東京ITスクール
TOKYO IT SCHOOL

プログラミング演習
演習問題概要

版数	発行日	改定内容
1.0	2022/04/01	初版発行

東京ITスクール
TOKYO IT SCHOOL

目次

1. はじめに

この資料はプログラミング演習で使用する演習問題について以下の内容が記載されております。
問題の概要:その問題を解く際に使われる単元や技術
プログラムの動作の概要:プログラムを動作したときの概要
難易度:どれくらいで完成できるかの目安時間(解説などの時間は考慮していない)
解説のポイント:問題の解説の際に重要なポイント

2. 演習問題の実施時間

各演習問題については以下の表の時間を制限時間として演習を実施してください。
なお、どの問題を実施するかは受講生の習熟度に合わせて決めてください。各問題の難易度や概要に関しては「プログラミング演習_演習問題概要」を参照してください。

演習問題名			学習項目	制限時間
プログラミング演習_基本構文	section01_array	challnege01	配列	15 分
		challenge02	配列	30 分
		challenge03	多次元配列	35 分
	section02_conditional_branch	challnege01	if 文	30 分
		challenge02	switch 文	35 分
		challenge03	if 文	1 時間 40 分
	section03_repetition	challenge01	for 文	30 分
		challenge02	for 文のネスト	35 分
	section014_summally	challenge01	Java 上巻	2 時間 30 分
		challenge02		1 時間 50 分
		challenge03		2 時間 20 分
		challenge04		1 時間 30 分
		challenge05		1 時間 20 分
		challenge06		1 時間 20 分
		challenge07		2 時間 20 分
プログラミング演習_オブジェクト指向	challenge01		クラスとメソッド	2 時間
	challenge02		static 選手	2 時間
	challenge03		継承	2 時間
	challenge04		抽象クラス・インターフェイス	3 時間
じゃんけんプログラム			Java 総合	2 日(12 時間)
団体じゃんけん			Java 上巻	2 日(12 時間)
インディアンポーカー			Java 上巻	2 日(12 時間)

モグラたたき(配列版)	Java 総合	2 日(12 時間)
オブジェクトじゃんけん	Java 総合	2 日(12 時間)
モグラたたき(コレクション版)	Java 総合	2 日(12 時間)
浅草ジャマイカホール①	Java 上巻	2 時間
浅草ジャマイカホール②	Java 総合	2 時間

3. プログラミング演習_基本構文

- **section01_array**
  1. **challenge01**

     問題の概要

     以下の内容を使用して実装する。

     - 配列の初期化
     - 配列の要素の呼び出し
     - 配列の要素数を調べる

     プログラムの動作の概要

     プログラムを実行すると、コンソールに配列の配列の初期化で代入された値を添え字の順で表示し、最後に配列の要素数を表示する。

     難易度

     配列が理解できている受講生で 10 分程度

     解説のポイント

     - 配列の要素の添え字は 0 から始まること。
     - 配列の添え字の最大値は「配列の要素数-1」であること。
     - 配列の長さ（要素数）を求めるときは「配列変数名.length」と記述すること。

  2. **challenge02**

     問題の概要

     以下の内容を用いて実装する。

     - 配列の要素への値の代入
     - 加算代入を用いた省略表記
     - Scanner クラス

     プログラムの動作の概要

     コンソール上で 5 つの整数を入力し、配列に代入される。配列に代入された値と値の合計値を出力する。

     難易度

     配列が理解できている受講生で 15 分程度

     解説のポイント

     - Scanner クラスの使用法に関して補足説明をすること。
     - 配列の要素の値の代入を行う際には「配列変数名[添え字] = 値 ; 」と記述すること。
     - 代入演算子にて「sum += array[0]」と書くことで「sum = sum + array[0]」を短縮して記述ができること。

3. **challenge03**

問題の概要

以下の内容を用いて実装する。

- 多次元配列の生成
- 多次元配列の代入
- 多次元配列の要素数を調べる
- Scanner クラス

プログラムの動作の概要

コンソール上で 2 クラス計 6 人の点数を入力し、多次元配列に代入する。各クラスの合計点数整数で出力し、平均点数を浮動小数点で出力する。

難易度

配列を理解できている受講生で 20 分程度

解説のポイント

- 多次元配列の宣言は「型名[][] 配列変数名 = new 型名 [1 次元目の要素数][2 次元目の要素数]；」となる。
- 要素の呼び出しは「配列変数名[1 次元目の添え字][2 次元目の添え字]」となる。
- 2 次元目の配列の長さを求めるときは「配列変数名[添え字].length」と書くこと。
  「配列変数名.length」では 1 次元目の配列の長さが求められる。

- **section02_conditional_branch**

1. **challenge01**

問題の概要

以下の内容を用いて実装する。

- if~else if~else 文
- 比較演算子
- 論理演算子
- Scanner クラス

プログラムの動作の概要

コンソール上で 100 以下の数値を 3 つの整数を入力し、3 つの平均値を小数で表示する。その後、平均点に応じてメッセージを表示する。入力する整数で 100 より大きい数値が入力された場合は 3 つの数値が入力された後メッセージを表示して、プログラムを終了する。

