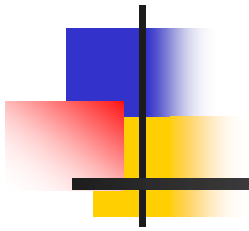
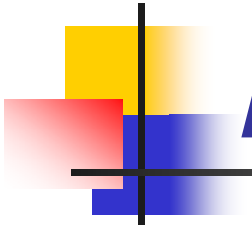


C#の基礎

株式会社ＯＳＫ
ビジネスオブジェクト技術課
小井土　亨



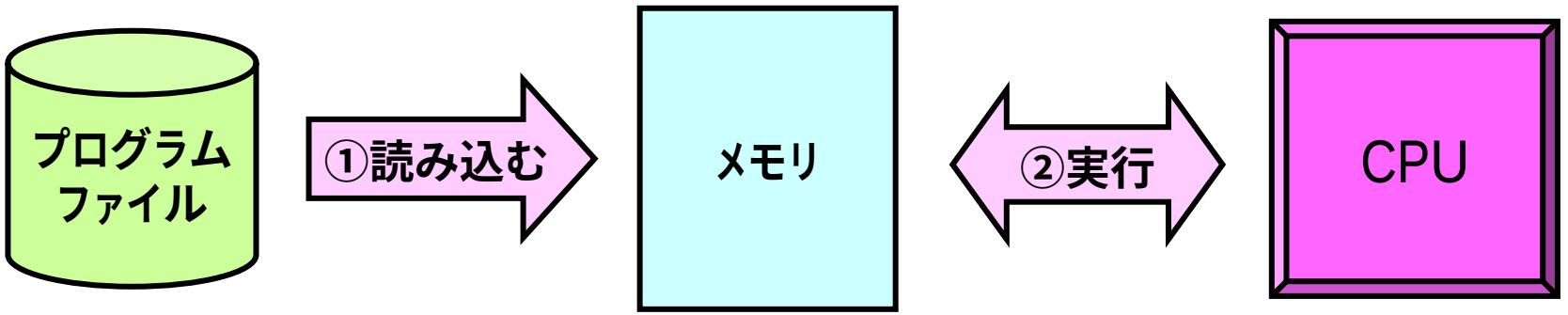
Agenda

- 復習 (基礎の基礎)
  - どのようにしてプログラムは動作するか？
  - どのようにしてプログラムを作成するか？
- C#の基礎
  - データ型、変数
  - 制御文 (if , for)
  - 演算子
  - メソッド
- プログラムを作成する
- クラスを使ったプログラミング
  - クラス、名前空間、プロパティ

