

## 第十一回 演習問題（デバッグ等）

## 諸注意

- 「Kadai1.java」, 「Monster.java」, 「MonsterManager.java」, 「Clicker.java」の4ファイルを Web から提出する。
- 指定した処理が正常に動くようであれば、適宜独自メソッド等を実装してくれて構わない。
- コピペ発覚時は見せた側も見せてもらった側も両方0点とする。
- 必ず**コンパイルエラーのない状態で提出すること**（自動採点したいのでコンパイルエラーがあると、全て0点になってしまう）。
- 課題の途中で提出することになった場合、**コンパイルエラーさえ出なければ、課題の途中の状態でも提出してくれて構わない**。一部のメソッドだけが実現できていない場合、コンパイルエラー出ないならばそのままの状態でも提出してくれてよい。
- 主にコンソール出力で評価しているため、デバッグに用いたようなコンソール出力が残っていないように気をつけること。**基本的にコンソール出力を指定しない限りは、課題内でコンソール出力はないものとする**。
- **Package は使わないこと**（デフォルトパッケージで実装する）。Package で実装すると、自動採点がうまくいきません。

## 課題 1

1	問題設定	引数の list から最大値を取得するメソッド getMax() を部下が開発したのだがどうも間違っている様子である。次の手順でデバッグせよ。getMax() は以下のコードを Kadai1.java に張り付けること。
	getMax()	<pre> public static int getMax(int[] array){     int max = 0;     for(int val : array){         if(max &gt; val) max = val;     }     return max; } </pre>
	諸注意	<ul style="list-style-type: none"> <li>● メソッドは static とすること。</li> <li>● メソッドは全クラス、全パッケージからアクセスできるようにすること。</li> </ul>
	手順 1	<p>何か誤っているのか次の void check() メソッドの実装を通して見極めよ。なお、半角スペース等は不要であり、[改行]は改行コードを意味する。</p> <p>以下 void check() の処理手順とする。</p> <ol style="list-style-type: none"> <li>1. int 型配列 array を作成する。 (初期値は 0,1,2,3,4,5,6,7,8,9 とする)</li> <li>2. 次の書式で array の中身をコンソール出力する。 0,1,2,3,4,5,6,7,8,9,[改行]</li> <li>3. getMax(array) を println() でコンソール出力する。</li> </ol>
	手順 2	<p>手順 1 の結果を経て、デバッガの機能を用いて getMax() をデバッグせよ。デバッガの機能を使ったかどうかを我々が確認するすべはないが、是非練習だと思って</p> <ol style="list-style-type: none"> <li>1. ブレークポイントを立てて</li> <li>2. デバッグを実行し</li> <li>3. ステップ・イン、ステップ・オーバーを活用し</li> <li>4. バグの原因箇所を見極めよ。</li> </ol>
	手順 3	<p>引数によっては最大値が定義できない場合が存在する。</p> <ol style="list-style-type: none"> <li>1. どのような引数のケースか考察せよ。</li> <li>2. getMax() のはじめて該当の引数を if 文で捉え、意図的に IllegalArgumentException を発生させよ。 エラーメッセージは「引数が異常です」とせよ。</li> </ol>
	採点基準	<p>0. check() が正しく実装されている。</p> <ol style="list-style-type: none"> <li>1. バグ 1 が改善されている。</li> <li>2. バグ 2 が改善されている。</li> <li>3. バグ 3 が改善されている。</li> </ol> <p>※バグは実は 3 つある。</p>

## 課題 1：手順 1 解答例 (Kadai1.java)

```
public static void check(){
    int[] array = {0,1,2,3,4,5,6,7,8,9};
    for(int i:array) System.out.print(i + ",");
    System.out.println();
    System.out.println(getMax(array));
}
```

## 課題 1：手順 2 解答例 (Kadai1.java)

1. int max = 0; の行あたりにブレークポイントを立ててデバッグを行う。
2. ステップオーバーしつつ変数の値を確認していくと、max の値が更新されない事がわかる。
3. for 文の中をよく見ると、if(max > val) の向きが逆であることがわかる。
4. if の条件を修正する。

## 課題 1：手順 2. 5 解答例 (Kadai1.java)

1. 入力する int 型配列の要素を変えて再度確認してみることにする。
2. 例えば int[] array = {-5, -4, -3}; を入力して実施してみると出力が 0 になる。
3. デバッガの機能を用いて変数を確認すると、max の初期値が原因であることが判明する。
4. max の初期値を int 型の最小値にする（または array[0] を用いても良いが、その場合 array の null チェックが必要になる）。

