明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

プログラミング演習2
クラス

中村，高橋，小林，橋本

先週のまとめ

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- オブジェクト指向のさわりを学んだ
  - 色んなクラス
    - ArrayList, PImage, Minim, AudioPlayer, String
  - インスタンス化
    - Human nkmr = new Human();
  - インスタンス
    - nkmr
  - インスタンス変数
    - nkmr.speed
  - インスタンスメソッド
    - nkmr.move()
  - コンストラクタ
    - Human(){ 初期化処理 }

本日の流れ

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- 13:30-14:00 宿題の解説
- 14:00-15:10 座学＋演習（10分休憩）
- 15:20-15:30 課題提示
- 15:30-16:30 課題に取り組む
- 16:30が課題の提出期限
- 16:30-17:00 課題の解説

はじめに

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- ちょっとEP演習の発表会がすごかったのでいきなり進みすぎましたね。申し訳ない。

- 中間試験は今年度はやりません。その代わりにクラスを使った課題作成に挑戦してもらいます。

これからやること

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- こんな感じの中身（どんな感じで管理しているか）はよくわからないけど，メソッドを使うことで手軽に使える仕組みを実現していく！
- 「クラス」という知識を習得する
  - プログラムがよりわかりやすくなる
  - カプセル化などにより問題の切り分けができる
  - 使い回しができる！
  - 他人とプログラムを共有するときに，内部を気にせずに利用することができる

  などなどのメリット

例：動物園を実現する

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- 猫，犬，猿，象，熊がいる動物園を再現する
  - 表示場所に関する情報

    猫(catX, catY), 犬(dogX, dogY), 猿(monkeyX, monkeyY),
    象(elephantX, elephantY), 熊(bearX, bearY)
  - それぞれの動物の描画

    drawCat(), drawDog(), drawMonkey(), drawElephant(), drawBear();
  - それぞれの動物の移動

    moveCat(), moveDog(), moveMonkey(), moveElephant(), moveBear();
  - それぞれの動物の睡眠

    sleepCat(), sleepDog(), sleepMonkey(), sleepElephant(),
    sleepBear();
  - などを用意し，それぞれをプログラムの内部から呼び出す必要あり

かなり面倒で混乱のもと

動きは動物に任せたい

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- 猫，犬，猿，象，熊を定義
  - それぞれの座標は意識したくない
    - cat.x, cat.y, dog.x, dog.y, monkey.x, monkey.y, ...
    - 内部で適当に処理してもらう
  - 描画はシンプルにしたい
    - cat.draw(), dog.draw(), monkey.draw(), elephant.draw(), ...
  - 移動もシンプルにしたい
    - cat.move(), dog.move(), monkey.move(), elephant.move(), ...
  - 睡眠も任せてしまう
    - cat.sleep(), dog.sleep(), monkey.sleep(), elephant.sleep(), ...

すべてを XXXX . 機能 という形に！

オブジェクト指向（クラス）

オブジェクト指向とは

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- ざっくり説明すると，色々な値や機能をもつもの

  （例） シューティングゲーム上の敵
  - 現在位置（X座標，Y座標）
  - 何らかの移動機能（移動の関数）
  - 何らかの描画機能（描画の関数）

  をもっており，プログラムから移動しろ，描画しろと命令を送るだけで，その中身（変数の状態）や処理がどうなっているかを意識せずに利用可能

- 他人が何をどう考え実行するかを気にせず，「～をやっておいて」とお願いする感覚

たとえば

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- 人間というクラスを定義する
  - 人間には名前という変数
  - 現在地という場所に関する変数
  - 年齢という変数

  などがある
- 人間には下記のメソッドがある
  - 移動する
  - 食べる
  - 喋る
  - 聞く

  など

変数を直接触らない

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- メソッドだけで色々と制御できるように！
- 時計を実現することを考える
  - なかの制御系を直接動かさない
  - 時・分・秒を変更するために、ダイヤルを引っ張り出し、ダイヤルを回す
  - ボタンを押すことでアラームのセットなど

クラスの定義

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- Ball クラスを定義

  class クラス名 {

  　クラスの諸要素に関する定義

  }

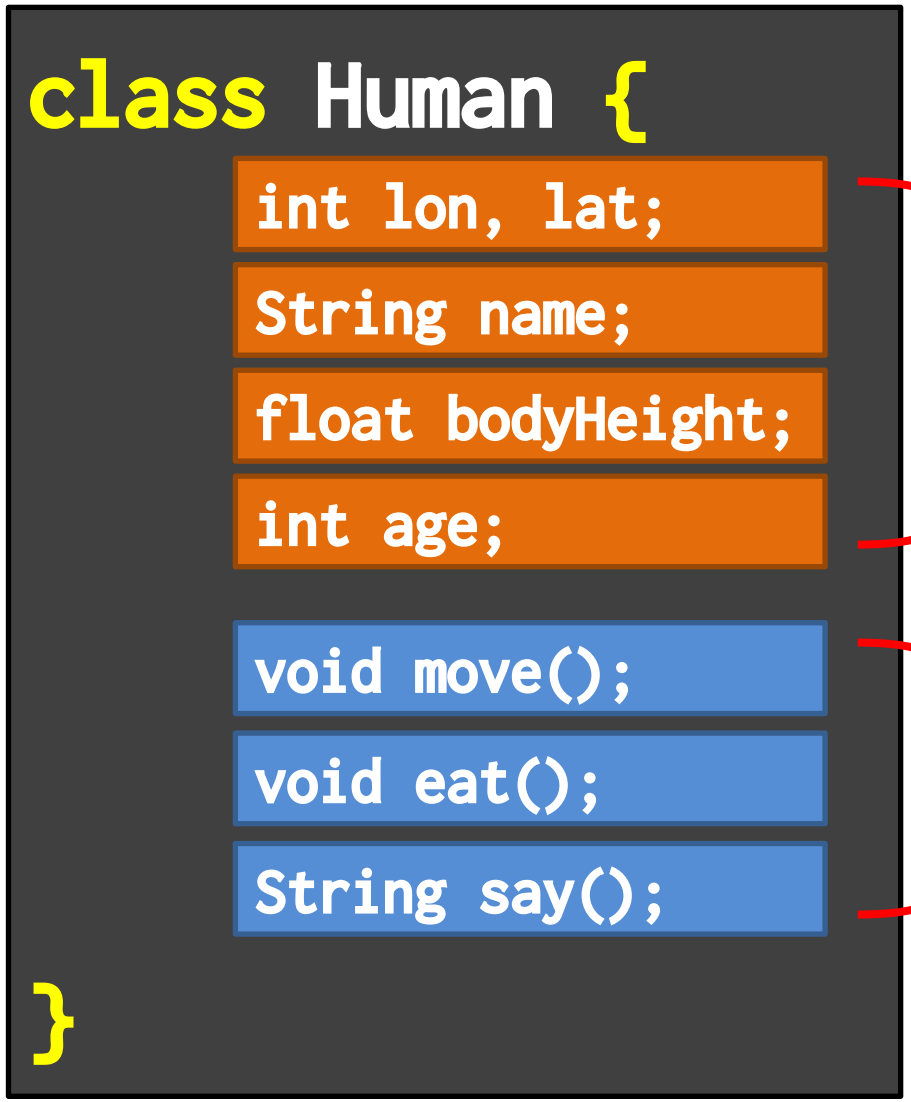
```
class Ball {

}
```

```
class Human {

}
```

```
class Animal {

}
```

