クラス

ゲームプログラミングA #02

向井 智彦

先週のおさらい

- Unity & C# 導入
  - Roll-a-Ball
  - 演算、入出力、変数、条件分岐
- やらなかったこと
  - Debugクラス、MonoBehaviourクラス
  - public
  - Startメソッド、Updateメソッド

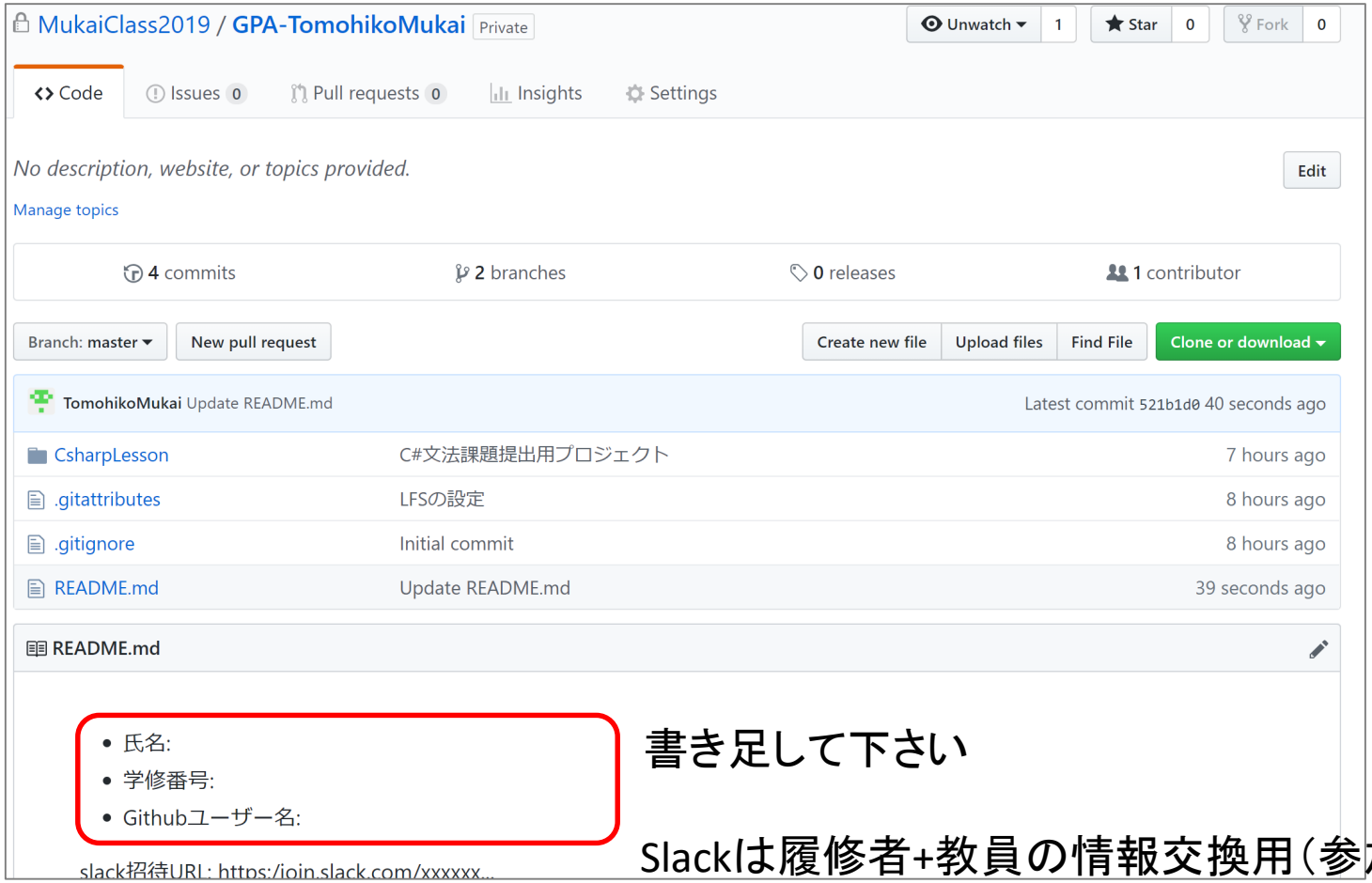
今週の目標

- C#プログラミング（配列、反復、メソッド）

- クラスの概要の理解
- Unity C#クラスの利用

事前準備: 履修者情報の追記

- MukaiClass2019/GPA-〇〇



書き足して下さい

Slackは履修者+教員の情報交換用（参加は任意）

事前準備: 演習用プロジェクトの取得

- MukaiClass2019 / GPA-〇〇 / CsharpLesson

ゲームプログラミング演習・実習A (2019)