難易度

条件分岐を理解できている受講生で 15 分程度

解説のポイント

- if 文を使用した条件分岐の書き方は「if(条件){ 条件が true だった時の処理 }」と書くこと。
- 「||」は左辺と右辺のどちらか一方が true であれば全体の評価を true となること。
- 「==」は両辺の値が等しいかどうかを判定する比較演算子であること。
- else if は複数の条件を使った分岐を実行させる際に使用する。条件判定は必ず上から実行され、最初に条件を満たした処理を実行すること。

2. **challenge02**

問題の概要

- switch 文

- Scanner クラス

プログラムの動作の概要

コンソール上で日付を整数で入力し、日付に応じた曜日を出力する。正しくない日付を入力された場合は入力内容が正しくないメッセージを表示する。

難易度

条件分岐を理解している受講生で 20 分程度

解説のポイント

- switch 文は値の完全一致のみ比較を行うことができる。
- 複数の case 文に対して処理を 1 つのみにすることも可能。今回の場合では case で指定した値のどれかに一致した場合に対応した処理を行うようになっている。
- break 文を記載しないとそこから下の処理がすべて実行されてしまい、想定外の結果になる可能性がある。
- 指定したいずれの値にも該当しないときの処理は default 文に記述する。

3. challenge03

問題の概要

以下の内容を用いて実装する。

- boolean 型の変数を用いたフラグ
- if 文
- 論理演算子
- break 文の使い方
- Scanner クラス

プログラムの動作の概要

コンソール上で月と日を整数で入力して、入力した月日までのその年で経過した日数と残りの日数を出力する。不正な月日が入力された場合はその時点でメッセージを出力し、プログラムを終了する。

難易度

条件分岐を理解している受講生で 1 時間程度

解説のポイント

- 日付の入力チェックを行う際に boolean 型の変数を用意し、その中に入力チェックの結果を格納するようにする「フラグ」について。
- 「System.exit(0)」はシステムを正常終了するための処理。
- switch 文を利用した経過日数の計算の際に break 文を書かずに実行する。switch 文は case の値に一致したところまで処理を飛ばすことが本来の役割のため、break 文を実行しなければそこから下の処理を実行することができる。

・ section03_repetition

1. challenge01

問題の概要

以下の内容を用いて実装する。

- for 文
- if~else 文
- boolean 型の変数でのフラグ

- final 修飾子
- Scanner クラス

プログラムの動作の概要

コンソール上で整数を入力し、素数かどうかを判定する。結果に応じてメッセージを出力する。

難易度

繰り返しが理解できている受講生で 15 分程度

解説のポイント

- 今回のループカウンタの初期値は 2 にすること。これは素数が 1 とその数でしか割れないため、2 から始めることで割り切れなかったら素数とすることができる。
- 繰り返しの条件はループカウンタが入力された値の半分以下までと設定する。これは処理回数をできるだけ減らすことで無駄な処理を行わないため。
- 割り切れるかどうかの判定には「%」を用いて、余りが 0 かどうかで判定する。

2. challenge02

問題の概要

以下の内容を用いて実装する。

- for 文のネスト
- if~else 文
- boolean 型の変数でのフラグ
- final 修飾子
- 文字列連結演算子
- Scanner クラス

プログラムの動作の概要

コンソール上で整数を入力し、入力された値までの素数をすべて出力する。

難易度

繰り返しが理解できている受講生で 20 分程度

解説のポイント

- for 文のネストを行うことで内側の for 文を何回繰り返すか制御することができる。
- for 文のネストのうち、内側の for 文は素数かどうかの判定を行い、外側の for 文はその判定を 2 から入力された値まで繰り返すためのもの。
- 外側の for 文が 2 回目の処理を行うとき、内側の for 文は初期化の式から実行される。
- 素数を表示するための String 型の変数はスコープの関係で for 文の外で宣言をする。

・ section04_sumally

1. challenge01

問題の概要

以下の内容を用いて実装する。

- 配列の初期化
- while 文
- 配列の生成
- for 文
- if~else if~else 文
- デクリメント

- continue 文
- break 文
- 配列の要素の呼び出し
- エスケープシーケンス
- Scanner クラス

プログラムの動作の概要

コンソール上で券売機の再現を行う。コンソール上で任意の文字列を入力し、Enter を押す。表示された内容の中から整数を 3 つまで入力して、注文をする。その時表示されていない数値を入力した場合、メッセージが出力され、再度入力を求められる。3 つ数値が入力されるか 9 が入力されたら注文された商品の合計金額を出力する。商品が 1 つも注文されずに 9 が入力されたらメッセージを出力し、最初の処理に戻る。金額を入力し、注文した商品の引換券を出力する。その後おつりがある場合はおつりの金額を出力して、最初の処理に戻る。この処理をプログラムが停止するまで繰り返す。

難易度

配列、条件分岐、繰り返しを理解している受講生で 2 時間程度

解説のポイント

- 商品名を保存する配列と価格を保存する配列をそれぞれ作成する。各商品の名前と対応する価格は同じ添え字になるように保存する。
- 今回の処理である注文の受付から引換券の発見までの処理は while 文を使用して無限ループさせる。
- 入力された注文内容は用意しておいた注文内容を入れる、int 型の配列に入れて保存する
- 注文を保存する配列の 0 番の要素の値が 0（注文がない）の時は注文受付の処理に戻すため、continue を使用して、以降の処理をスキップさせる。
- 合計金額の算出は注文内容を保存している配列の各要素の値を取り出し、その値は価格を保存している配列の要素の取り出しの添え字に使用することで注文した商品の価格を取り出すことができる。取り出した価格は用意した合計金額を保存する変数に加算する。
- 引換券を発券するときに表示する商品名は合計金額を算出するときの各商品の金額を取り出すときと同じ原理を使用して、商品名を取り出す。