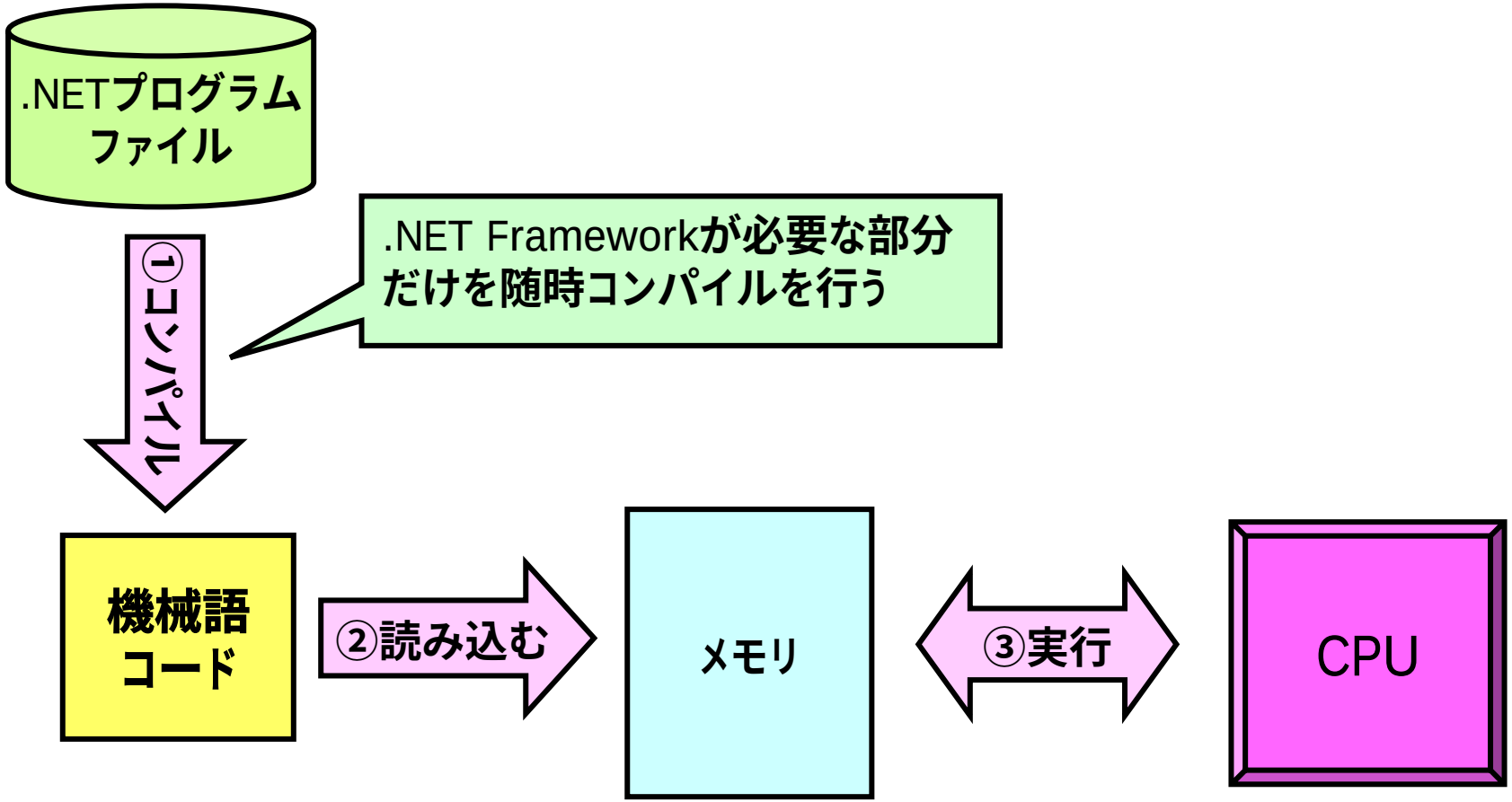
どのようにしてプログラムは動作するか？ 予備知識

- CPUとは
  - Central Processing Unit
  - プログラム命令を解釈し、実行する装置
- メモリとは
  - コンピュータ本体の中で、情報を記憶しておく場所
  - 情報には、プログラムとデータがある
- I/Oポートとは
  - CPUとデバイスの間でデータをやり取りするためのポート
- OSとは
  - Operating System
  - ファイルの管理、メモリの管理、入出力の管理、プログラムの実行管理、ユーザーインターフェースの提供などを行なう基本ソフトウェア
- プロセスとは
  - プログラムを実行するときに、OSがプログラムを管理するための単位
- ファイルとは
  - 記録ディスクに保存されたデータやプログラムの個々のまとまり