- 木2-3@2-501
- 首都大学東京システムデザイン学部インダストリアルアートコース 2019年度春期
- 担当教員： 向井智彦（email: mukai@tmu.ac.jp, 日野1号館2階247室）
- 講義全体に関する質問は 講義ページのIssueへ記載して下さい．個別課題に関する質問はそれぞれのIssueへ記載して下さい．

アナウンス

- [2019/04/11] Unityの授業標準バージョンは、演習室環境に合わせて「2018.3.12f1」とします。

スケジュール

第1回目：ガイダンス

1. スライド
2. Unityのインストール(macos)
3. GitHub Desktopのインストール(macos)

第2回：クラス

1. スライド
2. 課題レポジトリのクローン

プログラミング基礎(C/C++言語)履修者向け

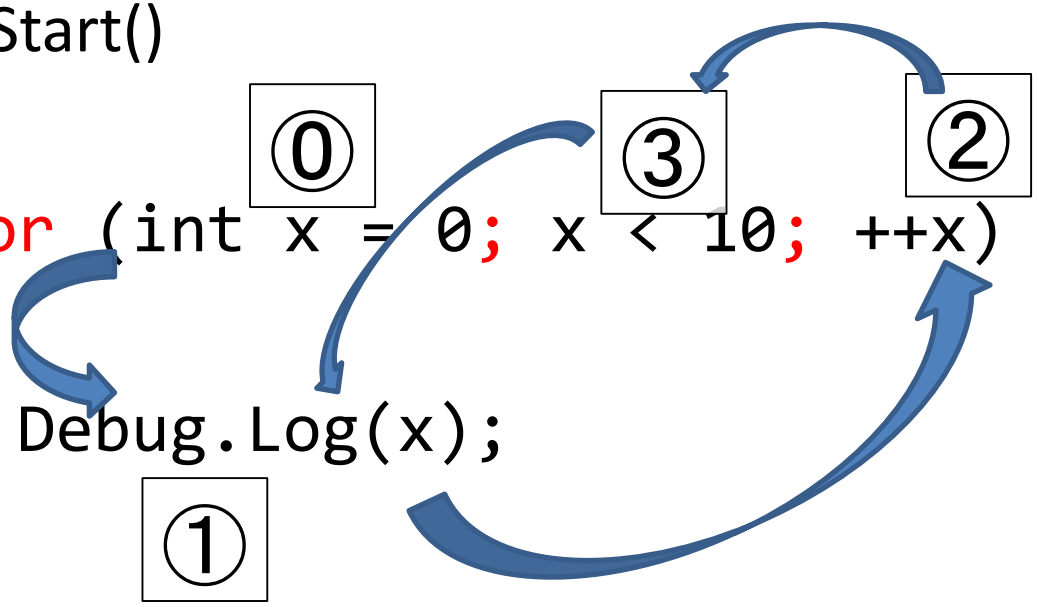
C#プログラミング「反復」

「n回だけ繰り返す」をfor文で

```
using UnityEngine;
public class TestScript : MonoBehaviour {
   void Start()
   {
      for (int x = 0; x < 10; ++x)
      {
         Debug.Log(x);
      }
   }
}
```

for文は反時計回りに読み書き

```
using UnityEngine;
public class TestScript : MonoBehaviour {
   void Start()
   {
      for (int x = 0; x < 10; ++x)
      {
         Debug.Log(x);
      }
   }
}
```



for文の動作図解

```
using UnityEngine;
public class TestScript : MonoBehaviour {
  void Start()
  {
    for (int x = 0; x < 10; ++x)
    {
      Debug.Log(x);
    }
  }
}
```

⓪初期設定

for文の動作図解

```
using UnityEngine;
public class TestScript : MonoBehaviour {
  void Start()
  {
    for (int x = 0; x < 10; ++x)
    {
      Debug.Log(x);
    }
  }
}
```

①反復対象処理の実行

for文の動作図解

```
using UnityEngine;
public class TestScript : MonoBehaviour {
  void Start()
  {
    for (int x = 0; x < 10; ++x)
    {
      Debug.Log(x);
    }
  }
}
```