2. challenge02

問題の概要

以下の内容を用いて実装する。

- 配列の要素の取り出し
- for 文
- 拡張 for 文(for 文でも代用可)
- if 文
- デクリメント
- break 文

プログラムの動作の概要

配列要素のシャッフルでは 1 から 10 の整数が入った配列の要素を Random クラスでシャッフルし、出力する。

重複なしの配列では 1 から 10 の整数を重複なしでランダムに配列に代入し、コンソール上に出力する。

難易度

配列、条件分岐、繰り返しを理解している受講生で 1 時間程度

解説のポイント

配列要素のシャフル

- シャッフルを実行するときはループカウンタで現在のインデックスを指定して、値を別の変数に代入して、避難させておく
- ランダムに指定した要素の値を現在のインデックスに代入した後にランダムで指定した要素に避難させておいた値を代入する。これで値を入れ替えることができる

重複なしの配列

- ランダムで生成した値は必ず+1 する。今回は 0 から 9 の値をランダムに生成しているが、配列の中に入れる値は 1 から 10 のため値を+1 する必要がある。
- for 文のネストを行い、内側の for 文でここまでに入れてきた値と重複がないか確認する処理を記述する。もし、重複していた場合は外側の for 文のループカウンタを-1 して、break 文で処理を中断する。ループカウンタを-1 しないと該当する要素に値が入らずに次の要素に値を入れる処理になってしまう。

3. challenge03

問題の概要

以下の内容を用いて実装する。

- 配列
- for 文のネスト
- if~else 文
- Scanner クラス

プログラムの動作の概要

コンソール上で要素数を整数で入力する。入力された値の数だけ横棒グラフを作成してコンソール上に出力する。棒グラフは*で表現し、*の個数は 1 から 10 のランダムで決まる。

難易度

配列、条件分岐、繰り返しを理解している受講生で 1 時間 30 分程度

解説のポイント

横棒グラフの表示

- for 文のネストを使用し、内側の for 文は要素の値の数だけ*を表示する処理を書く。外側の for 文は内側の for 文の処理を配列の要素数だけ繰り返す。
- 体裁崩れを防ぐために行番号は下 1 桁のみ表示をするようにする

縦棒グラフの表示

- イメージとしては 10 マス×10 マスの表をイメージして、1 番上の行は要素の値が 10 の要素は*を表示して、それ以外は空白にする。2 番目の行は要素の値が 9 の要素は*を表示して、それ以外は空白を表示する。これを 1 番下の行まで繰り返す。「System.out.print();」は横方向にのみ表示ができるが縦方向に表示する機能はないため、このように処理する必要がある。

4. challenge04

問題の概要

以下の内容を用いて実装する。

- 配列

- if~else 文
- for 文
- 拡張 for 文(for 文で代用可)
- Scanner クラス

プログラムの動作の概要

コンソール上に 1 から 100 までの数字をランダムで重複なしに出力する。ただし、ランダムで 1 つの数値を出力しない。出力しなかった数値を特定する処理を出力する。最後に整数を入力し、入力された値に応じてメッセージを出力する。

難易度

配列、条件分岐、繰り返しを理解している受講生で 1 時間程度

解説のポイント

- for 文を使用して、1 から 100 までの値をすべて配列の要素に代入する。この時 Random クラスを使用して、決めた値を代入しないように if 文を使用して条件分岐させること。
- シャッフルを実行するときはループカウンタで現在のインデックスを指定して、値を別の変数に代入して、避難させておく。
- ランダムに指定した要素の値を現在のインデックスに代入した後にランダムで指定した要素に避難させておいた値を代入する。これで値を入れ替えることができる。
- 読まれなかった数字を特定させるための処理は、 1 から 100 までの合計と読まれた数字が入っている配列の各要素の値の合計の差から求めることができる。また、合計を求める際に代入演算子を利用することで短く記述することができる。

5. challenge05

概要

以下の内容を用いて実装する。

- 配列
- for 文のネスト
- if 文
- Scanner クラス

プログラムの動作の概要

10 個のランダムな数値を配列に代入し、コンソール上に出力する。その後配列の中の値をバブルソートで値を昇順に並び替えてコンソールに出力する。

難易度

配列、条件分岐、繰り返しを理解している受講生で 30 分程度

解説のポイント

- バブルソートは for 文のネストを使用して実装する。外側の for 文は現在のインデックスを対象とするための for 文。内側の for 文は対象となっているインデックスより小さいインデックスと値を比較するための for 文である。
- 現在のインデックスの値と数字を入れ替えるときは、現在のインデックスの値を別の変数に代入し、避難させる。その後、現在のインデックスに入れ替え先の値を代入する。入れ替えたインデックスの値に避難させた値を代入して、値の入れ替えを完了させる。