どのようにしてプログラムは動作するか？
プログラムが動作する手順

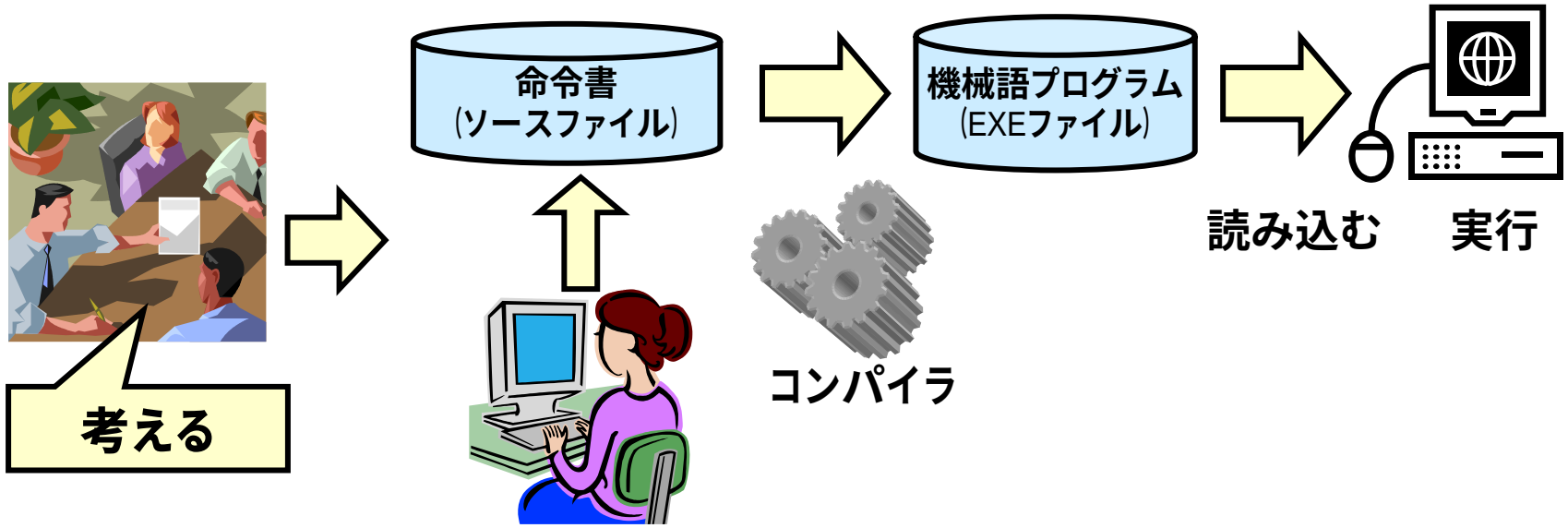


どのようにしてプログラムは動作するか？
.NETプログラムが動作する手順



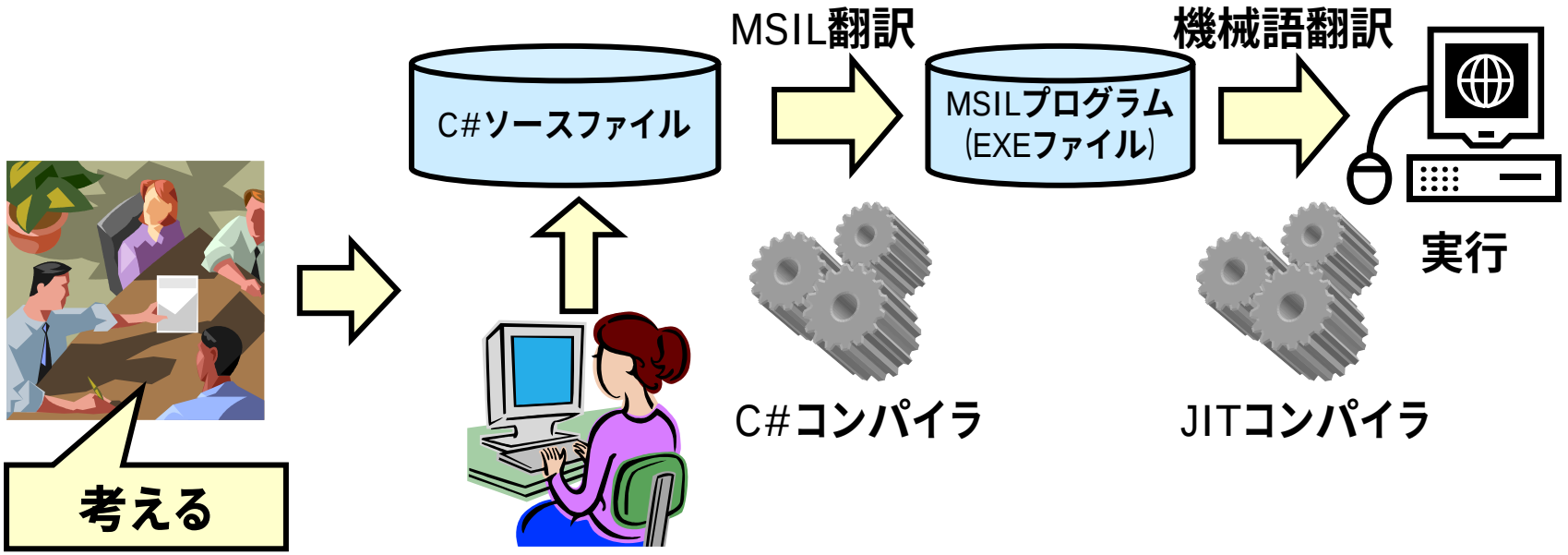
どのようにしてプログラムを作成するか？
プログラムを作成するということは

- プログラムを作成するということは
  - コンピュータをどのように動作させるか考える
  - 人間が理解できる言語で、命令書を作成する
  - 命令書をコンピュータが理解できる機械語に変換する
  - 機械語をメモリに読み込んで実行させる



どのようにしてプログラムを作成するか？ C#でプログラムを作成するということは

- **プログラムを作成するということは**
  - **コンピュータをどのように動作させるか考える**
  - **C#言語で、ソースファイルを作成する**
  - **ソースファイルを.NET Frameworkが理解できるMSILに変換する**
  - **MSILをコンピュータが理解できる機械語に変換し、メモリに読み込んで実行させる**

