしかないねん。
3 号室はないって何回ゆうたらわかんねん？
何号室を見ますか？（-1：興味なし）＞ecc
部屋番号は整数やって何回ゆうたらわかんねん！
何号室を見ますか？（-1：興味なし）＞
```

● J2Kad14A「例外処理まとめ」（実践編 P.48、List②-6 のアレンジ）

割り算をして結果を配列に格納する処理を作成せよ。また**課題完成時の画面**を参考に例外発生時の処理も作成せよ。

① int 型配列（要素数 5）を準備する。
② 割られる数（A÷B の A に該当する数）をキーボードから入力する。
③ 割る数（A÷B の B に該当する数）は 0 から 9 までの値を乱数で決める。
④ 計算結果を配列に格納する。このとき格納先のインデックスは 0 から 9 まで（これも乱数で決定）とする。
⑤ ②～④を無限ループで繰り返す。なお 0 除算の例外が発生したときにループから抜けるものとする。

課題完成時の画面

```
割られる数を入力してください＞10
9 で割ります！
計算結果は 1 です！
配列の 4 番目に格納します！
処理が正常に行われました！
finally ブロックの処理です！

割られる数を入力してください＞ecc
int 型でない値が入力されました！
finally ブロックの処理です！

割られる数を入力してください＞10
7 で割ります！
計算結果は 1 です！
配列の 8 番目に格納します！
配列に格納できません！
finally ブロックの処理です！

割られる数を入力してください＞10
0 で割ります！
0 除算が発生しました！          ←―― 0 除算が発生したらループを抜けて終了する
finally ブロックの処理です！
終了します！
```

● J2Kad14S「じゅげむじゅげむ・・・」

「じゅげむじゅげむ・・・」(←むちゃくちゃ長い名前) を表示したい。ただしむちゃくちゃ長い名前なのでいくつかのフレーズに分けて Phrase クラスで管理している。**リスト１** で名前が表示できるように Phrase クラスを完成させよ。必要であればメンバを追加してもよい。

リスト１:「じゅげむじゅげむ・・・」(J2Kad14S クラス)

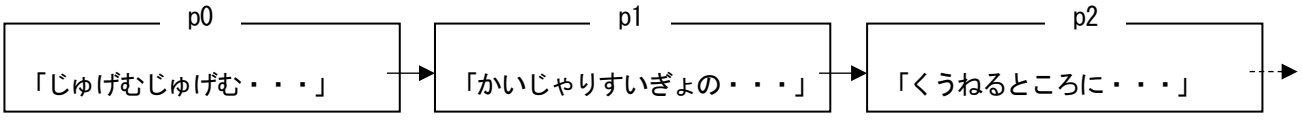
```
public class J2Kad14S {
    public static void main(String[] args) {
        // インスタンスの生成
        Phrase p0 = new Phrase("じゅげむ じゅげむ ごこうのすりきれ");
        Phrase p1 = new Phrase("かいじゃりすいぎょの すいぎょうまつ うんらいまつ ふうらいまつ");
        Phrase p2 = new Phrase("くうねるところに すむところ やぶらこうじの ぶらこうじ");
        Phrase p3 = new Phrase("パイポパイポ パイポのシューリンガン");
        Phrase p4 = new Phrase("シューリンガンのグーリンダイ");
        Phrase p5 = new Phrase("グーリンダイのポンポコピーのポンポコナーの");
        Phrase p6 = new Phrase("ちょうきゅうめいの ちょうすけ");
        // インスタンスの連結
        p0.setNext(p1).setNext(p2).setNext(p3).setNext(p4).setNext(p5).setNext(p6);
        // 表示
        Phrase p = p0;
        while (p != null) {
            System.out.println(p);          // フレーズの表示
            p = p.getNext();                // 次のフレーズへ
        }
    }
}
```

課題完成時の画面