6. challente06

問題の概要

以下の内容を用いて実装する。

- 配列
- for 文のネスト
- if 文

プログラムの動作の概要

10 個のランダムな数値を配列に代入し、コンソールに出力する。配列の要素の値を選択ソートで降順に並び変える。並び替えた後、配列の要素をすべて出力する。

難易度

配列、条件分岐、繰り返しを理解している受講生で 30 分程度

解説のポイント

- 選択ソートは for 文のネストを使用して実装する。外側の for 文は対象となるインデックスを順番に指定するために使用する。内側の for 文は対象となるインデックスよりも添え字が大きいインデックスの中から値が大きい要素を特定するための for 文となる。
- 内側の for 文の初期化は外側の for 文のインデックス+1 とする。こうすることで対象となっているインデックスよりも添え字が大きい要素だけをソートの対象とすることができる。

7. challenge07

問題の概要

以下の内容を用いて実装する。

- 多次元配列
- for 文のネスト
- Scanner クラス
- System.out.printf を用いた出力の書式指定(実装は必須ではない)

プログラムの動作の概要

コンソール上でクラス数を整数で入力し、入力された値の数だけ、多次元配列の 1 次元目の要素を生成する。その後クラスの人数を整数で入力し、入力された数だけ点数の入力を受け付ける。これをクラス数だけ繰り返す。最後にクラス毎の合計点数と平均点数、全体の合計点数と平均点数をコンソールに出力する。

難易度

配列、繰り返しを理解している受講生で 1 時間 30 分程度

解説のポイント

- 多次元配列を生成する際に、1 次元目の要素数のみを決めて、2 次元目の要素数を決めずに配列を生成することができる。この時、1 次元目の要素数の数だけ配列を生成して、その参照値を 1 次元目の要素に代入する必要がある。
- 多次元配列のすべての要素に対して同じ操作を行う場合には for 文のネストを使用する。外側の for 文では 1 次元目の要素、内側の for 文では 2 次元目の要素を扱う。
- 2 次元目の要素数を調べる時には「配列変数名[1 次元目の添え字].length」で調べることができる。
- 「System.out.printf():」は標準出力の書式を指定するメソッド。今回の模範解答で使用している「%6d」は整数を 10 進数で表示し、桁数を 6 桁と指定している。「%6.1f」は小数を小数点数で表示し、桁数を整数部分は 6 桁、小数部分は 1 桁と指定している。

4. プログラミング演習_オブジェクト指向

※注意

プログラミング演習_オブジェクト指向には事前に用意されているファイルがあるため、作成しなくてはいけないファイルや編集するファイルを確認してください。

- **challenge01**

  問題の概要

  以下の内容を用いて実装する。

  - クラス
  - オブジェクト生成
  - メソッド
  - アクセス修飾子
  - アクセサ(getter、setter)
  - コンストラクタ
  - toString メソッド
  - オーバーライド
  - Scanner クラス

  プログラムの動作の概要

  コンソール上で名前、性別、年齢、誕生日の年、月、日を入力し、入力値を使用してオブジェクト生成する。入力した情報をコンソールに出力する。

  難易度

  クラス、メソッド、コンストラクタ、オーバーライドを理解している受講生で 1 時間程度

  解説のポイント

  - フィールドの型は基本型以外にも作成したクラスを型とすることもできる。
  - コンストラクタは複数定義することができる。ただし引数の個数、型、順番が異なることが条件。
  - コンストラクタの呼び出しはオブジェクト生成をしたときの new の後の「クラス名();」で呼び出される。()の中には引数を入れる。
  - コンストラクタの中で他のコンストラクタを呼び出す際には「this(引数)」と記述する。
  - コンストラクタ内の「this.フィールド名」はフィールド名と引数名が同じ場合、引数と区別するためにフィールド名に「this」を付ける。
  - クラスの toString()メソッドをオーバーライドすることで、クラス型の変数を直接出力したときにハッシュ値の代わりに指定した文字列を出力することができる。

- **challenge02**

  問題の概要

  以下の内容を用いて実装する。

  - static 選手
  - do~while 文
  - Scanner クラス

  プログラムの動作の概要

  コンソール上で名前、性別、年齢、誕生日の年、月、日、部署番号を入力する。部署番号について、正しく選択されていない場合は再度入力を受け付ける。入力値を使用してオブジェクト生成を 2 回行う。それぞれのオブジェクトの toString メソッドを利用して、入力された内容をコンソールに出力する。

難易度

static 変数・メソッド、繰り返しを理解している受講生で 1 時間程度

解説のポイント

- static 変数や static メソッドはオブジェクト生成をしなくても、他のクラスで利用することが可能。ただし、private がついている場合は別のクラスで直接呼び出すことができない。
- final 修飾子が変数に付与されている場合は、処理の途中で値の変更ができない。static と組み合わせることで定数を定義することができる。
- コンストラクタ内で使用されている前置インクリメントは値を代入する前に変数の値を+1 する。
- static 変数を呼び出すときには「クラス名.変数名」で呼び出す。Static メソッドを呼び出すときは「クラス名.メソッド名()」で呼び出す。
- コンストラクタに private 修飾子が付与されている場合、そのコンストラクタを使用してオブジェクト生成を行うことができない。
- do~while 文は処理を行った後に繰り返し処理を行うか判定する。