クラスの定義

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室



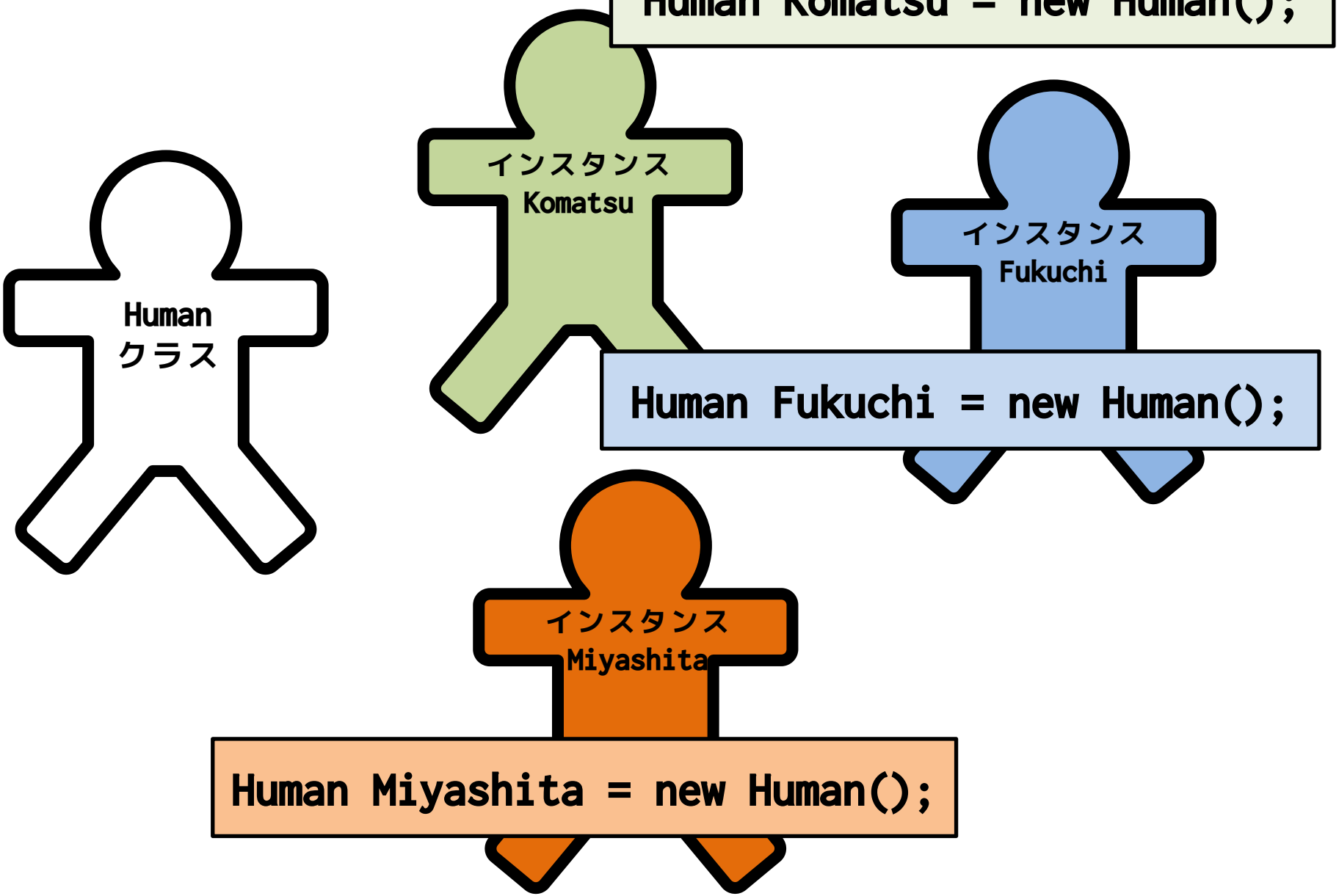
インスタンス変数

インスタンスメソッド

インスタンス化

明治大学総合数理学部
先端メディアサイエンス学科

```
Human Komatsu = new Human();
```



定義したクラスの使い方

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- クラスの中で変数を定義すると，そのクラス内の変数として使うことができる
  - クラス内で変数を定義
  - クラスを使う場合は new！
  - クラス内の変数を使う場合はドットでつなぐ

```
class Ball
{
  float posX;
  float posY;
  float speedX;
  float speedY;
}

Ball ball;
ball = new Ball();
ball.posX += ball.speedX;
ball.posY += ball.speedY;
```

インスタンス化

クラス名　変数名 = new クラス名();