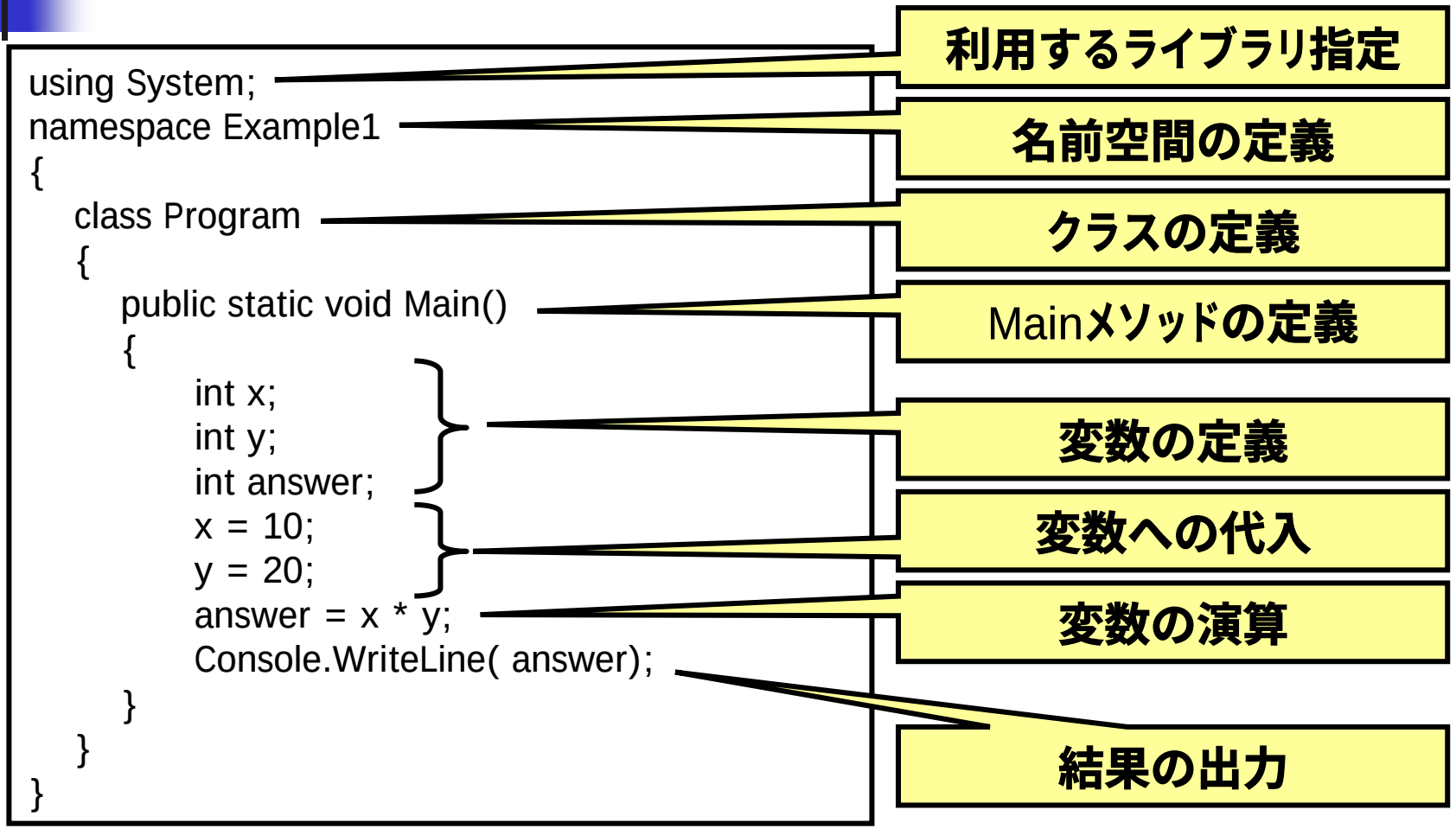


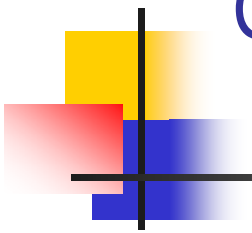
どのようにしてプログラムを作成するか？
プログラムを作成するには

- プログラムを作成するときの流れ
  - 目的をしっかり理解する
  - 目的を実現するために必要な項目を洗い出す
  - 各項目の実現方法を決定する
  - 実現方法をプログラムとして作成する
- ポイント
  - 入力と出力を明確にして、具体的なテストケースを考える
    - 例　足し算を行うプログラムを作成する場合
      - 1+2=3, 2+0=0,2+ -3 = -1, -2 + -1 = -3
  - Whatから初めてHowを決める
    - What =何をしなければならないかという視点で考える
    - How =どのような手順で実現するかという視点で考える
    - 最初からHowの視点で考えない