- **challenge03**

問題の概要

以下の内容を用いて実装する。

- 継承
- スーパークラスのコンストラクタ
- ポリモーフィズム
- Scanner クラス

プログラムの動作の概要

Gladiator クラスのオブジェクトを 2 回オブジェクト生成する。その際に、引数に名前、性別、年齢、武器の名前、攻撃力、防具の名前、防御力を入れる。どちらかの HP が 0 になるまで勝負のメソッドを実行する。勝負のメソッドでは 3 つのパターンに分かれる。両者攻撃の場合は素のダメージ受ける。片方が攻撃の場合、もう片方は防御が成功するか判定する。成功した場合、防御力の分だけダメージを減らす。失敗した場合、素のダメージを受ける両者攻撃しない場合はメッセージを出力する。最後に両者の HP を出力して、次の処理に移る。攻撃や防御が成功するかどうかは Random クラスを利用する。
最後に勝者の情報を表示してプログラムを終了する。もし、同時に HP が 0 になった場合は別のメッセージを出力して、プログラムを終了する。

難易度

継承まで理解できている受講生で 1 時間程度

解説のポイント

- スーパークラスのコンストラクタを呼び出すときは「Super(引数)；」と記述する。
- Gladiator クラスが継承している Human クラスのフィールドはすべて private 修飾子がついているため Gladiator クラスから直接呼び出して、値を代入することができない。そのため Gladiator クラスのコンストラクタ内で直接値を代入することはできない。
- battle メソッド内での攻撃判定の結果は boolean 型の変数に入れて、if 文で使用する。攻撃判定のメソッドを if 文の中で直接呼び出すと if 文の条件判定の度にメソッドが実行され、予期せぬ実行結果になる可能性がある。
- main メソッド内で呼び出している battle メソッドなど Main クラスで定義したメソッドは static メソッドとする必要がある。static メソッドが呼び出せるメソッドは他の static メソッドかメソッド内でオブジェクト生成したクラスのメソッドとなる。

- **challenge04**

  問題の概要

  以下の内容を用いて実装する。

  - インターフェイス
  - 抽象クラス
  - オーバーライド
  - Scanner クラス

  プログラムの動作の概要

  コンピューターとプレイヤーでじゃんけん形式の勝負を行う。行動は攻撃、魔法、スペシャルの中から選択する。各行動をじゃんけんに当てはめると攻撃がパー、魔法がチョキ、スペシャルがグーとなる。行動を選択し、勝った方は相手に攻撃力分だけダメージを与える。先に HP が 0 以下になった方を負けとする。最後に勝った方のプレイヤーの自己紹介を出力する。

  難易度

  抽象クラスとインターフェイスまで理解できている受講生で 3 時間程度

  解説のポイント

  - キャラの HP を表示するときに for 文を使用して、配列変数 duelist の各要素の名前と HP の表示を同じメソッドで呼び出すことができている。これは型がスーパークラスやインターフェイスによってメソッド名が統一されているためである。これがポリモーフィズム(多態性)となる。
  - 決闘後のキャラの HP を表示する処理で名前を出力する際に AbstractHuman クラスにキャストしている。これは変数 player に保存されているオブジェクトはサブクラスである PlayerGladiator クラス、ComputerGladiator クラスのオブジェクトとなる。それぞれのクラスのスーパークラスである AbstractHuman クラスにキャストすることで共通のメソッド名である getName()メソッドを呼び出すことができる。

5. じゃんけんプログラム

実施タイミング

Java 下巻終了後

問題の概要

以下の内容を用いて実装する。

- 繰り返し(for 文、do~while 文)
- if~else if~else 文
- 論理演算子
- 例外、例外処理
- クラス
- メソッド
- 定数

プログラムの動作の概要

じゃんけんプログラムはすべてで 4 つのバージョンに分かれている。

- Ver1.0
  コンソール上で 0 から 2 の数値を選択し、じゃんけんの手を決める。コンピューターの手はランダムで決め、勝敗を決定する。じゃんけんは 3 回行い、勝利数に応じてメッセージを出力する。
- Ver2.0
  Ver1.0 に加えて、例外が発生した際にメッセージを出力させる。また、じゃんけんの手を選択するときに不正な値が入力されたときに再度入力を受け付け、メッセージを出力する。
- Ver3.0
  Ver2.0 までに作成したプログラムを SystemMain クラス、JankenJudge クラス、Player クラスに分けて記述しなおす。表示内容に変更はない。
- Ver4.0
  Ver3.0 に以下の内容を書き加える
  1. 対戦回数を自由に選択できる。
  2. コンピューターの手をランダムではなく、戦略性を持たせる。パターンに関しては自由だが 1 つは実装すること。(例：同じ手は出さない、相手の手をまねるなど)
  3. 対戦人数(コンピューターの数)を自由に選択できる。
  4. 上記以外のオリジナル機能の実装。

難易度

Java を理解している受講生で 12 時間程度

解説のポイント

- じゃんけんの勝敗の判定は引き分けのとき、全員グー以外のとき、全員パー以外のとき、全員チョキ以外のときに分けて判定を行う。
- 例外処理を実装するときには try-catch 文で行う。キャッチする例外は Java 7 から複数の例外を指定することが可能となった。
- 入力チェックには do-while を利用して、正しい入力がされていない間は処理を続けるような処理を行う。
- オブジェクト指向に則る際には main メソッドには最低限の処理だけ記述し、基本となる処理はすべて他のクラスに記述する。
- 対戦回数を自由に選択できるようにするために、入力された値の数だけ、じゃんけんを行う処理を繰り返すように for 文の条件を変更する。