他の例

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

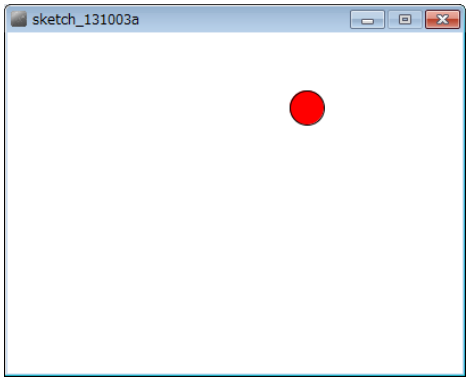
- 時計というクラスを作る
  - 時計のインスンタンス変数
    - 現在時間（時分秒）の情報
    - 目覚ましのON/OFF状態管理変数
    - 目覚ましの設定時間
  - 時計のインスタンスメソッド
    - 分変更メソッド
    - 時計停止メソッド（秒針停止）
    - 目覚まし設定時間変更メソッド
    - 目覚ましのON/OFF切り替えメソッド

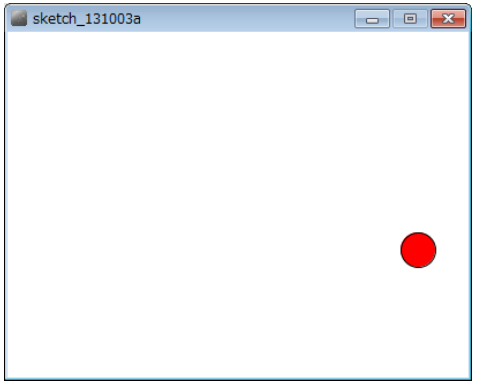
端で跳ね返るオブジェクト

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

(Q1) 400x300のウインドウ内で，画面中央から毎フレームx方向に2ピクセル，y方向に3ピクセルずつ移動する直径が30の赤い円が右端・左端・上端・下端に来ると跳ね返るようにするには？

- 考え方
  - 右端・左端・上端・下端で衝突する時の条件を整理
  - 衝突した時の速度を反転させる
    - `speedX = -speedX;`
    - `speedY = -speedY;`





```
float posX;
float posY;
float speedX;
float speedY;

void setup()
{
  size(400, 300);
  posX = random(0, width);
  posY = random(0, height);
  speedX = random(-5, 5);
  speedY = random(-5, 5);
  fill(255, 0, 0);
}
```

今までの知識で
プログラムを組むと

```
void draw()
{
  background(255);
  posX += speedX;
  posY += speedY;
  if(posX > width){
    posX = width * 2 - posX;
    speedX = -speedX;
  }
  if(posX < 0){
    posX = -posX;
    speedX = -speedX;
  }
  if(posY > height){
    posY = height * 2 - posY;
    speedY = -speedY;
  }
  if(posY < 0){
    posY = -posY;
    speedY = -speedY;
  }
  ellipse(posX, posY, 30, 30);
}
```

```
float posX;
float posY;
float speedX;
float speedY;

void setup()
{
  size(400, 300);
  posX = random(0, width);
  posY = random(0, height);
  speedX = random(-5, 5);
  speedY = random(-5, 5);
  fill(255, 0, 0);
}
```

変数を定義

場所と速度を設定

機能ごとに
分解すると

```
void draw()
{
  background(255);
  posX += speedX;
  posY += speedY;
  if(posX > width){
    posX = width * 2 - posX;
    speedX = -speedX;
  }
  if(posX < 0){
    posX = -posX;
    speedX = -speedX;
  }
  if(posY > height){
    posY = height * 2 - posY;
    speedY = -speedY;
  }
  if(posY < 0){
    posY = -posY;
    speedY = -speedY;
  }
  ellipse(posX, posY, 30, 30);
}
```

移動

描画

```
float posX;
float posY;
float speedX;
float speedY;

void setup()
{
  size(400, 300);
  posX = random(0, width);
  posY = random(0, height);
  speedX = random(-5, 5);
  speedY = random(-5, 5);
  fill(255, 0, 0);
}
```

変数を定義
クラス内に

場所と速度を設定
.initialize()

機能ごとに
分解すると

```
void draw()
{
  background(255);
  posX += speedX;
  posY += speedY;
  if(posX > width){
    posX = width * 2 - posX;
    speedX = -speedX;
  }
  if(posX < 0){
    posX = -posX;
    speedX = -speedX;
  }
  if(posY > height){
    posY = height * 2 - posY;
    speedY = -speedY;
  }
  if(posY < 0){
    posY = -posY;
    speedY = -speedY;
  }
  ellipse(posX, posY, 30, 30);
}
```

移動：.move()

描画：.display()

クラスを作るときは別タブで

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

Ballクラス

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

```
class Ball
{
  float posX;
  float posY;
  float speedX;
  float speedY;

  void initialize()
  {
    posX = random(width);
    posY = random(height);
    speedX = random(-5, 5);
    speedY = random(-5, 5);
  }

  void display(){
    fill(255, 0, 0);
    ellipse( posX, posY, 30, 30 );
  }
```

変数を定義
クラス内に

場所と速度を設定
.initialize()

描画: .display()

移動: .move()

```
  void move(){
    posX += speedX;
    posY += speedY;
    if(posX > width){
      posX = width * 2 - posX;
      speedX = -speedX;
    }
    if(posX < 0){
      posX = -posX;
      speedX = -speedX;
    }
    if(posY > height){
      posY = height * 2 - posY;
      speedY = -speedY;
    }
    if(posY < 0){
      posY = -posY;
      speedY = -speedY;
    }
  }
}
```

Ballクラス

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

```
class Ball
{
  float posX;
  float posY;
  float speedX;
  float speedY;

  void initialize()
  {
    posX = random(width);
    posY = random(height);
    speedX = random(-5, 5);
    speedY = random(-5, 5);
  }

  void display(){
    fill(255, 0, 0);
    ellipse( posX, posY, 30, 30 );
  }
```

変数を定義
クラス内に

場所と速度を設定
.initialize()

描画: .display()