C#の基礎

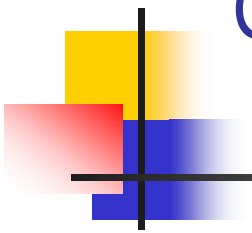
C#の基礎
簡単なプログラム





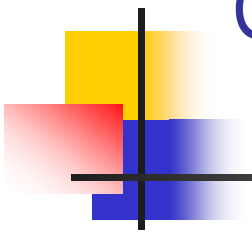
C#の基礎
名前空間とusing

- **名前空間とは**
  - **プログラム要素に対して名前付けする時の領域**
  - **名前空間を定義するとこで、自由に名前を付けることができる**
- **名前空間の定義**
  - namespace **空間名** { }
    - { }**内を名前空間とする**
- using
  - **名前空間を利用することを宣言する**
  - using **名前空間**;



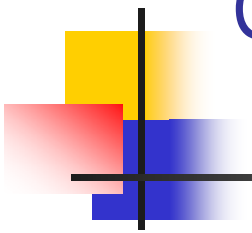
C#の基礎
Mainメソッド

- Mainメソッドとは
  - プログラムが開始される行
- 形式
  - public static void Main()



C#の基礎
文の区切り

- プログラムは文の集まりで作成する
- 文の終わり
  - セミコロン「;」で文を終わらせる
  - 改行は文の終わりではない
- 複文
  - 複数の文をブロックとしてまとめることができる
  - 中括弧「{}」で囲むことでブロックとする



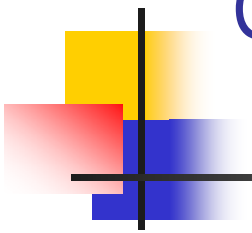
C#の基礎
変数の定義

- 変数とは
  - 値を記憶するための箱のようなもの
- 変数の宣言
  - 変数の型 変数名;
- 変数の型の種類
  - 整数　int
  - 実数　double
  - 文字列　string

C#の基礎
変数を利用した演算

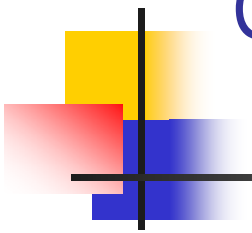
- 変数への値の代入
  - 変数名 = 値;
- 演算
  - 変数 = 変数　演算子　定数;
  - 変数 = 変数　演算子　変数;

演算子	意味
+	加算
-	減算
*	乗算
/	除算
%	余り算



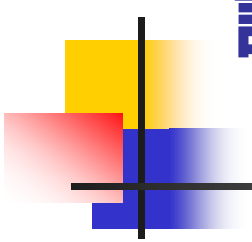
C#の基礎
入力と出力

- **出力**
  - .NET Frameworkが提供するConsoleクラスを利用する
  - **利用方法（リファレンス）**
    - Console.WriteLine(**変数**);
    - Console.WriteLine(**文字列**);
- **入力**
  - .NET Frameworkが提供するConsoleクラスを利用する
  - **利用方法（リファレンス）**
    - string**変数** = Console.ReadLine();



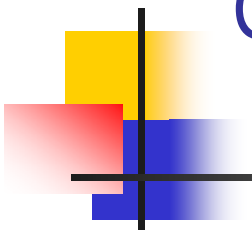
C#の基礎
コメントと字下げ

- コメントとは
  - ソースコードを読む人にプログラム内容を説明するためのもの
  - プログラムの動作とは関係なし
  - コメントは、任意の位置に記述することができる
  - 行コメント
    - 「//」移行がコメントとなる
  - ブロックコメント
    - 「/*」と「*/」で囲まれた範囲がコメントとなる
- 字下げ
  - ソースコードを見やすくするために、字下げを行う



課題
単位を変換するプログラムを作成する

- **マイル**(mi)**からキロメートル**(km)**へ変換**
  - 1 mi = 1.6093 km
- **ヤード**(yd)**からメートル**(m)**へ変換**
  - 1 yd = 0.9144 m
- **華氏** (Fahrenheit) **を摂氏** (Celsius) **に変換**
  - **変換式**　Celsius = 5/9 * (Fahrenheit – 32)
- **摂氏を華氏に変換**
- RGB**から**Gray(**グレースケール**)**へ変換**
  - Gray = R * 0.3 + G * 0.59 + B * 0.11

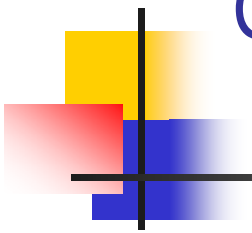


C#の基礎
複合代入演算子

- **複合代入演算子とは**
  - **代入文を容易にするために用意された演算子**
- **形式**
  - **変数 演算子= 式;**
- **例**
  - x = x + 100;　　x += 100;
  - y = y + z;　　y += z;
  - w = w + n - m;　　w += n - m;
  - z = z - 100;　　z -= 100;