②次の反復に
移る際に
実行される式

for文の動作図解

```
using UnityEngine;
public class TestScript : MonoBehaviour {
  void Start()
  {
    for (int x = 0; x < 10; ++x)
    {
      Debug.Log(x);
    }
  }
}
```

③反復の継続判定

for文の動作図解

```
using UnityEngine;
public class TestScript : MonoBehaviour {
  void Start()
  {
    for (int x = 0; x < 10; ++x)
    {
      Debug.Log(x);
    }
  }
}
```

①反復対象処理の実行　に戻る

「N回繰り返す」をwhile文で

```
using UnityEngine;
public class TestScript : MonoBehaviour {
   void Start()
   {
      int x = 0; //カウント0スタート
      while (x < 10)
      {
         Debug.Log(x);
         ++x; //カウントを1つずつ増やす
      }
   }
}
```

条件式の更新

```
void Start()
{
   int x = 0;
   while (x < 10)
   {
      Debug.Log(x);
   }   //無限ループ!!!
}
```

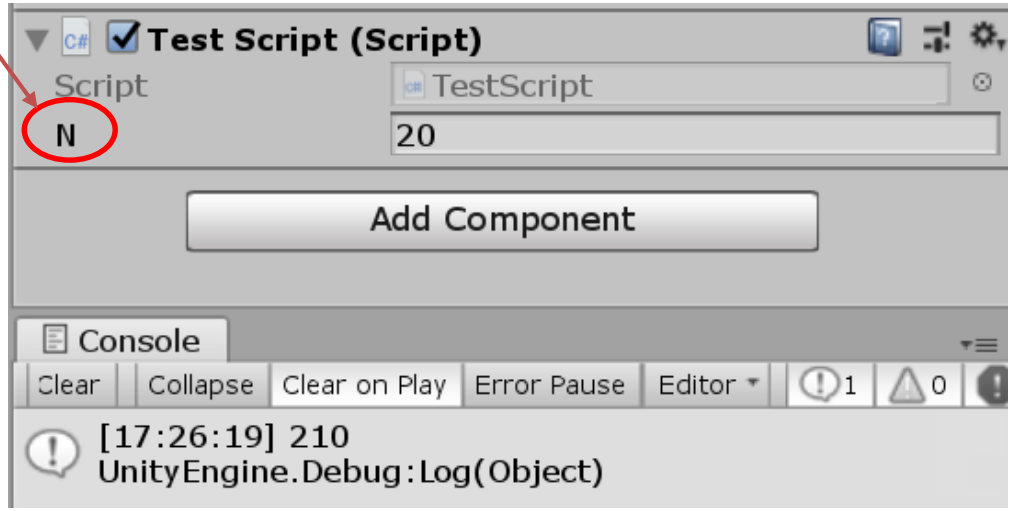
```
void Start()
{
   int x = 0;
   while (x < 10)
   {
      Debug.Log(x);
      ++x; //カウントアップ
   }
}
```

ポイント:

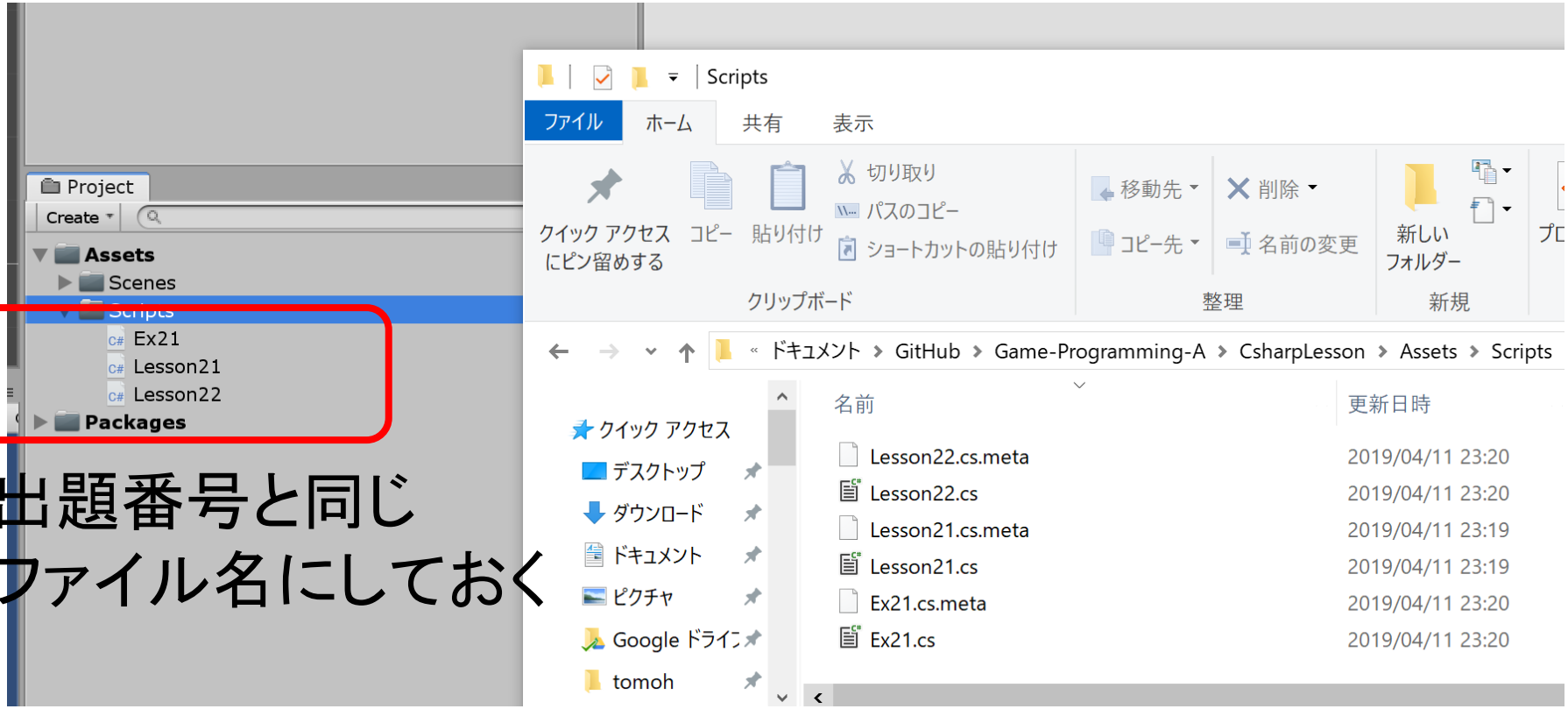
条件式に登場する変数を、反復のたびに更新

Lesson21

- １～nの正の整数の合計値を計算するC#スクリプトを作成
  - nは指定された正の整数



LessonおよびExの提出方法



プログラミング基礎(C/C++言語)履修者向け

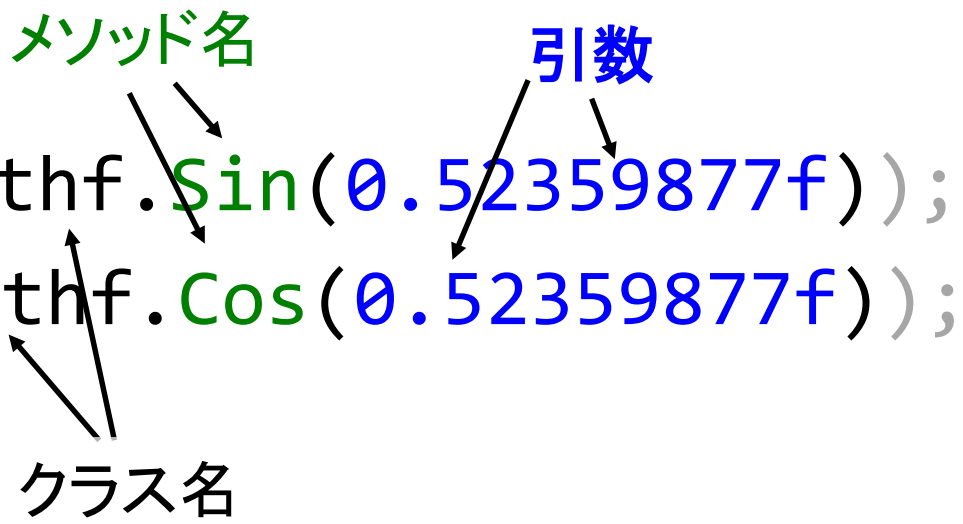
C#プログラミング「メソッド」

三角関数の例

```
using UnityEngine;
public class TestScript : MonoBehaviour {
   void Start()
   {
      Debug.Log(Mathf.Sin(0.52359877f));
      Debug.Log(Mathf.Cos(0.52359877f));
    }
}
```