```
じゅげむ じゅげむ ごこうのすりきれ
かいじゃりすいぎょの すいぎょうまつ うんらいまつ ふうらいまつ
くうねるところに すむところ やぶらこうじの ぶらこうじ
パイポパイポ パイポのシューリンガン
シューリンガンのグーリンダイ
グーリンダイのポンポコピーのポンポコナーの
ちょうきゅうめいの ちょうすけ
```

Phrase
- phrase : String …
+ Phrase(phrase : String) + toString() : String …

ヒント：次のフレーズを指すしくみを考えること



● J2Kad14X「ECC 苦情処理センター（Chain of Resposibility)」※検索すること

J2Kad05X「ECC 苦情処理センター」完成版相当に応援スタッフも呼ぶことのできるプログラムを業者に作ってもらった（次ページの**リスト1**)。のび太・ジャイアン・スネ夫だけでなく、必要であれば、しずかちゃんと出木杉くんも応援に入ることができる。ただし業者の作ったプログラムは**見るのもおぞましいとんでもないシロモノ**だった！

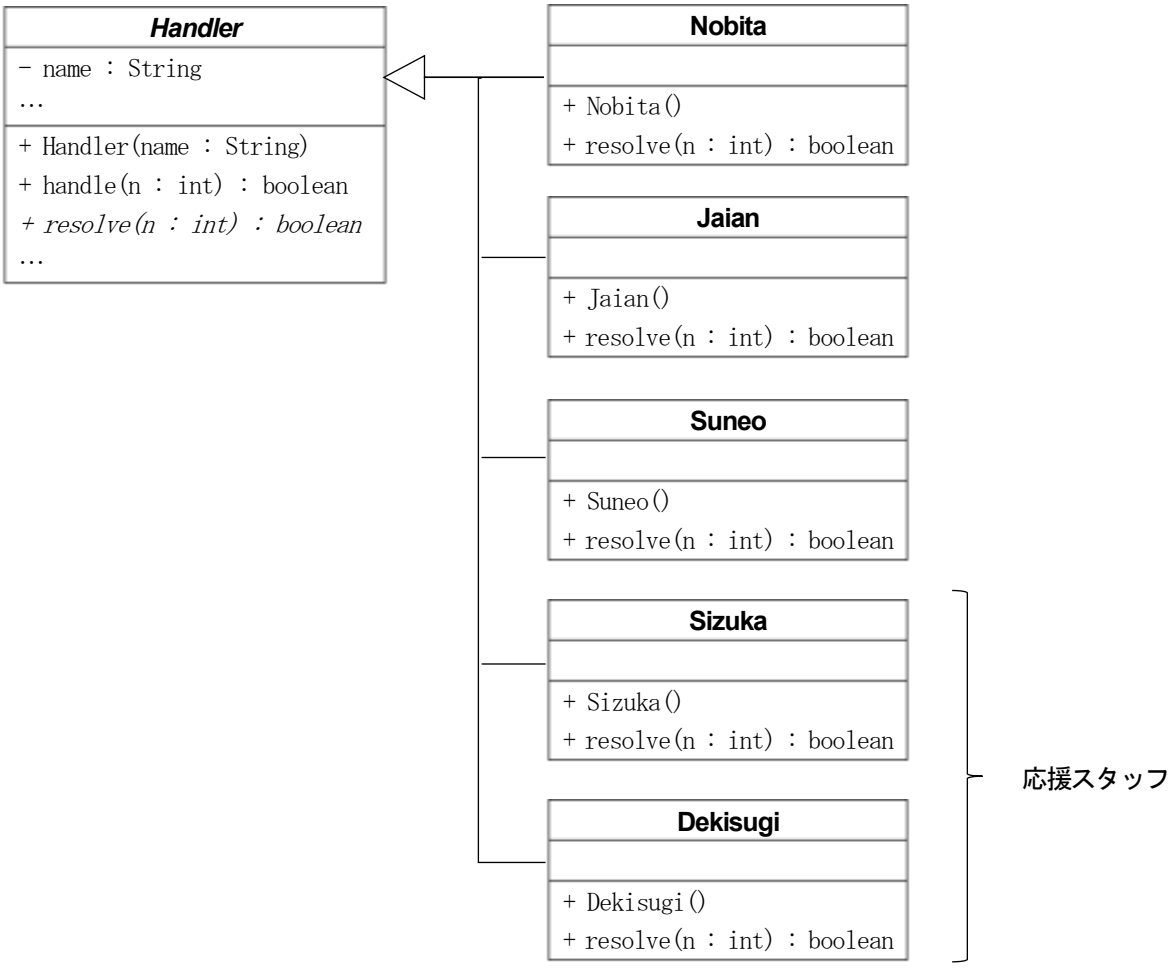
① スタッフクラスが未完成である。**リスト1**が動作するように各スタッフクラスを完成させよ。
② **リスト1**のコードを簡略化せよ。

優秀なスタッフたち

対応順	スタッフ	対応できる番号
1	のび太	20 未満のとき対応可能
2	ジャイアン	苦情番号とは関係なく、乱数（0～3）で 0 が出たら対応可能
3	スネ夫	3 の倍数のとき対応可能

さらに優秀なスタッフたち（応援スタッフ）

対応順	スタッフ	対応できる番号
4	しずか	2 の倍数のとき対応可能
5	出木杉	7 の倍数以外のとき対応可能



リスト１：見るのもおぞましいとんでもないコード（J2Kad14X クラス）