C#の基礎
インクリメント演算子、デクリメント演算子

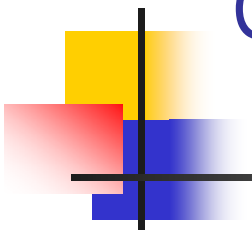
- インクリメント演算子、デクリメント演算子とは
  - 代入文を容易にするために用意された演算子
- インクリメント演算子
  - ++変数 あるいは 変数++
- デクリメント演算子
  - --変数 あるいは 変数--
- 例
  - x = x + 1;　　++x;
  - y = y - 1;　　--y;



C#の基礎
真偽値(bool値)

- **真偽値とは**
  - **真**(true)**と偽**(false)**のいずれかを表す**
  - bool**型**
- bool**変数**
  - **真偽値を扱うための変数**
- **例**

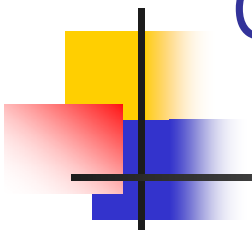
```
bool answer;

answer = 10 < 5;

Console.WriteLine( answer );
```



C#の基礎
関係演算子

- 関係演算子とは
  - 値同士の関係を表して、結果は真偽値となる
- 演算子

演算子	意味
==	等しい
!=	等しくない
>	より大きい
<	より小さい
>=	以上
<=	以下



C#の基礎
論理演算子

- 論理演算子とは
  - 真偽値同士の結びつき方を表す
- よく利用する演算子

演算子	意味
||	論理和
&&	論理積
!	否定

C#の基礎
制御文　選択文

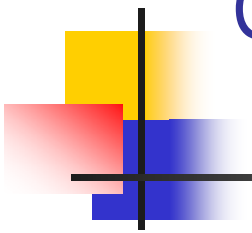
- if文
  - 処理を条件によって分岐したい場合に使用する
  - 条件には、関係演算子と論理演算子を組み合わせて利用する
- 形式
  - if(条件)文;
  - if(条件)文; else 文;
- 例

```
if( age < 20 )
{
    Console.WriteLine( "未成年" );
}
else
{
    Console.WriteLine( "成人" );
}
```

C#の基礎
制御文　繰り返し文

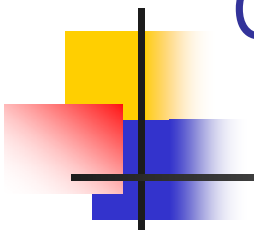
- for文
  - 指定した条件が成り立つ間、処理を繰り返す
- 形式
  - for(初期化;条件判定;インクリメント)文;
- 例

```
for( i = 1; i <= count; ++i )
{
    total += i;
}
```



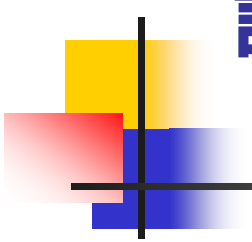
C#の基礎
配列

- 配列とは
  - 同じ名前を使って参照することができる、変数の集まり
  - 複数の次元を持つこともできる
- 配列の参照方法
  - 配列名に[]を付け、[]内に添え字を指定する
  - 添え字は、定数でも変数でも良い
  - 添え字は、0から始まり、サイズから1を引いた
- 一次元配列
  - 一般的な形式
    - 型[] 配列名 = new 型[サイズ];
  - 例 10個のint型配列
    - int[] sample = new int[10]
  - 配列のサイズ
    - 配列名.Length



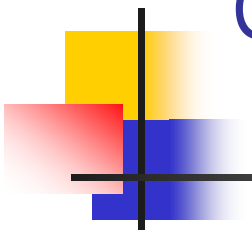
C#基礎
型変換

- キャスト
  - キャストとは
    - 型の変換を行う
  - 形式
    - (変換後の型)式
  - 例　y = (int)d; // dはdouble、yはint
- ToStringメソッド
  - 変数を文字列に変換する
  - 例　s = d.ToString()
- Parseメソッド
  - 文字列を数値型に変換する
  - 形式
    - 型.Parse(文字列)
  - 例 y = int.Parse(str);