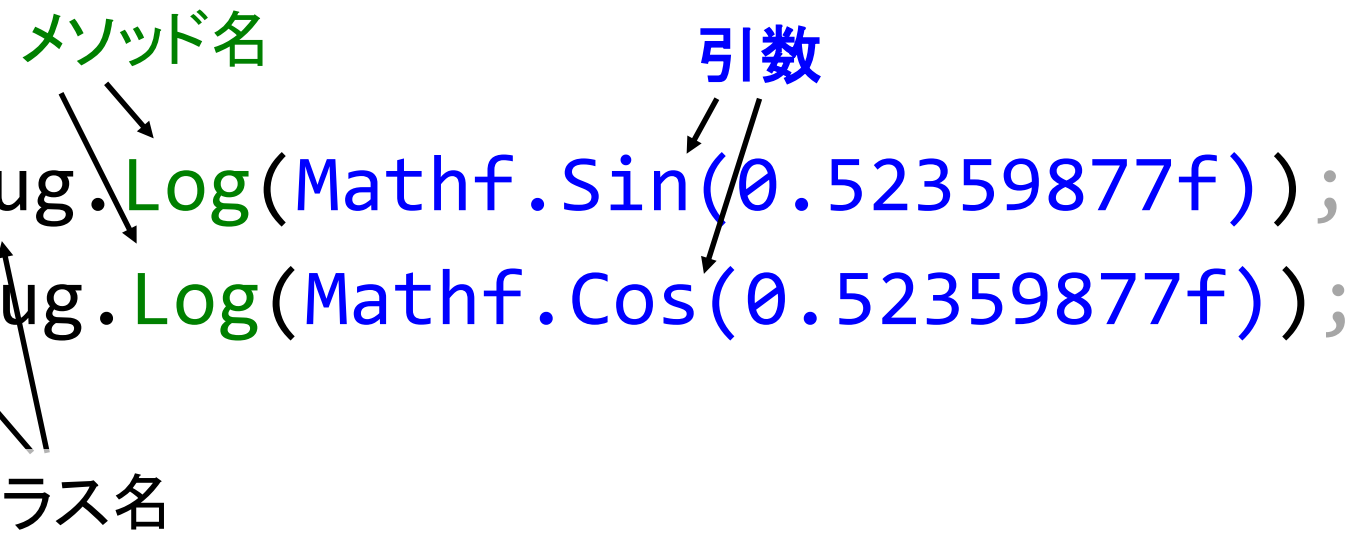
文法図解

```
using UnityEngine;
public class TestScript : MonoBehaviour {
   void Start()
   {
      Debug.Log(Mathf.Sin(0.52359877f));
      Debug.Log(Mathf.Cos(0.52359877f));
   }
}
```



文法図解

```
using UnityEngine;
public class TestScript : MonoBehaviour {
   void Start()
   {
      Debug.Log(Mathf.Sin(0.52359877f));
      Debug.Log(Mathf.Cos(0.52359877f));
   }
}
```



メソッド(関数)とは？

- 別のメソッドから引数を添えて呼ばれ、
- 何かしらの処理を実行し、
- 呼び出し元に何らかの値を戻す、
- 処理のひとまとまり

メソッドの基本形

```
float Sin(float) { return sin_value; }
```

↓

戻り値の型名　メソッド名 (パラメータ)　{　処理;　}

メソッドの動作イメージ

1. `Debug.Log(Mathf.Sin(0.52359877f));`

   最も内側の小括弧内を最優先に処理

2.「 `Mathf.Sin(0.52359877f)` 」の計算

   → 戻り値`0.5`が得られる

   = メソッド呼び出しが数値`0.5`で置き換わる

3. `Debug.Log(0.5)`

   → console に「`0.5`」が表示される

Lesson22

- 「Debug.Log」メソッドの代わりに「Debug.Warning」メソッドおよび「Debug.Error」メソッドを用い、その挙動の違いを調査
  - 使い方は同じ
  - 出力するメッセージは何でもOK
    - 困ったときの”Hello World”

プログラミング基礎(C/C++言語)履修者向け

C#プログラミング「配列」

C#の配列はC++の配列と何か違う

https://docs.microsoft.com/ja-jp/dotnet/csharp/programming-guide/types/

C#

- 宣言（初期値指定）
  - int[] a = {0, 1, 2};
- 宣言（初期値なし）
  - int[] a = **new** int[3];

説明は後ほど

- アクセス
  - a[0] = -1;
  - int x = a[2];

C++

- 宣言（初期値指定）
  - int a[3] = {0, 1, 2};
- 空の配列（初期値なし）
  - int a[3];
- アクセス
  - a[0] = -1;
  - int x = a[2];