```
public class J2Kad14X {
    public static void main(String[] args) {
         ⋮
        Nobita nobita = new Nobita();
        Jaian jaian = new Jaian();
        Suneo suneo = new Suneo();

        Sizuka sizuka = null;                // まだいない
        Dekisugi dekisugi = null;            // まだいない

        boolean helper = false;             // false：応援なし、true：応援あり

        while(true) {
            System.out.println();
            System.out.print("どうしますか？ (0：苦情を受け取る、1：応援を頼む、-1：もうやだ) ＞");
            int cmd = in.nextInt();
            if (cmd < 0) break;
            // 応援を頼む
            if (cmd == 1) {
                if (!helper) {
                    sizuka = new Sizuka();
                    dekisugi = new Dekisugi();
                    helper = true;
                }
                continue;
            }
            // 苦情処理
            int n = (int)(Math.random() * 100);
            System.out.println("苦情番号：" + n + "を受け付けた！");
            if (!nobita.handle(n)) {
                if (!jaian.handle(n)) {
                    if (!suneo.handle(n)) {
                        if (helper) {
                            if (!sizuka.handle(n)) {
                                dekisugi.handle(n);
                            }
                        }
                    }
                }
            }
        }
        System.out.println("おつかれさまでした！");
    }
}
```

課題完成時の画面

```
ここはECC苦情処理センターです！
優秀なスタッフが対応します！
のび太がスタンバイした！
ジャイアンがスタンバイした！
スネ夫がスタンバイした！

どうしますか？（0：苦情を受け取る、1：応援を頼む、-1：もうやだ）>0
苦情番号：72を受け付けた！
のび太：専門外です・・・
ジャイアン：私が対応します！

どうしますか？（0：苦情を受け取る、1：応援を頼む、-1：もうやだ）>0
苦情番号：40を受け付けた！
のび太：専門外です・・・
ジャイアン：専門外です・・・
スネ夫：専門外です・・・

どうしますか？（0：苦情を受け取る、1：応援を頼む、-1：もうやだ）>1
しずかがスタンバイした！
出木杉がスタンバイした！

どうしますか？（0：苦情を受け取る、1：応援を頼む、-1：もうやだ）>0
苦情番号：85を受け付けた！
のび太：専門外です・・・
ジャイアン：専門外です・・・
スネ夫：専門外です・・・
しずか：専門外です・・・
出木杉：私が対応します！

どうしますか？（0：苦情を受け取る、1：応援を頼む、-1：もうやだ）>-1
おつかれさまでした！
```

ヒント：のび太が対応できないときはジャイアンへ、ジャイアンが対応できないときはスネ夫へ処理を回すしくみを考える。
（「Chain of Responsibility」←検索）