課題
平均値を求める

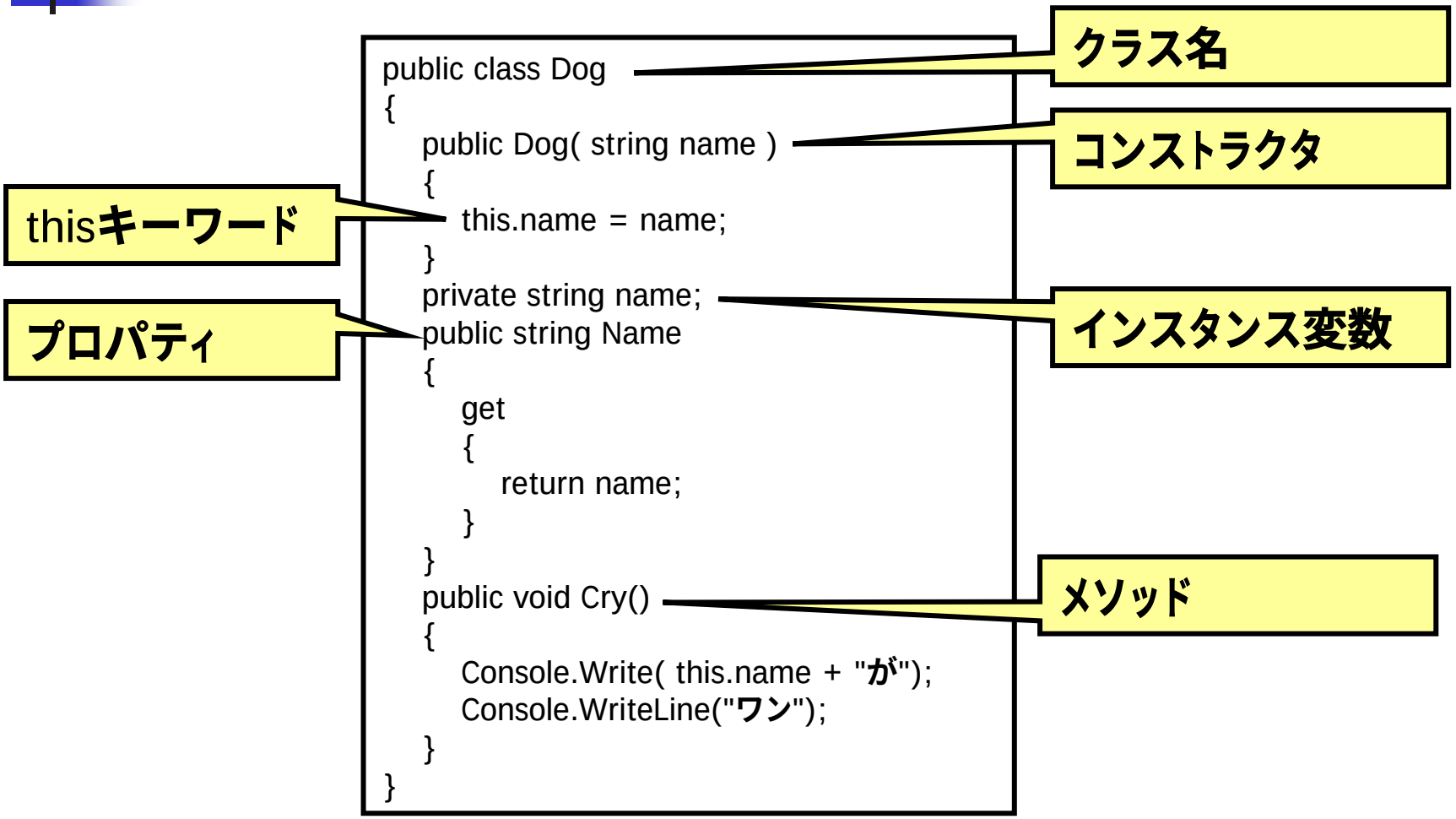
- Double型の10個の配列を定義し、平均値(Average)を求めるプログラムを作成する
- 最大値と最小値を出力する
- 値を入力できるようにする
- 配列の数を入力できるようにする
- 0以下の値は平均から除外するようにする

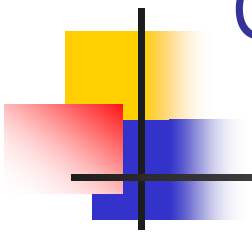


C#の基礎
クラスとインスタンス

- クラスとは
  - 分類、種類
  - 同種のものを集めて定義したもの
- インスタンスとは
  - クラスを具体的にしたもの
- 例
  - 犬クラス
    - ポチやタローはインスタンス
  - 社員クラス
    - 上田 雅史、中村 孝幸 はインスタンス

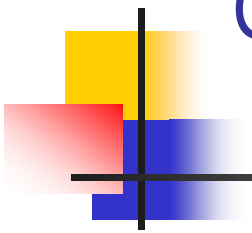
C#の基礎
クラスの一般形式





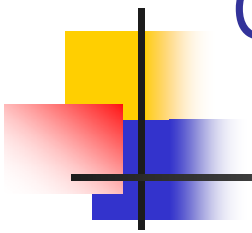
C#の基礎
アクセス修飾子

- アクセス修飾子とは
  - クラスやメンバーなどに、アクセスできる範囲を定めるためのもの
- アクセス修飾子の種類
  - public
    - すべてのメンバーからアクセスすることができる
  - private
    - クラスの中からのみアクセスすることができる