移動: .move()

```
  void move(){
    posX += speedX;
    posY += speedY;
    if(posX > width){
      posX = width * 2 - posX;
      speedX = -speedX;
    }
    if(posX < 0){
      posX = -posX;
      speedX = -speedX;
    }
    if(posY > height){
      posY = height * 2 - posY;
      speedY = -speedY;
    }
    if(posY < 0){
      posY = -posY;
      speedY = -speedY;
    }
  }
}
```

```
Ball ball = new Ball();

void setup()
{
  size(400, 300);

  ball.initialize();

  fill(255, 0, 0);
}

void draw()
{
  background(255);

  ball.move();

  ball.display();
}
```

変数を定義
インスタンス化

場所と速度を設定
.initialize()

移動: .move()

描画: .display()

プログラムを動かそう！

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- http://nkmr.io/lecture/ の第2回講義資料にある Ball.txt を利用しよう
  - Ballクラスの部分は別のタブに！（次ページで説明）

変数を追加する

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

Q. 起動するたびに半径を10〜50の間でランダムに設定

- 整数の radius という変数を定義
- initialize() で初期化する
- display() で radius を使って描画する
  - ellipse では直径を利用するため、radius*2 を利用

Ballクラス

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

```
class Ball
{
  float posX;
  float posY;
  float speedX;
  float speedY;
  int radius;

  void initialize()
  {
    posX = random(width);
    posY = random(height);
    speedX = random(-5, 5);
    speedY = random(-5, 5);
    radius = (int)random(10, 50);
  }

  void display(){
    fill(255, 0, 0);
    ellipse(posX, posY, radius*2, radius*2);
  }
```

```
  void move(){
    posX += speedX;
    posY += speedY;
    if(posX > width){
      posX = width * 2 - posX;
      speedX = -speedX;
    }
    if(posX < 0){
      posX = -posX;
      speedX = -speedX;
    }
    if(posY > height){
      posY = height * 2 - posY;
      speedY = -speedY;
    }
    if(posY < 0){
      posY = -posY;
      speedY = -speedY;
    }
  }
}
```

機能を追加する

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

Q. 円内でクリックすると、円の半径を半分にしていく

- クリックされた場所 (mouseX, mouseY) が円内だったら直径 radius を小さくする click() というメソッドを用意

- click メソッドの引数として mx, my を設定し中心座標(posX, posY)と半径 radius の関係で円内にあるかどうかを判定
  - 円内 → 半分
  - 円外 → なにもしない

機能を追加する

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

Q. 円内でクリックすると
円の半径を半分にする

```
class Ball
{
  float posX;
  float posY;
  float speedX;
  float speedY;
  int radius;

  void click(int mx, int my)
  {
    // 中心座標からクリック場所までの距離が
    // 半径以下であれば円内をクリック
    if(dist(posX, posY, mx, my) <= radius){
      radius = radius / 2;
    }
  }
```

```
Ball ball = new Ball();

void setup()
{
  size(400, 300);
  ball.initialize();
  fill(255, 0, 0);
}

void draw()
{
  background(255);
  ball.move();
  ball.display();
}

void mousePressed()
{
  ball.click(mouseX, mouseY);
}
```

コンストラクタ！

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- コンストラクタは new されたときに呼び出される場所．initialize() はそこで呼び出したら良いのでは？

```
class Ball
{
  float posX;
  float posY;
  float speedX;
  float speedY;
  Ball(){

  }
}
```

コンストラクタ！

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

```
class Ball{
  float posX;
  float posY;
  float speedX;
  float speedY;
  int radius;

  Ball(){
    initialize();
  }

  void initialize(){
    posX = random(width);
    posY = random(height);
    speedX = random(-5, 5);
    speedY = random(-5, 5);
    radius = (int)random(10, 50);
  }
```

コンストラクタで initizalize
メソッドを呼び出す！

```
Ball ball = new Ball();

void setup()
{
  size(400, 300);
  ball.initialize();
  fill(255, 0, 0);
}

void draw()
{
  background(255);
  ball.move();
  ball.display();
}

void mousePressed()
{
  ball.click(mouseX, mouseY);
}
```

課題2-2: basic_boundAll2

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

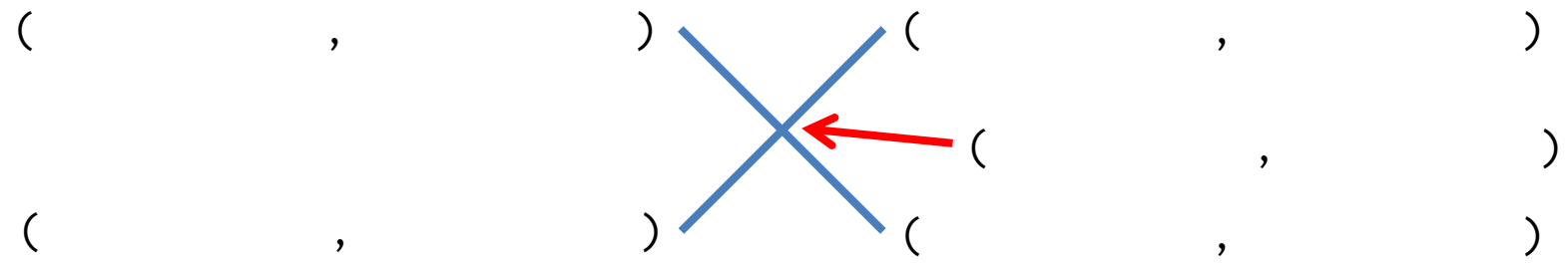
- Ball クラスを改良し，xが動き回るCrossクラスと△が動き回るTriangleクラスを作成せよ
- またこれを利用して5個の〇と，4個のxと，3個の△が画面内を動き回るプログラムを作成せよ
  - ただし，その速度はx、y方向それぞれ-5～5の実数値とせよ
  - また，〇は壁で跳ね返り，xと△は跳ね返らずに反対側から出てくるようにせよ

xが動き回るクラスを作ろう

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

Ballクラスを利用して、Crossクラスを作ろう！

- Ballクラスと、Crossクラスの違いは、表示される図形が「○」か「x」かなだけ！！
- Crossクラスのタブを作成し、Ballクラスをコピペ！
- BallをCrossに書き換える！
- 表示だけを変更したいので、displayの中身を変更！
- メインのプログラムで Cross を使っていこう！