## 課題 1：手順 3 解答例 (Kadai1.java)

1. 授業中に補足したと思うが、異常な入力は int[] array = new int[0] が与えられたときである。
2. この時現状の getMax() では入力の長さが 0 にもかかわらず 0 という解を返してしまうが、本来は入力異常なので例外落ちすることが望ましい。
3. ということで最終的には以下のコードとなる。

```
public static int getMax(int[] array){
    // 手順 2 対応：初期値が異常
    int max = Integer.MIN_VALUE;
    // 手順 3 対応：引数が new int[0] だった時の対応
    if(array.length <= 0)
        throw new IllegalArgumentException("引数が異常です");

    for(int val : array){
        // 手順 2 対応：max > val を max < val に修正
        if(max < val) max = val;
    }

    return max;
}
```

## 課題 2

2-1	問題設定	StringBuilder を用いた高速化を色々試してみよう.
	クラス名	Monster
	フィールド	name : 名前 (文字列型) hp : 体力 (int 型) ap : 攻撃力 (int 型) mp : 魔法力 (int 型) com : コメント (文字列型)
	コンストラクタ	上記すべてのフィールドを上から順に初期化する.
	諸注意	● フィールドは全て隠蔽すること.
	メソッド	<pre>public String toStringPlus()     テスト例のような文字列を返す.     ただし, StringBuilderは使わずに+で結合する.  public String toStringBuilder()     テスト例のような文字列を返す.     ただし, 文字列はStringBuilderを用いて結合する.</pre> <p>※テスト例の括弧は半角であり, 括弧閉じの後にのみ半角スペースが1つ挿入されるものとする.</p>
	テスト例	<p>Main.javaのmain()メソッドにでも貼り付けて実行.</p> <pre>Monster m = new Monster("はせがわ", 10, 50, 2, " 期末テスト頑張って. "); System.out.println(m.toStringPlus()); System.out.println(m.toStringBuilder());</pre>
	テスト例出力	<p>はせがわ[10,50,2] 期末テスト頑張って.  はせがわ[10,50,2] 期末テスト頑張って.</p>

## 課題 2-1 : 解答例 (Monster.java)

```
public class Monster {
    public static void main(String[] args){
        Monster m = new Monster("はせがわ", 10, 50, 2, "期末テスト頑張っ");
        System.out.println(m.toStringPlus());
        System.out.println(m.toStringBuilder());
    }
    private String name = "";
    private int hp = 0;
    private int ap = 0;
    private int mp = 0;
    private String comment = "";
    public Monster(String name, int hp, int ap, int mp, String comment){
        this.name = name;
        this.hp = hp;
        this.ap = ap;
        this.mp = mp;
        this.comment = comment;
    }

    public String toStringPlus(){
        return this.name + "[" + this.hp + "," + this.ap + ","
            + this.mp + "]" + this.comment;
    }

    public String toStringBuilder(){
        StringBuilder sb = new StringBuilder();
        sb.append(this.name);
        sb.append("[");
        sb.append(this.hp);
        sb.append(",");
        sb.append(this.ap);
        sb.append(",");
        sb.append(this.mp);
        sb.append("] ");
        sb.append(this.comment);
        return sb.toString();
    }
}
```

2-2	問題設定	StringBuilder を用いた高速化を色々試してみよう。
	クラス名	MonsterManager
	static フィールド	monsters : モンスターの一覧を管理するリスト (ArrayList<Monster>型)
	諸注意	<ul style="list-style-type: none"> <li>● フィールドは全て隠蔽すること。</li> <li>● メソッドはどこからでもアクセスできること。</li> <li>● フィールドもメソッドも全て static とする。</li> </ul>
	メソッド	<pre>void add(Monster m)     引数のインスタンスをstaticフィールドに追加する。  String toStringPlusPlus()     staticフィールドの中身全てを半角カンマ区切りで結合し戻り値として返す。ただし、各要素を文字列化するとき     にtoStringPlus()を用い、各要素間の結合にも+を用いよ。  String toStringPlusBuilder()     staticフィールドの中身全てを半角カンマ区切りで結合し戻り値として返す。ただし、各要素を文字列化するとき     にtoStringBuilder()を用い、各要素間の結合には+を用いよ。  String toStringBuilderPlus()     staticフィールドの中身全てを半角カンマ区切りで結合し戻り値として返す。ただし、各要素を文字列化するとき     にtoStringPlus()を用い、各要素間の結合には     StringBuilderを用いよ。  String toStringBuilderBuilder()     staticフィールドの中身全てを半角カンマ区切りで結合し戻り値として返す。ただし、各要素を文字列化するとき     にtoStringBuilder()を用い、各要素間の結合にも     StringBuilderを用いよ。</pre>
	テスト例	<p>Main.javaのmain()メソッドにでも貼り付けて実行</p> <pre>MonsterManager.add(new Monster("は", 10, 50, 2, "期末")); MonsterManager.add(new Monster("せ", 10, 50, 2, "ですと")); MonsterManager.add(new Monster("か", 10, 50, 2, "かんぱ")); MonsterManager.add(new Monster("わ", 10, 50, 2, "って"));  System.out.println(MonsterManager.toStringPlusPlus()); System.out.println(MonsterManager.toStringPlusBuilder()); System.out.println(MonsterManager.toStringBuilderPlus()); System.out.println(MonsterManager.toStringBuilderBuilder());</pre>
	テスト例 出力	<pre>は[10,50,2] 期末,せ[10,50,2] ですと,か[10,50,2] かんぱ,わ[10,50,2] って, は[10,50,2] 期末,せ[10,50,2] ですと,か[10,50,2] かんぱ,わ[10,50,2] って, は[10,50,2] 期末,せ[10,50,2] ですと,か[10,50,2] かんぱ,わ[10,50,2] って, は[10,50,2] 期末,せ[10,50,2] ですと,か[10,50,2] かんぱ,わ[10,50,2] って,</pre>