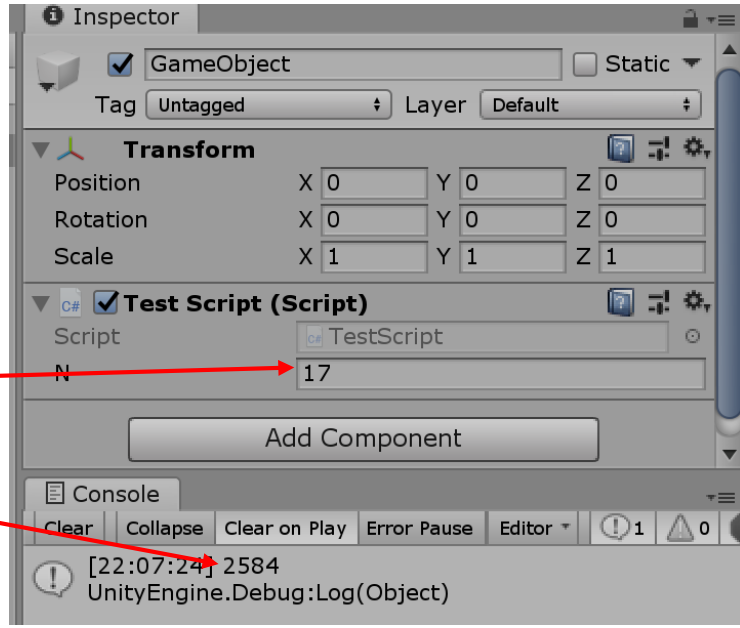
配列と反復: 合計値

```
using UnityEngine;
public class TestScript : MonoBehaviour {
  public int[] array = new int[5];
  void Start() {
    int sum = 0;
    for (int i = 0; i < 5; ++i) {
      sum += array[i];
    }
    Debug.Log(sum);
  }
}
```

Lesson23

- n番目の要素が(n - 1)番目と(n - 2)番目の要素の和（ただし0番目と1番目の要素は1）となるような、20個の要素を持つ配列を作成
  - つまりフィボナッチ数列

- 指定された添字（19以下）の要素を出力するC#スクリプト
  - a[0] = 0
  - a[17] = 2548;



クラス

データ型

https://docs.microsoft.com/ja-jp/dotnet/csharp/language-reference/keywords/types

- 整数型　　int
- 浮動小数点型　　float, double
- ブール型　　bool
  - その他 short, long, byte, uint, など

- 文字列型　　string
- ユーザー定義型クラス

クラス：　複数のメンバー（フィールドやメソッド）をまとめた型

```
1.  int a;
2.  float b;
3.  double c;
4.  char d;
```

```
1.  class data
2.  {
3.      int a;
4.      float b;
5.      double c;
6.      char d;
7.  }
```

※文法は後ほど説明

関連する複数フィールドの組合せ例

- 数学
  - 2次元座標，3次元座標，複素数，ベクトル
- 個人データ
  - 氏名，年齢，生年月日，住所，マイナンバー…
- 家族構成
  - 父，母，子，孫…
- 地理データ
  - 都市名，人口，面積，標高

Quick exercise 1

- 日野キャンパスを表す「数字」を複数挙げて下さい
  - 例）学生数

Quick exercise 2

- 先ほど挙げた「日野キャンパスを表す『数字』」から5つほど選択し，それらをC言語の変数で表すために，

  それぞれに適した型と

  わかりやすい変数名を考えて下さい．

Quick exercise 3

作成した5つの変数を，空欄に記入して下さい.

```
class Campus7Number
{




};
```

本日用いるクラスその１：Vector2

https://docs.unity3d.com/ja/current/ScriptReference/Vector2.html

- 2次元ベクトルクラス
  - 2Dゲーム開発時に多用
    - 2次元位置、2次元方向、2次元移動量 etc...
  - 基本的な数学演算を提供
    - 四則演算、比較演算、ベクトル長、内積 etc...