・ 対戦人数を増やす際には入力された値の数だけコンピューターのオブジェクト生成を行い、コンピューターの手はそれぞれ List に保存する。

6. 団体じゃんけん

実施タイミング

Java 上巻終了後

問題の概要

以下の内容を用いて実装する。

・ 配列
・ 多次元配列(追加課題で必要)
・ while 文
・ 条件分岐(if~else 文、switch 文)
・ Scanner クラス
・ System.out.printf()メソッド

プログラムの動作の概要

プレイヤーとコンピューターのじゃんけんをコンソール上で行う。プレイヤーとコンピューターは先鋒から大将までの 5 人チームとする。プログラムが起動したら、コンソール上で 1 から 3 の数値を入力して、じゃんけんの手を決める。コンピューターの手はランダムに決まる。勝敗を決め、負けたチームは次の選手が出るようにメッセージを出力する。先に大将に勝ったチームが勝利とし、メッセージを出力して、プログラムが終了する。

追加課題では、勝敗が決まった後に各試合の結果を出力する。

難易度

配列、条件分岐、繰り返しを理解している受講生で 4 時間程度

解説のポイント

・ 先鋒から大将までのチームのポジションは String 型の配列に保存して管理する。
・ プレイヤーとコンピューターの現在のポジション(今は先鋒かどうかなど)はそれぞれ変数で管理しておく。
・ じゃんけんの勝敗の判定は最初にあいこかどうか(両者の手が同じ値かどうか)の判定を行い、プレイヤーの手に応じてコンピューターがどんな手のときにプレイヤーの勝利とするかの判定を行う。
・ じゃんけんの結果があいこだった時は「continue;」を使用して、以降の処理をスキップさせる。
・ 勝敗が決まった後に負けたチームのポジションの変数を+1 して、次の要素を取り出す。次の要素を取り出す際にポジションの変数が配列の要素数未満であることを条件にする必要がある。この条件がないと例外が発生する。
・ じゃんけんの処理は while 文を使用して、どちらかのポジションが選手数未満である間繰り返す。
・ 追加課題での試合結果の記録は多次元配列を使用して結果を保存する。1 次元目には試合回数(何試合目か)、2 次元目には各選手のポジション、出した手を保存する。つまり、要素数は 1 次元目が 9、2 次元目は 4 となる。
・ 試合結果を表示する際には for 文を使用して、1 次元目の添え字にループカウンタを使用する。2 次元目はそれぞれ決まっているため for 文のネストではなく、直接添え字を指定する。
・ 「System.out.printf():」は標準出力の書式を指定するメソッド。今回の模範解答での「%-3s」は左詰めで文字列を最大 3 文字分の幅で表示するという意味になる。

7. インディアンポーカー

実施タイミング

Java 上巻終了後

問題の概要

以下の内容を用いて実装する。

- 配列
- 多次元配列
- 繰り返し(for 文、while 文、do~while 文)
- 条件分岐(if~else if~else 文、switch 文)
- Scanner クラス

プログラムの動作の概要

プレイヤーとコンピューターの 2 名でのインディアンポーカーを行う。プログラムを起動するとお互いのプレイヤーはジョーカーを除いた 52 枚のトランプの中から捨て札にないカードを 1 枚ずつカードが配られる。相手の手札のスートと数字がコンソールに表示される。プレイヤーはコンソール上で整数を入力し、手札を交換するか勝負するか決める。手札の交換は最大 2 回まで行うことができる。コンピューターは交換を行わない。勝負の時にお互いの手札のスートと数字を表示し、勝敗を決める。最後に交換時に捨てたカード、勝負に使用した手札及びそれまでの捨て札をすべて表示する。これを 3 回行う。勝負を 3 回行った後、最終結果を表示して、プログラムを終了する。

難易度

配列、条件分岐、繰り返しを理解している受講生で 7 時間程度

解説のポイント

- トランプのカードは int 型の多次元配列で管理をする。1 次元目はスート、2 次元目は各スートの数字を値として持たせる。この時 A は K より強いため、数字は 1 ではなく、14 を代入する。
- プレイヤーとコンピューターの手札は int 型の配列で管理する。配列にはスートと数字の 2 つの値を代入する。
- 捨て札の管理は多次元配列を使用する。1 次元目の要素数は 4(スートの数)、2 次元目の要素数は 13(数字の数)とし、捨て札にあるときは要素の中に 1 などの値を入れる。
- プレイヤーとコンピューターにカードを配るときには捨て札にないか確認するために do~while 文を使用して、捨て札の配列の要素の値が 1 などのときはやり直すようにする。
- 数字が 11 から 14 のときはそれぞれアルファベットで表記をする必要があるため、switch 文などでアルファベットが表示されるようにする。
- 捨て札の表示は捨て札の配列の要素の値に 1 が入っている添え字と同じ添え字をトランプのカードの配列で指定して表示をする。

8. モグラたたき(配列版)

実施タイミング

Java 下巻終了後

問題の概要

以下の内容を用いて実装する。

- クラス
- メソッド
- static 選手
- 繰り返し
- 条件分岐
- 例外処理
- コンストラクタ(追加機能実装時)
- インターフェイス(追加機能実装時)
- オーバーライド(追加機能実装時)
- Math クラス(追加機能実装時)
- 配列(追加機能実装時)