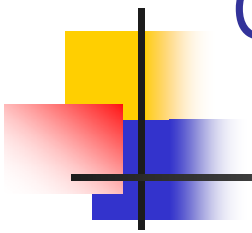
C#の基礎
クラスの定義

- クラス定義の一般形式
  - アクセス修飾子　class クラス名　{ }
- インスタンス変数の定義
  - アクセス修飾子　データ型　変数名;
  - 一般にアクセス修飾子は、privateとする
  - クラス内では、thisキーワードを使用して参照する
    - 例 this.name
- 例
  - private string name;



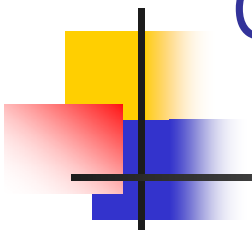
C#の基礎
メソッドの定義

- メソッドとは
  - クラスが提供する機能
- メソッド定義の一般形式
  - アクセス修飾子　戻り値の型 メソッド名(パラメータリスト){}
- 戻り値とは
  - メソッドから処理結果を返す値のこと
  - メソッド定義では、戻り値の型を定義する
  - void型
    - 戻り値を返さないことを示す型
    - 戻り値を返さないときに利用する
- パラメータ
  - メソッドを呼び出すときに渡すデータ



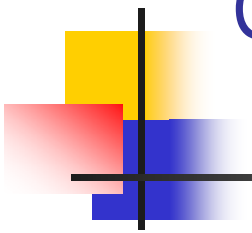
C#の基礎
コンストラクタ

- コンストラクタとは
  - クラスのインスタンスを生成するための特殊なメソッド
  - パラメータが異なれば、複数用意することができる
  - 戻り値の定義はしない
- メソッド定義の一般形式
  - アクセス修飾子　クラス名(パラメータリスト){}
  - アクセス修飾子は、基本的にはpublic



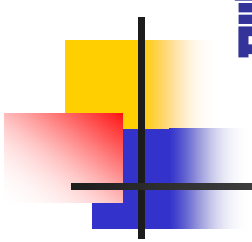
C#の基礎
プロパティの定義

- プロパティとは
  - クラスが提供する情報や属性
  - 一般的には、設定と参照を行うことができる
- プロパティ定義の一般形式
  - アクセス修飾子　戻り値の型 プロパティ名{ get{} set{} }
  - アクセス修飾子は基本的にpublic
- 参照だけのプロパティ
  - getだけを用意することで、参照だけのプロパティを用意することができる



C#の基礎
インスタンスの利用

- インスタンス変数の定義
  - インスタンスを保存する変数を用意する
  - 一般形式
    - クラス名　インスタンス変数名;
- インスタンスの生成
  - クラスからnew演算子を使用して、インスタンスを生成することができる
  - インスタンス変数に代入する
- メソッドの呼び出し
  - インスタンス変数.メソッド名で、呼び出しを行う
- プロパティの参照、設定
  - インスタンス変数.プロパティ名で、参照と設定を行う



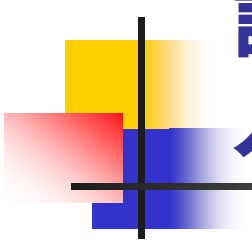
課題1
試験(Examination)クラスを作成する

- 名前プロパティを作成する
- 名前を設定できる、コンストラクタを作成する
- 点(Score)プロパティを作成する
- 試験にパスしたか回答するメソッドを作成する
  - メソッド名は、IsPass()とする
  - メソッドの戻り値は論理値(bool)とする
  - 試験にパスするためには、60点以上とる必要がある

課題　その2
レジ(Register)クラスを作成する

- 合計(Sum)プロパティを作成する
- お金の入れる(PutMoney)メソッドを作成する
- お金の取り出し(TakeMoney)メソッドを作成する
- 各貨幣のプロパティを作成する
  - このとき、入金と出金は、最適(最小)の枚数になると仮定する
    - 千札(ThousandNote)
    - 百円(HundredCoin)
- レジクラスの作成時に、各貨幣の値を設定するコンストラクタを作成する

注意
処理をシンプルにするために、貨幣は千円と百円だけとし、扱う金額も百円以下はなしとする



課題　その3
人(Person)クラスを作成する

- Personクラスの作成時に、誕生日を設定するコンストラクタを作成する
- 誕生日プロパティを作成する
- 年齢プロパティを作成する
- 指定した日にちのバイオグラフを計算するメソッドを作成する
  - 計算式
    - 身体のリズム = SIN(日数/23*PI*2)
    - 感情のリズム = SIN(日数/28*PI*2)
    - 知性のリズム = SIN(日数/33*PI*2)