Ballを改良しCrossを作る

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

```
class Cross{
  float posX;
  float posY;
  float speedX;
  float speedY;

  Cross(){
    posX = random(width);
    posY = random(height);
    speedX = random(-5, 5);
    speedY = random(-5, 5);
  }

  void display(){
    line(x-15, y-15, x+15, y+15);
    line(x+15, y-15, x-15, y+15);
  }
```

```
  void move(){
    posX += speedX;
    posY += speedY;
    if(posX > width){
      posX = width * 2 - posX;
      speedX = -speedX;
    }
    if(posX < 0){
      posX = -posX;
      speedX = -speedX;
    }
    if(posY > height){
      posY = height * 2 - posY;
      speedY = -speedY;
    }
    if(posY < 0){
      posY = -posY;
      speedY = -speedY;
    }
  }
}
```

Crossクラスを使うと

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

```
Cross cross1;
Cross cross2;
Cross cross3;
void setup() {
  size( 400, 300 );
  cross1 = new Cross();
  cross2 = new Cross();
  cross3 = new Cross();
}

void draw() {
  background(255);
  cross1.move();
  cross2.move();
  cross3.move();
  cross1.display();
  cross2.display();
  cross3.display();
}
```

端で跳ね返る50個のボール

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

400x300の画面内にX座標Y座標ともにランダムに配置された50個の円を描き，それぞれのX，Y方向へのスピードに応じて移動するようにせよ

- posX，posY という座標の配列と，speedX，speedY というXおよびY方向の速度をもつ配列を導入！

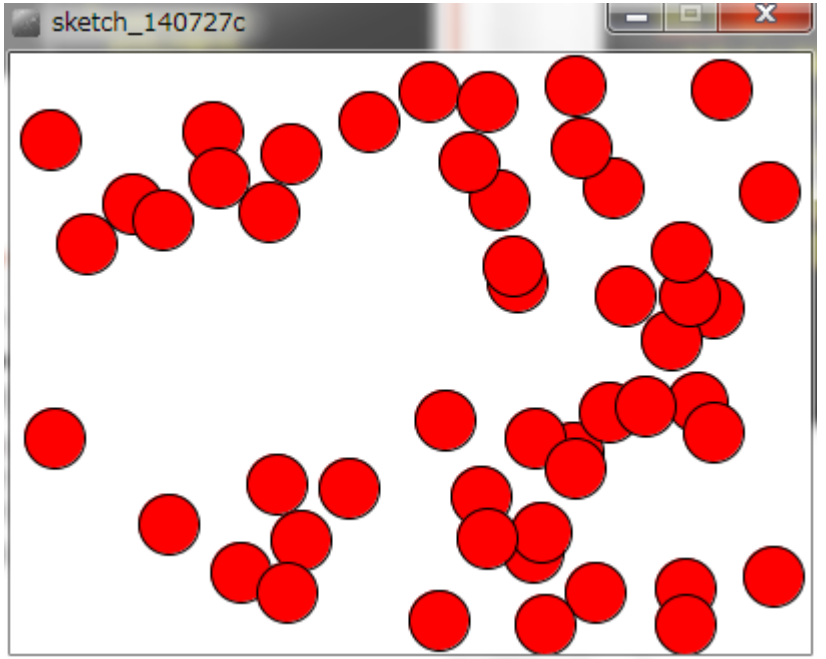
オブジェクト＋配列

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- 50個の丸を動かすには配列を使う！

```
Ball[] balls = new Ball[50];

void setup(){
  size(400, 300);
  for(int i=0; i<50; i++){
    balls[i] = new Ball();
  }
}

void draw(){
  background(255);
  for(int i=0; i<50; i++){
    balls[i].move();
    balls[i].display();
  }
}
```

型 [] 配列名 = new 型 [要素数];



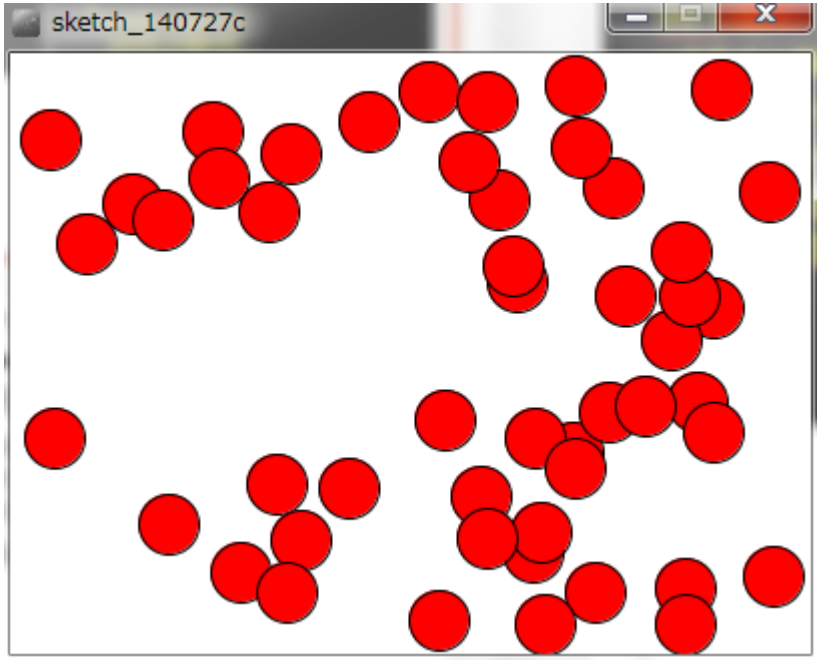
オブジェクト＋配列

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- 50個の丸を動かすには配列を使う！

```
Ball[] balls = new Ball[50];

void setup() {
  size( 400, 300 );
  for(int i=0; i<balls.length; i++){
    balls[i] = new Ball();
  }
}

void draw(){
  background(255);
  for(int i=0; i<balls.length; i++){
    balls[i].move();
    balls[i].display();
  }
}
```