- 2つの要素 [x, y] はいずれも float 型
  - 代入する浮動小数値の最後には" f "を付加
    - 20.0f 、-157.7f など

X座標とY座標へのアクセス

1. Vector2 a = new Vector2()
   // Vector2クラスインスタンス a
2. a.x = 10.0f; // a + ピリオド + フィールド名 x
3. a.y = 20.0; // a + ピリオド +フィールド名 y
   // ※右辺がdouble型なのでエラー
4. Debug.Log(a);

同じ働きをするプログラム

```
float ax;
float ay;
ax = 10.0f;
ay = 20.0f;
float bx = 5.0f;
float by = -6.0f;
float cx = ax + bx;
float cy = ay + by;
Debug.Log(cx+" "+cy);
```

```
Vector2 a = new Vector2();
a.x = 10.0f;
a.y = 20.0f;
Vector2 b =
    new Vector2(5.0f, -6.0f);
Vector2 c = a + b;
Debug.Log(c);
```

スッキリまとまって読みやすい（ようになる）

クラスインスタンスの生成

変数として利用できる状態になったクラスの実体

基本型インスタンス

- int a;
- float b;
- char c;

配列型インスタンス

- int[] a = new int[10];
- float[] b = new float[10];
- char[] c = new char[10];

ユーザークラスインスタンス

- Player player = new Player();
- Enemy enemy = new Enemy();
- Bomb ;bomb = new Bomb();
- Campus7Number hino =
  new Campus7Number();

newによるクラスインスタンス生成

- `Vector2 a = new Vector2();`
- `Vector2 b = new Vector2(1.0f, 1.0f);`

- `int c;` ⇔ `int c = new int();`
- `float d;` ⇔ `float d = new float();`
- `string e;` ⇔ `string e = new string();`
- `dobule[] f = {0.0, 1.0};`
  ⇔ `double[] f = new double[]{0.0, 1.0};`

- int、int[]、float、double[]、bool、stringなどはC#にあらかじめ用意されている特別なクラス
- Vector2クラスやDebugクラスはUnityのクラス

クラスはC#プログラムの基本要素

https://docs.microsoft.com/ja-jp/dotnet/csharp/tour-of-csharp/classes-and-objects

- あらゆる型はクラス
  - int 型＝クラス名「int」
    - int型の変数「int a;」で、「a. 」とタイプすると…？
    - 授業外: System.Int32クラスの別名
  - int 型の配列＝「 int[] 」という名称のクラス
    - 授業外 : System.Int32[]クラスの別名

- 機能や意味単位にクラスを分け、組合わせる
  - Unity: Vector2、Debug、AudioSource、Rigidbody...

本日用いるクラスその２：Mathf

https://docs.unity3d.com/ja/current/ScriptReference/Mathf.html

- 数学クラス
  - 指数、対数、三角関数などのメソッドを提供
    - Mathf.Exp、Mathf.Log、Mathf.Sin、Mathf.Acos...
  - 一般的な定数（円周率π、無限∞など）も提供
    - Mathf.PI、Mathf.Infinity
  - インスタンスは作らない（newしない）
    - 「Mathf m = new Mathf();」はエラーではないが、特に意味をなさない

Vector2 vs Mathf

Vector2

- 2次元ベクトル一般に共通するメンバーのまとまり
  - 意味や用途が異なる2Dベクトルが複数存在する
- 個別の2Dベクトルは個別のインスタンスとして生成・操作

```
Vector2 a = new Vector2();
Vector2 b = new Vector2();
a.x += 10.0f; // キャラAのX移動
b.y += 20.0f; // キャラBのY移動
```

Mathf

- 一般的な数学関数や定数を表すメンバーのまとまり
  - バリエーションが存在しない

    Mathf a = new Mathf();

    Mathf b = new Mathf(); の2つが違う数学を表す!?
- インスタンスは生成せず、「クラス名.メンバー名」で利用

  例）Mathf.Sin(Mathf.PI / 3.0f);

用語のまとめ

- クラス
  - 複数のデータや関数をまとめた型
- クラスインスタンス
  - クラスを実体化した個別のデータ
- new
  - クラスインスタンスを生成する命令
- メンバー
  - クラスおよびインスタンスの構成要素
- フィールド
  - クラスもしくはインスタンスが持つメンバー変数
- メソッド
  - クラスもしくはインスタンスが実行できるメンバー関数

今週のまとめ

- C#プログラミング（配列、反復、メソッド）
  - 反復と関数（メソッド）の文法はC++とほぼ同様
  - C#の配列＝配列型クラスのインスタンス

- C#のクラスの理解
  - 用語：クラス、メンバー、フィールド、メソッド、クラスインスタンス、new
- Unity C#クラスの利用
  - Vector2 & Debug