## 課題 2-2：解答例 (MonsterManager.java)

```
import java.util.ArrayList;

public class MonsterManager {
    private static ArrayList<Monster> monsters = new ArrayList<Monster>();

    public static void add(Monster m){
        monsters.add(m);
    }

    public static String toStringPlusPlus(){
        String res = "";
        for(Monster m:monsters){
            res += m.toStringPlus() + ",";
        }
        return res;
    }

    public static String toStringPlusBuilder(){
        String res = "";
        for(Monster m:monsters){
            res += m.toStringBuilder() + ",";
        }
        return res;
    }

    public static String toStringBuilderPlus(){
        StringBuilder sb = new StringBuilder();
        for(Monster m:monsters){
            sb.append(m.toStringPlus());
            sb.append(",");
        }
        return sb.toString();
    }

    public static String toStringBuilderBuilder(){
        StringBuilder sb = new StringBuilder();
        for(Monster m:monsters){
            sb.append(m.toStringBuilder());
            sb.append(",");
        }
        return sb.toString();
    }
}
```

## 課題 3 (サービス問題：課題 2 までで既に 100 点ある)

3	問題設定	<p>下記の URL から働クリッカーというアプリを DL し使ってみよ。働クリッカーは 5 年ほど昔に一瞬流行った「クッキークリッカー」に類似するアプリである。</p> <p>働クリッカーの特徴はここから先にある。同アプリでは JavaScript でチートコードを入力し、プログラムに働かせることができるのである。詳細は URL を参照してほしいが、基本動作は下記に示しておく。</p>
	DL 先 URL	<p>働クリッカー</p> <p><a href="http://hsgw-nas.fuis.u-fukui.ac.jp/files/Hatara-Clicker-win64.zip">http://hsgw-nas.fuis.u-fukui.ac.jp/files/Hatara-Clicker-win64.zip</a></p> <p>働クリッカー配布元 (説明等)</p> <p><a href="https://zeny.io/products/working-clicker/">https://zeny.io/products/working-clicker/</a></p>
	使用方法	<p>【普通に遊ぶ】</p> <ul style="list-style-type: none"> <li>・「ゲームを始める」をクリックする</li> <li>・「働く」をクリックするとお金が稼げる。</li> <li>・資格等を購入すると時給が上がったりする。</li> </ul> <p>【チートコード】</p> <ul style="list-style-type: none"> <li>・「C」のキーを押すとコンソールが開く</li> <li>・コンソール上部にコードを記述する。</li> <li>・Restart→Run でプログラムを実行できる。</li> <li>・総資産 1 億で一応ゲームクリア。</li> <li>・コンソール右上の?にリファレンスがある。</li> </ul>
	諸注意	<p>基本的に Java や C 言語と同じように記述できるが、変数宣言時には型名を全て var とする点だけ JavaScript は特殊である。</p> <p>なお、本日学習した高速化はあまり関係ない。ただ面白いからサービス問題として載せておくだけである。</p>
	採点基準	<p>ゲームクリアに 10 秒をきった場合、提出を受け付ける。該当のソースコードをコピーし、「Clicker.java」として提出せよ (Java のコードじゃないのでエラーが出るかもしれないが気にしなくて良い)。</p> <p>なお、長谷川が適当に遊んだ結果、長谷川の保有する PC のスペックで、0.409s を達成した。</p>