配列変数名.length
で配列の長さを取得



オブジェクト＋配列

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- 300個の丸を動かすには配列の定義を変更

```
Ball[] balls = new Ball[300];

void setup() {
  size(400, 300);
  for(int i=0; i<balls.length; i++){
    balls[i] = new Ball();
  }
}

void draw(){
  background(255);
  for(int i=0; i<balls.length; i++){
    balls[i].move();
    balls[i].display();
  }
}
```

配列変数名.length
で配列の長さを取得

ArrayListと組み合わせ

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- クリックするたびにBallがうまれる！

```
ArrayList<Ball> balls = new ArrayList<Ball>();

void setup() {
  size(400, 300);
}

void draw(){
  background(255);
  for(int i=0; i<balls.size(); i++){
    balls.get(i).move();
    balls.get(i).display();
  }
}

void mousePressed(){
  Ball ball = new Ball();
  balls.add( ball );
}
```

ArrayListと組み合わせ

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- クリックするたびにBallがうまれる！

```
ArrayList<Ball> balls = new ArrayList<Ball>();

void setup() {
  size(400, 300);
}

void draw(){
  background(255);
  for(int i=0; i<balls.size(); i++){
    balls.get(i).move();
    balls.get(i).display();
  }
}

void mousePressed(){
  balls.add(new Ball());
}
```

こう書いてもOK

ArrayListと組み合わせ

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- クリックするたびにBallがうまれる！

```
ArrayList balls = new ArrayList();

void setup() {
  size(400, 300);
}

void draw(){
  background(255);
  for(int i=0; i<balls.size(); i++){
    Ball ball = (Ball)balls.get(i);
    ball.move();
    ball.display();
  }
}

void mousePressed(){
  balls.add(new Ball());
}
```

定義をこうする場合は

キャストして使う！

宿題2-1：hw_listBall

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- ウインドウ1000x1000を用意し，Ballクラスを利用してマウスクリックするたびにランダムな場所に丸が生まれ，上下左右にランダムな速度で動き，上下左右の端で跳ね返るプログラムを作成せよ．なお，クリックでいくらでも作成できるようにせよ
- また，Ballクラスを改良してそれぞれの丸の色は初期値としてランダムに設定するようにせよ

  ［ヒント］色情報を持つインスタンス変数を追加する

  – int red; (0 - 255)

クリックした場所に登場させたい！

- たくさん増やして管理するには
  ArrayListを使う！

コンストラクタで定義

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- インスタンスを作るときに初期位置を
  マウスの位置に！

**コンストラクタ内で
mouseX，mouseYを使う？**

```
class Ball{
  float posX;
  float posY;
  float speedX;
  float speedY;

  Ball(){
    posX = mouseX;
    posY = mouseY;
    speedX = random(1,5);
    speedY = random(1,5);
  }
```

- これだと色々使い回せないので困る！！！
- 普通に作ったときにおかしくなる

コンストラクタで定義！

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- インスタンスを作るときに初期位置を明示的に決めたい！

  new Ball(mouseX, mouseY)

  と指定したい！

コンストラクタとして座標を指定できるものを用意する！

```
class Ball{
  float posX;
  float posY;
  float speedX;
  float speedY;

  Ball(){
    posX = random(width);
    posY = random(height);
    speedX = random(1,5);
    speedY = random(1,5);
  }

  Ball(float x, float y){
    posX = x;
    posY = y;
    speedX = random(1,5);
    speedY = random(1,5);
  }
```

new Ball(mouseX, mouseY)

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- クリックするたびにその場所からBallがうまれる！

```
ArrayList<Ball> balls = new ArrayList<Ball>();

void setup() {
  size(400, 300);
}

void draw(){
  background(255);
  for(int i=0; i<balls.size(); i++){
    balls.get(i).move();
    balls.get(i).display();
  }
}

void mousePressed(){
  Ball ball = new Ball(mouesX, mouseY);
  balls.add( ball );
}
```

newで指定！

オーバーロード！

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- Ball(){…}とBall(float x, float y){…}と2つあってもよいの？
  - 引数の数とか型が違えば大丈夫！！
  - new のときに，どう呼び出すかの違い

    Ball() は Ball b = new Ball();

    Ball(float x, float y) は
    Ball b = new Ball(500.0, 100.0);

    で，それぞれ呼び出される．

オーバーロード（多重定義）
同じ関数の名前だけれど，引数の数や型を変えて振る舞いを変えることができるもの

new Ball(red, green, blue)

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- 色を指定したBallを作る

```
ArrayList<Ball> balls = new ArrayList<Ball>();

void setup() {
  size(400, 300);
}

void draw(){
  background(255);
  for(int i=0; i<balls.size(); i++){
    balls.get(i).move();
    balls.get(i).display();
  }
}

void mousePressed(){
  Ball ball = new Ball(random(255), random(255), random(255));
  balls.add( ball );
}
```

```
class Ball{
    :
  int red;
  int green;
  int blue;

  Ball(int r, int g, int b){
    posX = random(width);
    posY = random(height);
    speedX = random(1,5);
    speedY = random(1,5);
    red = r;
    green = g;
    blue = b;
  }
```