Ex11：航空機運賃計算

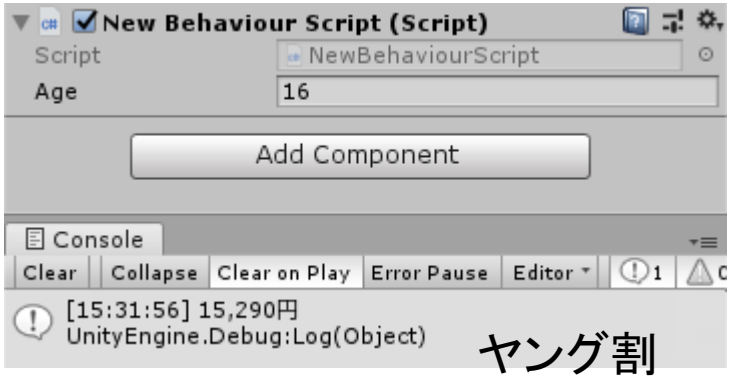
前提：　今年の5月5日朝に東京から長崎まで航空機利用で帰省します。ある運航会社の大人普通運賃は39,390 円ですが、年齢によってさまざまな割引があります。まず、満3歳以上満12歳未満の小児運賃は19,740 円です。満3歳未満の幼児は無料です。また、65歳以上は15,290 円、満12歳以上満22歳未満も15,290 円です。

課題：　入力された年齢に応じて運賃を計算するC#スクリプトを作成

実行結果例

大人普通運賃

小児普通運賃

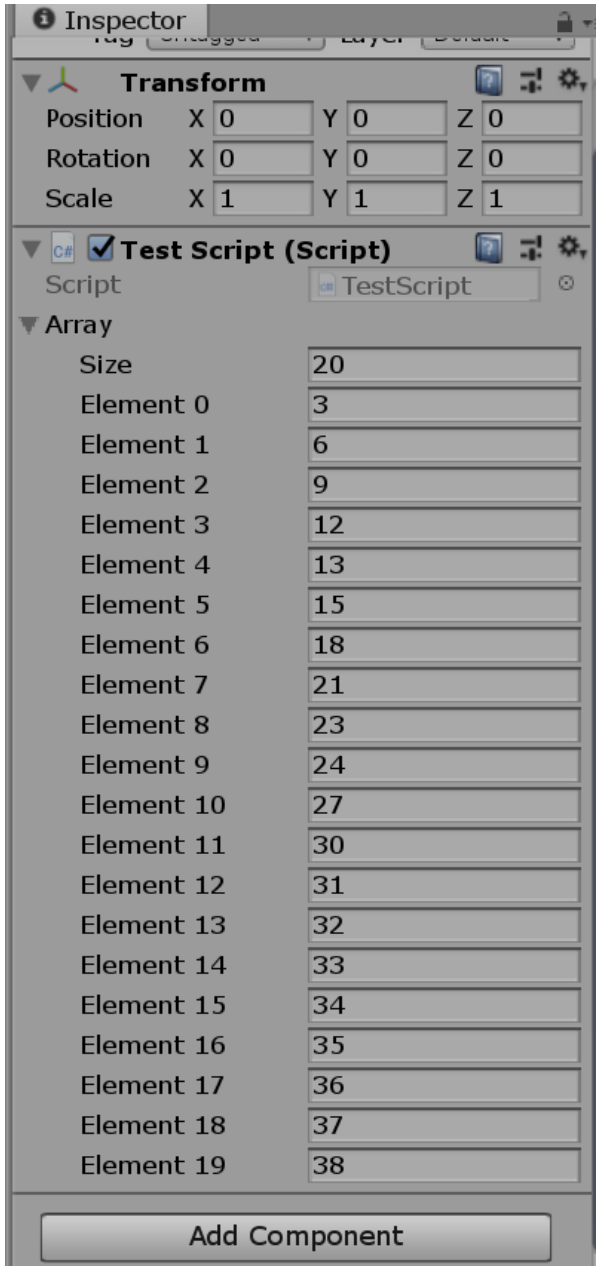


ヤング割

幼児運賃

Ex21:「3」

- 正の整数を1、2、3、…と順番に検査し、「3」を含む、あるいは「3の倍数」である数のみを配列に追加するC#スクリプトを作成する。なお、配列の長さが20に到達した時点で走査を終了すること。



実行例

Ex21のヒント

- 該当する数字が20個＝最後の数字が何かは事前にわからない　→ while文が適する

- 3で割り切れる数＝3で割ったときの余りが0

- 3を含む数
  - 一の位が3 ＝ 10で割ったときの余りが3
  - 十の位が3 についてはノーヒント

来週の予告

- 続・クラス
  - クラス継承
  - MonoBehaviourクラスの利用
  - 名前空間
  - アクセシビリティ