プログラムの動作の概要

コンソール上でプレイヤーとコンピューターでモグラたたきを行う。プログラムを起動すると、コンソール上に 1 から 4 のマスが表示される。プレイヤーが先攻となり、1 から 4 の整数を入力する。入力された整数とモグラのいる位置が一致していれば「HIT!」と出力し、一致していない場合は「(・*・)」と出力する。次にコンピューターの番となり、ランダムで 1 から 4 の整数を選択し HIT 判定を行う。これをどちらか先にモグラをたたく、または両者ともたたくまで処理を続ける。なお、モグラが出てくる位置はランダムかつ 1 匹とする。

追加課題ではモグラの数を 3 匹に増やし、マスも 9 マスにする。HIT したモグラはその後出てこないようにし、先に全部のモグラを叩いた方が勝利とする。

難易度

Java を理解できている受講生で 10 時間程度

解説のポイント

- モグラの配置からコンピューターの結果表示までは do~while 文を使用して実装する。繰り返し条件は両者モグラを当てられていないこと。
- モグラの位置を決定するときは Math クラスの random メソッドを使用する。これは 0.0 から 1.0 の中からランダムで値を返す。これを 10 倍して、4 で割ったときの余り、つまり 0 から 3 の値で位置を決定する。
- モグラの位置は Player クラスのフィールドの Mole クラスの変数 mole の中に入っているため、モグラの位置を return する必要がない。
- static メソッドを呼び出すときには「クラス名.メソッド名」で呼び出す。
- インターフェイスの実装はクラス名に「implements インターフェイス名」で指定する。
- インターフェイス内の抽象クラスは必ずオーバーライドしなくてはいけない。
- User クラスと Computer クラスには Mole クラス型の配列をフィールドとして持たせる。叩かれているかの判定の保存は Mole クラスのフィールドである isHitFlag で行う。
- 追加機能ではモグラの配置からコンピューターの結果表示までの繰り返し条件は、haveMoles メソッド(モグラが残っているかどうかのメソッド)が true かどうかで判定する。

9. オブジェクトじゃんけん

実施タイミング

Java 下巻 17 章終了後

問題の概要

以下の内容を用いて実装する。

- クラス
- メソッド
- コンストラクタ
- Math クラス
- 条件分岐
- 繰り返し

プログラムの動作の概要

コンソール上で 2 名のプレイヤーがじゃんけんを行う。最初にコンソール上に何回戦目か表示する。各プレイヤーの手はランダムに決まり、勝敗を決定する。これを 3 回繰り返す。最後に両者の勝利数を比較し、結果に応じてメッセージを出力する。

難易度

クラス、メソッドまで理解できている受講生で 7 時間程度

解説のポイント

- プレイヤーの手をランダムに決める際に使用した Math クラスの random メソッドは 0.0 から 1.0 の中からランダムで値を返すメソッド。今回はそれを 3 でかけることで 0.0 から 3.0 の数字をランダムで出るようにした。
- じゃんけんの勝敗は条件分岐を使って、プレイヤー 1 が勝つ条件を「|||」でつなげて判定し、あいこの場合は勝者なしとする。
- 勝利数の判定についても勝者がいるときはそちらのオブジェクトを返し、引き分けの場合はオブジェクトを null で返す。
- 最後の結果の表示の際は勝者が null でなければそのオブジェクトの情報を表示し、null であれば別のメッセージを出力するようにする。

10. モグラたたき(コレクション版)

実施タイミング

Java 下巻終了後

問題の概要

以下の内容を用いて実装する。

- クラス
- メソッド
- static 選手
- 繰り返し
- 条件分岐
- 例外処理
- コンストラクタ(追加機能実装時)
- インターフェイス(追加機能実装時)
- オーバーライド(追加機能実装時)
- Math クラス(追加機能実装時)
- コレクションフレームワーク(追加機能実装時)

プログラムの動作の概要

コンソール上でプレイヤーとコンピューターでモグラたたきを行う。プログラムを起動すると、コンソール上に 1 から 4 のマスが表示される。プレイヤーが先攻となり、1 から 4 の整数を入力する。入力された整数とモグラのいる位置が一致していれば「HIT!」と出力し、一致していない場合は「(・*・)」と出力する。次にコンピューターの番となり、ランダムで 1 から 4 の整数を選択し HIT 判定を行う。これをどちらか先にモグラをたたく、または両者ともたたくまで処理を続ける。なお、モグラが出てくる位置はランダムかつ 1 匹とする。

追加課題ではモグラの数を 3 匹に増やし、マスも 9 マスにする。HIT したモグラはその後出てこないようにし、先に全部のモグラを叩いた方が勝利とする。

難易度

Java を理解できている受講生で 10 時間程度

解説のポイント

- モグラの配置からコンピューターの結果表示までは do~while 文を使用して実装する。繰り返し条件は両者モグラを当てられていないこと。
- モグラの位置を決定するときは Math クラスの random メソッドを使用する。これは 0.0 から 1.0 の中からランダムで値を返す。これを 10 倍して、4 で割ったときの余り、つまり 0 から 3 の値で位置を決定する。
- モグラの位置は Player クラスのフィールドの Mole クラスの変数 mole の中に入っているため、モグラの位置を return する必要がない。
- static メソッドを呼び出すときには「クラス名.メソッド名」で呼び出す。
- インターフェイスの実装はクラス名に「implements インターフェイス名」で指定する。
- インターフェイス内の抽象クラスは必ずオーバーライドしなくてはいけない。
- User クラスと Computer クラスには Mole クラス型の List をフィールドとして持たせる。叩かれているかの判定の保存は Mole クラスのフィールドである isHitFlag で行う。
- 追加機能ではモグラの配置からコンピューターの結果表示までの繰り返し条件は、haveMoles メソッド(モグラが残っているかどうかのメソッド)が true かどうかで判定する。