課題3-1: basic_clickSmall

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

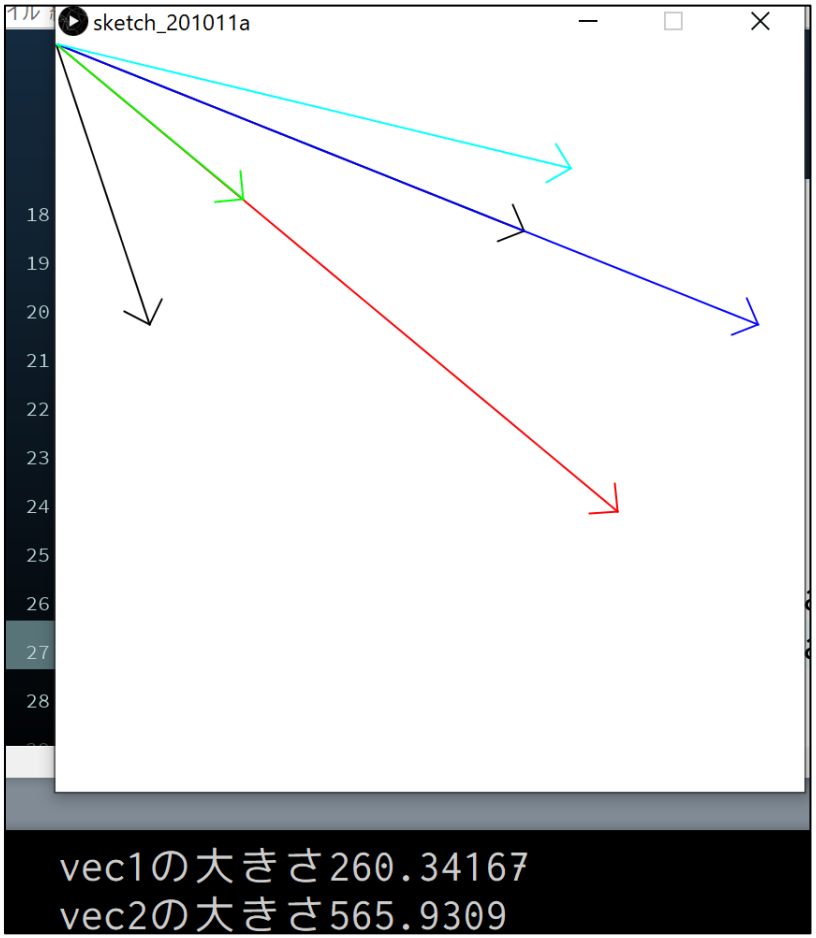
- 600x600のウインドウ内を動き回る5つの赤色の○を描画せよ。ただし、初期の円の半径は100ピクセルとし、円内をクリックするたびに半径の大きさが半分になっていくようにせよ
  - Ballというクラスを定義して実現せよ。
  - ○の初期位置はY座標が300の位置に、X座標が100，200，300，400，500と横一列に並べるようにせよ。コンストラクタでXY座標を受け取ると良い。
  - クラス内ではmouseX，mouseYを使わないようにせよ
  - ○は上下左右の壁で跳ね返るようにせよ（速度は-5〜5の実数値）
    - 跳ね返りを、円が壁に接したときに跳ね返るようにするか、円の中心で跳ね返るようにするかは、判断を任せます（壁に接したときに跳ね返るほうがそれっぽいけどね）

課題3-2: basic_MyVector

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- 配布する basic_MyVector.zip の MyVectorクラスのベクトルの
  - 足し算 add
  - 引き算 subtract
  - 掛け算 multiply
  - 割り算 divide
  - 大きさ magnitude

  の計算を行うメソッドを完成させ，右図のような出力になるようにせよ



課題3-3: basic_MyCharacter

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- 800x800のウインドウ内を動き回るキャラクタを描画するクラスMyCharacterとそれを使ったプログラムを作成せよ
  - ただし，MyCharacterクラス内のインスタンス変数は下記の2つのみとせよ（来週利用するため）
    - int centerX;
    - int centerY;
  - コンストラクタ（引数なし）で初期位置をウインドウ内にランダムに設定するようにせよ
  - キャラクタは (centerX, centerY) の位置に描画するが、2種類の感情で描画するようにせよ。具体的には、displayHappy() メソッドで喜びを displaySad() メソッドで悲しみを表現するようにせよ
  - 引数なしの move() というメソッドを用意し、自由に移動させるようにせよ。移動はランダムでも構わないが、ウインドウからは出ないようにせよ。なお、キャラクタ独自の動きがあるとなお良い。
- また、次ページのメインプログラムを用いて描画せよ（次ページにクラスがどうなるかも示す）

課題3-3: basic_Myharacter

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- メインプログラムとクラス定義は下記のような感じ

```
MyCharacter myChara;

void setup(){
  size(800, 600);
  myChara = new MyChara();
}

void draw(){
  background(255);
  myChara.move();
  if(mousePressed){
    myChara.displaySad();
  } else {
    myChara.displayHappy();
  }
}
```

```
class MyCharacter {
  int centerX;
  int centerY;

  MyCharacter(){
  }

  void move(){
  }

  void displayHappy(){
  }

  void displaySad(){
  }
}
```

宿題3-1: hw_GenomeAnalyzer

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- 以下の条件を満たすGenomeAnalyzerクラスを作成し，次のページで示すプログラムで動作させよ
  - コンストラクタの引数として塩基配列を文字列として与え，インスタンス変数でその塩基配列情報を保持するようにせよ
  - 返り値が実数値のcalcRatioA(), calcRatioT(), calcRatioG(), calcRatioC()というインスタンスメソッドを作成し，それぞれA,T,G,Cのそれぞれの出現率を返すようにせよ
    - わかる人は，引数をchar型にし，メソッドはcalcRatio()だけでもよい
  - 返り値が整数のlength()というインスンタンスメソッドを作成し，塩基配列の長さを返すようにせよ
  - 引数として文字列パターンを与え，返り値としてその文字列が塩基配列内で現れる回数を整数値で返すcountPatternメソッドを作成せよ．なおAAAAAAはAAAを2回とカウントせよ
    - 引数プログラム内でそれぞれcountPattern("AAA"), countPattern("TTT"), countPattern("GGG"), countPattern("CCC")のそれぞれの出現回数を出力するようにせよ

宿題3-1: hw_GenomeAnalyzer

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室


```
GenomeAnalyzer analyzer;

void setup() {
  size(400, 300);
  // hw_Covid19Genome の塩基配列を利用せよ↓
  analyzer = new GenomeAnalyzer("すんごく長い塩基配列");

  println("塩基配列の長さは" + analyzer.length() + "文字");

  println("Aの出現率", analyzer.calcRatioA());
  println("Gの出現率", analyzer.calcRatioG());
  println("Tの出現率", analyzer.calcRatioT());
  println("Cの出現率", analyzer.calcRatioC());

  println("AAAの出現回数", analyzer.countPattern("AAA"));
  println("GGGの出現回数", analyzer.countPattern("GGG"));
  println("TTTの出現回数", analyzer.countPattern("TTT"));
  println("CCCの出現回数", analyzer.countPattern("CCC"));
}
```

