



プログラミング演習2

ArrayList, HashMap

中村, 小松, 小林, 鹿喰

小テストの解答について



- 11/4から返却可能なので、希望者は小松部屋に来るように！
- 本日の小テストの解答については、本日中にアップロードします（他クラスの方もしっかりチェックしておきましょう！）

中間試験について



- 11月10日 4限目に311教室にて実施
 - 何らかの理由により欠席する場合は,【事前】に小松先生に連絡すること. また, 病欠の場合は医師の診断書を取ること.
 - 試験は4限開始なので, 電車が止まったとかそういう理由は(1日じゅう止まらない限り)認めません
- 試験範囲
 - 前期, 後期 (ArrayListとHashMapを除く) 分から
 - 3組・4組の4回分の小テスト(を多少いじったもの)から半分以上出します(他クラスの分までしっかり復習しておくように!)

先週の課題について

- クラスを交換して扱えるように
– 色々と便利になったけれど...

課題4-3 Robotクラスを交換

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



- 4-2で他の人が作成したマイロボットクラスと、自分のマイロボットクラス両方を用いて、ロボットを表示するプログラムを作成せよ

– 考え方

- 資料共有フォルダから他の人のをコピーして使用(基本的に隣の人のを使う)
- マイロボットクラスを提出する際の気配り:
 - (1)基本的なメソッドは、コンストラクタと、void init(), void move(), void display(), void pressed() の4個。他の機能がある場合はコメントに書いて使ってもらおう!
 - (2) .pdeファイルだけを共有するので、画像や音声等の外部ファイルは使わない。
- Enjoy!

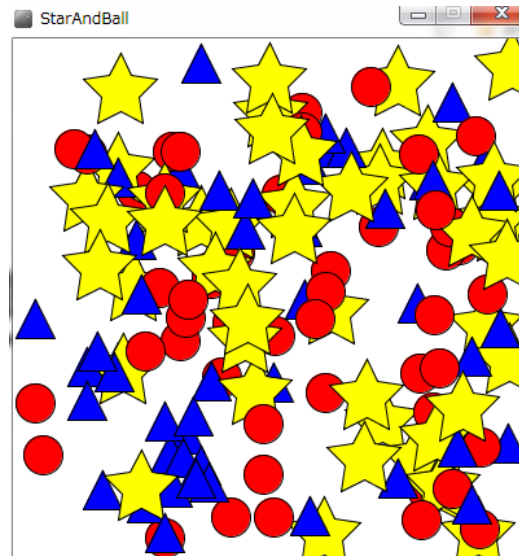
```
ItoRobo0404 itoken;  
MyRobot aero;  
AbeRobo0402A abetyan;  
ObuchiRobo0414A cat;  
YoshiokaRobo0450 airou;  
SasakiRobo0427A teitoku;  
KoikeRobo0423A koike;  
TodaRobo0437A toda;
```

```
void setup(){  
  size(800,600);  
  itoken=new ItoRobo0404();  
  aero=new MyRobot();  
  abetyan=new AbeRobo0402A();  
  airou=new YoshiokaRobo0450();  
  teitoku=new SasakiRobo0427A();  
  cat=new ObuchiRobo0414A();  
  koike=new KoikeRobo0423A();  
  toda=new TodaRobo0437A();  
}  
void draw(){  
  background(255);  
  itoken.move();  
  aero.move();  
  abetyan.move();  
  airou.move();  
  teitoku.move();  
  cat.move();  
  koike.move();  
  toda.move();  
  if(mousePressed){  
    itoken.pressed();  
    aero.pressed();  
    abetyan.pressed();  
    airou.pressed();  
    teitoku.pressed();  
    cat.pressed();  
    koike.pressed();  
    toda.pressed();  
  }else{  
    itoken.display();  
    aero.display();  
    abetyan.display();  
    airou.display();  
    teitoku.display();  
    cat.display();  
    koike.display();  
    toda.display();  
  }  
}
```

課題3-2



- Objectクラスを継承し，☆が動き回るStarクラスを作成せよ．また，BallクラスとStarクラス，Triangleクラスを利用して，50個の赤色丸と50個の黄色星と50個の青色△を動かすようにせよ
 - 星の内部は塗りつぶせないようだったら，線の色を黄色にせよ



解答例

```
Star [] stars = new Star [50];  
Ball [] balls = new Ball [50];  
Triangle [] triangles = new Triangle [50];
```

```
void setup() {  
  size( 400, 400 );  
  for ( int i=0; i<50; i++ ) {  
    stars[i] = new Star();  
    balls[i] = new Ball();  
    triangles[i] = new Triangle();  
  }  
}
```