11. 浅草ジャマイカホール①

実施タイミング

Java 上巻終了後

問題の概要

以下の内容を用いて実装する。

- 配列
- 入出力
- デクリメント
- if~else 文
- for 文(追加機能実装時)

プログラムの動作の概要

コンソール上で 1 から 30 の座席番号を整数で入力する。入力した座席番号が予約済みの場合は予約済みであるメッセージを出力し、予約可能であれば予約が完了したメッセージを出力してシステムを終了する。なお、予約済みの席はプログラムの中で事前に決めておく。

追加機能ではシステム実行時に残りの座席数に応じて異なるメッセージを出力する。

難易度

Java 上巻まで理解できている受講生で 1 時間 30 分程度

解説のポイント

- 座席の情報は boolean 型の配列で管理を行う。このとき要素数は座席数の 30 を指定する。
- 座席番号を入力された後、入力値をデクリメントする。入力値を使って配列の添え字を指定するため 0 から 29 に補正する必要がある。
- 入力された座席番号は「Integer.parseInt()」メソッドを使用して、String 型から int 型に変換する。
- 座席が予約されているかどうかの条件では指定された配列の要素の値で判定を行う。
- 予約されている座席を数えるときは for 文を使用して、配列のすべての要素の値を確認し、予約済みになっている座席数を数える。
- 座席数に応じたメッセージは予約されている座席が 30 席、16 席以上、15 席以下のときで異なるため、「else if」を使用して、条件分岐させる。

12. 浅草ジャマイカホール②

実施タイミング

Java 下巻終了後

問題の概要

以下の内容を用いて実装する。

- 配列
- 条件分岐
- 繰り返し
- クラス
- メソッド
- 例外
- コレクションフレームワーク

プログラムの動作の概要

浅草ジャマイカホール②はバージョンによって動作が変わる。

- Ver1.0
  コンソール上に 1 から 30 の座席番号を整数で入力する。指定した座席が予約済みか空席かどうかで異なるメッセージを出力し、プログラムを終了する。
- Ver1.1
  Ver1.0 に加えて、IOException が発生した際の例外処理として、メッセージを出力する。
  さらに、入力した座席番号が予約済みだった場合、メッセージを出力し。再度座席番号を入力させる。
- Ver2.0
  座席予約時に座席番号とユーザー番号を入力し、保存できるようにする。
  また、Seat クラスを新しく作成し、Ver1.1 まで配列で管理していた予約情報を Seat クラス型の List で管理するようにする。
  ソースコードに直接書いていた値などはすべて定数に置き換える。
- Ver3.0
  入力する内容を名前、電話番号、メールアドレス、性別、年齢に変更する。座席番号は入力せず以下のロジックを使用して、座席の予約を行う。数字が低いもの程優先順位が高い。
  1. 両側の座席が空いている席を選択する。(1 番と 30 番は選択しない)
  2. 両側の座席に座っている人が同性である席を選択する。(1 番と 30 番は選択しない)
  3. どこでもいいので空いている席を予約する。

難易度

Java を理解できている受講生で 1 時間 30 分程度

解説のポイント

- 座席の情報は boolean 型の配列で管理を行う。このとき要素数は座席数の 30 を指定する。
- 座席番号を入力された後、入力値をデクリメントする。入力値を使って配列の添え字を指定するため 0 から 29 に補正する必要がある。
- 入力された座席番号は「Integer.parseInt()」メソッドを使用して、String 型から int 型に変換する。
- 座席が予約されているかどうかの条件では指定された配列の要素の値で判定を行う。
- 予約されている座席を数えるときは for 文を使用して、配列のすべての要素の値を確認し、予約済みになっている座席数を数える。
- 例外が発生する可能性のある処理を try ブロックの中に入れる。catch ブロックの引数には例外処理を行いたい

例外クラスを指定する。ブロックの中には引数で指定した例外が発生したときに行う処理を記述する。

- 指定した座席が予約されているときに再度入力させるときは「do~while」を使用する。
- Java で定数を作成するときはフィールドに「static final」を指定する。
- List や配列は整数や文字列以外にもクラスを型として指定することができる。この時、値はオブジェクトの参照になる。
- コレクションフレームワークでは「<>」の中に値の型を指定する。
- 両側の席が予約済みかどうかは for 文で 2 番目の席から 29 番目の席までをすべて調べる。調べる方法は現在のインデックスの前後の座席の予約情報を取得し、予約済みか判定する。
- 両側の席が同性かどうかの判定は両側の席が空席かどうかの判定と同じロジックを使用する。違いは調べるときに空白かどうかではなく、Seat クラスの User の gender と現在予約しようとしているユーザーの gender が一致しているかで判定を行う。