宿題3-2: hw_SlideShow

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- 以下の条件を満たすSlideShowクラスを作成し、スライドショーとして動作するプログラムを作成せよ
  - 内部に画像のリストを持つ（ArrayListでも配列でもよい）
  - show() メソッドにより現在の画像と、左右移動のボタンを表示
  - prev() メソッドが呼び出されると前の画像に移動
  - next() メソッドが呼び出されると後の画像に移動
  - 左移動のボタン（どのような形状でもよい）が押されると、prev() メソッドが呼び出されて前の画像に代わり、next() メソッドが呼び出されると後の画像に変わるようにせよ（画像は循環してもしなくてもよい）
  - 他に機能を追加しても良い

宿題3-3: hw_CalcVector

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- 2つのベクトルをメソッドでセットし，様々な計算を可能とするVectorOperationクラスを作成し，次ページのプログラムで動作を確認せよ
  - 引数を実数値の配列としてベクトルを与え，インスタンス変数としてベクトルを配列として管理するようにする setVectorA(), setVectorB()という2つのメソッドを作成せよ
  - 返り値が実数のcalcVectorMagnitudeA(), calcVectorMagnitudeB()メソッドを作成し，それぞれのベクトルの大きさを求めて返すようにせよ
  - 返り値が実数のcalcInnerProduct()メソッドを作成し，内部にもつ2つのベクトルの内積を求めて返すようにせよ
  - 返り値が実数のcalcCosSimilarity()メソッドを作成し，内部に持つ2つのベクトルのコサイン類似度を計算して返すようにせよ．2つのベクトルの類似度は，次ページ以降に示すコサイン類似度を利用することで求めることができる
  - 返り値が実数のcalcArea()メソッドを作成し，内部の2つのベクトルから面積を求め，返すようにせよ．2つのベクトルからなる三角形の面積は，次ページ以降の式で求めることが可能である

宿題3-3: hw_CalcVector

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- setVectorAとsetVectorBで指定されるベクトルの次元（配列の要素数）は，同じであると決めつけて良い

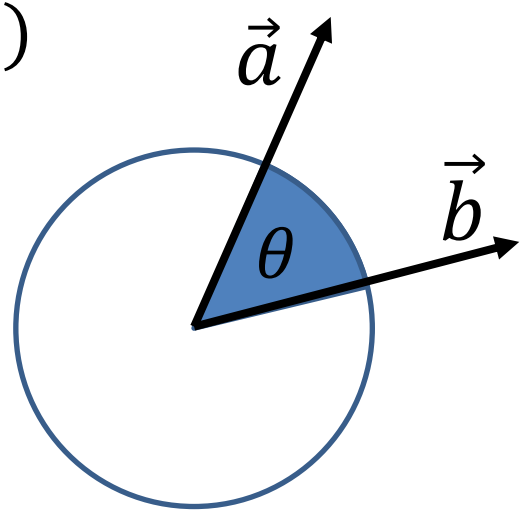
```
void setup() {
  VectorOperation vo = new VectorOperation();
  float[] vectorA = {10, 0, 1, 9};
  float[] vectorB = {8, 1, 1, 4};
  vo.setVectorA(vectorA);
  vo.setVectorB(vectorB);

  println("Aの大きさ", vo.calcVectorMagnitudeA());
  println("Bの大きさ", vo.calcVectorMagnitudeB());

  println("内積", vo.calcInnerProduct());
  println("コサイン類似度", vo.calcCosSimilarity());
  println("面積", vo.calcArea());
}
```

コサイン類似度

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- ベクトル $\vec{a}$ と $\vec{b}$ のなす角度を利用した類似度

$$\vec{a} = (a_1, a_2) \text{ と } \vec{b} = (b_1, b_2)$$



- ベクトルの内積の計算は...

$$\vec{a} \cdot \vec{b} = |\vec{a}| \left|\vec{b}\right| cos\theta$$

$$cos\theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \left|\vec{b}\right|} = \frac{a_1 \cdot b_1 + a_2 \cdot b_2}{\sqrt{a_1^2 + a_2^2} \cdot \sqrt{b_1^2 + b_2^2}}$$

コサイン類似度：3次元の場合

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- ベクトル $\vec{a}$ と $\vec{b}$ のなす角度を利用した類似度

$$\vec{a} = (a_1, a_2, a_3), \vec{b} = (b_1, b_2, b_3)$$

$$cos\theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \left|\vec{b}\right|} = \frac{a_1 \cdot b_1 + a_2 \cdot b_2 + a_3 \cdot b_3}{\sqrt{a_1^2 + a_2^2 + a_3^2} \cdot \sqrt{b_1^2 + b_2^2 + b_3^2}}$$

コサイン類似度：N次元の場合

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- ベクトル $\vec{a}$ と $\vec{b}$ のなす角度を利用した類似度

$$\vec{a} = (a_1, a_2, a_3 \ldots, a_N), \vec{b} = (b_1, b_2, b_3 \ldots, b_N)$$

$$cos\theta = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \left|\vec{b}\right|} = \frac{\sum_{i=1}^{N} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{N} a_i^2} \cdot \sqrt{\sum_{i=1}^{N} b_i^2}}$$

三角形の面積

明治大学総合数理学部
先端メディアサイエンス学科
中村研究室

- ベクトル $\vec{a}$ と $\vec{b}$ のからなる三角形の面積Sは下記の式で求めることができる

$$S = \frac{1}{2}\sqrt{|\vec{a}|^2 \left|\vec{b}\right|^2 - (\vec{a}\cdot\vec{b})^2}$$

  - 三次元の場合はこんな感じ

$$\vec{a} = (a_1, a_2, a_3), \vec{b} = (b_1, b_2, b_3)$$

$$S = \frac{1}{2}\sqrt{|\vec{a}|^2 \left|\vec{b}\right|^2 - (\vec{a}\cdot\vec{b})^2}$$

$$= \frac{1}{2}\sqrt{\left(a_1^2 + a_2^2 + a_3^2\right)\left(b_1^2 + b_2^2 + b_3^2\right) - (a_1\cdot b_1 + a_2\cdot b_2 + a_3\cdot b_3)^2}$$