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

なんかやっぱり
無駄な気がする





- Array型は同じ型のものを複数管理するクラス
 - `String [] students = new String [101];`
 - `int [][] lights = new int [100][100];`
 - `Ball [] balls = new Ball [50];`
- 違う形のオブジェクトをどう管理したら良い？
 - 先週やった色々な Robot 型を管理したい！

実を言うと

- 親クラスと一緒に呼び出すメソッドが同じならまとめて配列に！

```
Robot [] robots = new Robot [8];
```

```
void setup(){  
  size(800,600);  
  robots[0]=new ItoRobo0404();  
  robots[1]=new MyRobot();  
  robots[2]=new AbeRobo0402A();  
  robots[3]=new YoshiokaRobo0450();  
  robots[4]=new SasakiRobo0427A();  
  robots[5]=new ObuchiRobo0414A();  
  robots[6]=new KoikeRobo0423A();  
  robots[7]=new TodaRobo0437();  
}  
void draw(){  
  background(255);  
  for( int i=0; i<robots.length; i++ ){  
    robots[i].move();  
    if( mousePressed ){  
      robots[i].pressed();  
    } else {  
      robots[i].display();  
    }  
  }  
}
```




- 親クラスと一緒に呼び出すメソッドが同じならまとめて配列に！

```
Object [] objs = new Object [150];

void setup() {
  size( 400, 400 );
  for ( int i=0; i<50; i++ ) {
    objs[i*3+0] = new Star();
    objs[i*3+1] = new Ball();
    objs[i*3+2] = new Triangle();
  }
}

void draw() {
  background(255);
  for ( int i=0; i<150; i++ ) {
    objs[i].move();
    objs[i].display();
  }
}
```

覚える必要はないけれど



- ポリモーフィズム (多態性, 多様性)
 - 複数の型に属することを許すこと. 親クラスから継承された各種のインスタンスは, それぞれ親クラスの型に代入することができる. 呼び出されるのは, その継承されたクラスのメソッドとなるため, 親クラスにそのメソッドがないとエラーになる
- アップキャスト
 - 親クラスで定義されたものに, 継承されたクラスのインスタンスが代入されると, 自動的にアップキャストされる
 - ダウンキャストは問題だけれど, アップキャストは基本的に問題なし (全部継承されるため)

数の定義面倒

- 配列の大きさを事前に指定しておくのは柔軟性がなくて面倒！
- もっと柔軟にオブジェクト集合を扱いたい！

```
Robot [] robots = new Robot [8];
```

```
void setup(){  
  size(800,600);  
  robots[0]=new ItoRobo0404();  
  robots[1]=new MyRobot();  
  robots[2]=new AbeRobo0402A();  
  robots[3]=new YoshiokaRobo0450();  
  robots[4]=new SasakiRobo0427A();  
  robots[5]=new ObuchiRobo0414A();  
  robots[6]=new KoikeRobo0423A();  
  robots[7]=new TodaRobo0437();  
}  
  
void draw(){  
  background(255);  
  for( int i=0; i<robots.length; i++ ){  
    robots[i].move();  
    if( mousePressed ){  
      robots[i].pressed();  
    } else {  
      robots[i].display();  
    }  
  }  
}
```



- ArrayList型は，可変で色々なものを格納できるクラス

(例) `ArrayList list = new ArrayList();` で定義

– ArrayListの要素数を取得

- `list.size();`

– ArrayListに対するaddで要素を追加

- `list.add(value);` // valueを追加

– ArrayListに対するremoveで要素を削除

- `list.remove(index);` // index番目を削除

– ArrayListに対するgetで要素を取得

- `list.get(index);` // index番目のオブジェクトを取得

ArrayList + Object

- ArrayList型の定義
- add()でリストに追加
- size()でリストのアイテム数を取得
- get()でリストからアイテム(オブジェクト)を取得

```
ArrayList list = new ArrayList();
void setup(){
    size(800,600);
    list.add( new ItoRobo0404() );
    list.add( new MyRobot() );
    list.add( new AbeRobo0402A() );
    list.add( new YoshiokaRobo0450() );
    list.add( new SasakiRobo0427A() );
    list.add( new ObuchiRobo0414A() );
    list.add( new KoikeRobo0423A() );
    list.add( new TodaRobo0437() );
}
void draw(){
    background(255);
    for( int i=0; i<list.size(); i++ ){
        Robot robo = (Robot)list.get(i);
        robo.move();
        if( mousePressed ){
            robo.pressed();
        } else {
            robo.display();
        }
    }
}
```

ArrayList + Object



```
ArrayList list = new ArrayList();
```

```
void setup() {  
    size( 400, 400 );  
    for ( int i=0; i<50; i++ ) {  
        list.add( new Star() );  
        list.add( new Ball() );  
        list.add( new Triangle() );  
    }  
}
```

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

```
for ( int i=0; i<50; i++ ) {  
    Star star = new Star();  
    list.add( star );  
    Ball ball = new Ball();  
    list.add( ball );  
    Triangle tri = new Triangle();  
    list.add( tri );  
}
```



list.size() で数を取得

list.get(i) でi番目の要素を取得

(Object) で Object 型と明示



- HashMap型は、何かをキーとして値を格納するものであり、キーを利用して値を取り出せる

```
HashMap<キーの型, 格納する値の型> dic;
```

```
dic = new HashMap<キーの型, 格納する値の型>();
```

- HashMapの要素数を取得

- dic.size();

- HashMapに対する要素を追加

- dic.put(key, value); // keyでvalueを追加

- HashMapに対するgetで要素を取得

- dic.get(key); // keyに該当する値を取得

HashMap



- 辞書みたいなもの
- 何かのキーから
値を取得する

```
HashMap<String, String> aadic  
    = new HashMap<String, String>();
```

```
void setup() {  
    aadic.put( "haa", "(°Д°)ハア?" );  
    aadic.put( "ase", "(;´Д`)" );  
    aadic.put( "kita", "キター(°▽°)ー!" );  
}
```

```
void draw() {  
    background(255);  
    println( aadic.get("haa") );  
    println( aadic.get("ase") );  
    println( aadic.get("kita") );  
}
```


HashMap

```
HashMap<String, Robot> dict;  
dict = new HashMap<String, Robot>();
```

```
void setup() {  
    dict.put( "miyashita", new MiyaRobot() );  
    dict.put( "komatsu", new KomatsuRobot() );  
    dict.put( "kobayashi", new KobaRobot() );  
}
```

```
void draw() {  
    background(255);  
    Robot robo = (Robot)dict.get( "miyashita" );  
    robo.display();  
    Robot robo = (Robot)dict.get( "komatsu" );  
    robo.display();  
    Robot robo = (Robot)dict.get( "kobayashi" );  
    robo.display();  
}
```

ソート(並び替え)



- 来年度のアルゴリズム基礎(福地先生)で登場
- `int [] data = { 50, 30, 80, 90, 40, 100, 20, 55, 35, 60};`
という配列の値を並び替えるにはどうするか？
- バブルソート
 - 一番左から順に隣と比較していき, 隣り合う要素の
大小を入れ替えていくことによって並び替えを行う
 - 入れ替えが起こらなかったら終了！

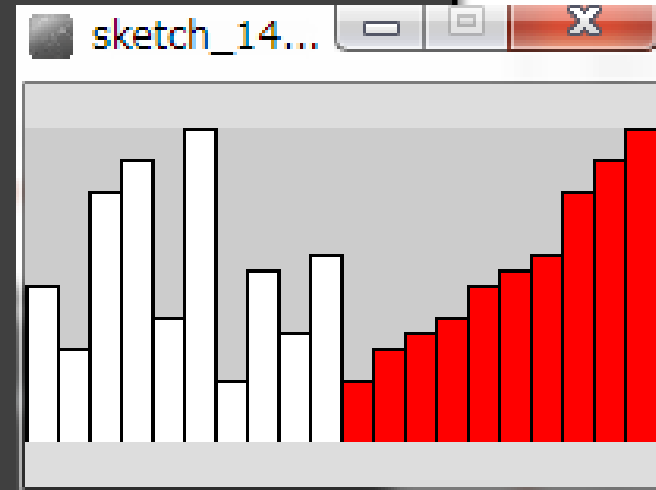
ソートしてみよう！



```
int [] data = {  
    50, 30, 80, 90, 40, 100, 20, 55, 35, 60  
};  
  
size( 200, 100 );  
for ( int i=0; i<data.length; i++ ) {  
    rect( i*10, height-data[i], 10, data[i] );  
}
```

// ここで並べ替える！

```
fill( 255, 0, 0 );  
for ( int i=0; i<data.length; i++ ) {  
    rect( i*10+100, height-data[i], 10, data[i] );  
}
```





- 課題5-1
 - 先週のロボット課題(他人のRobotを使う)を単純な配列で実現せよ. ただしロボットは5個以上とする
- 課題5-2
 - 同じくArrayListを使って実現せよ. ただしロボットは10個以上とする
- 課題5-3
 - 配列の値を並び替えてみよう!
 - そしてその結果を赤色の四角形で出力せよ!



```
size( 200, 100 );  
background( 255 );  
int [] data = new int [100];  
for ( int i=0; i<data.length; i++ ) {  
    data[i] = (int)random(100);  
    line( i, height, i, height-data[i] );  
}  
  
// ↓並び替え開始  
  
// ↑並び替え終了  
  
stroke( 255, 0, 0 );  
for ( int i=0; i<data.length; i++ ) {  
    line( i+100, height, i+100, height-data[i] );  
}
```