# Java講座

## ブロック崩し

情報科学部コンピュータ科学科
2年　竹中　優

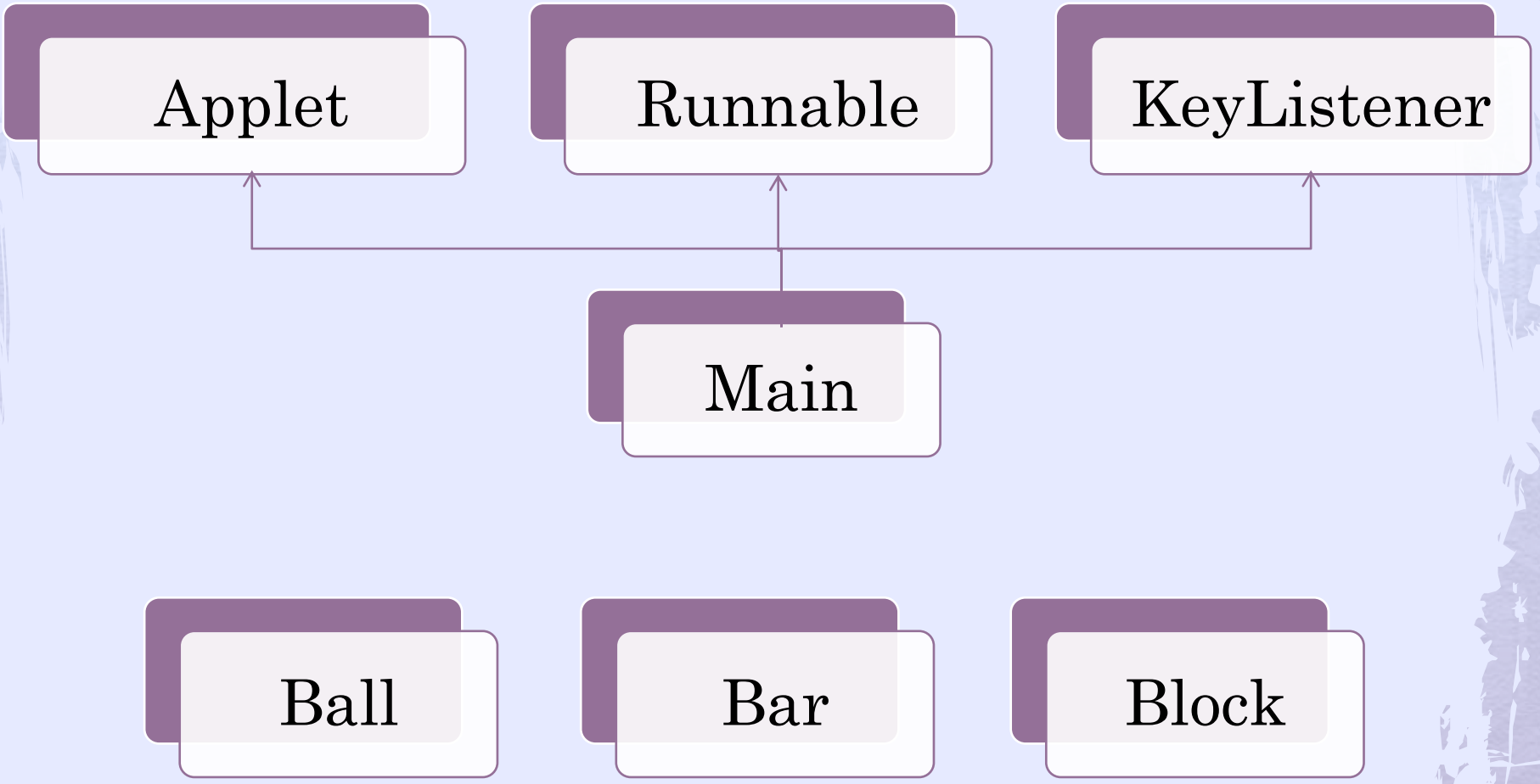
# 内容

- ブロック崩しに必要なクラスを考えよう
  - クラス構造を考える
  - クラスを設計する
- 当たり判定
- 実行クラスを完成させる

# ブロック崩しに必要なクラスを考えよう

- とりあえず、ボールとバーとブロックを表すクラスが必要である。
- あとは、それらをまとめる実行クラス(Appletクラスのサブクラス)が必要である。
- クラス名は、それぞれBall, Bar, Block, Mainクラスで良いだろう。

# クラス構造



# クラス設計　Mainクラス

# クラス設計　Mainクラス

- スーパークラス：Applet
  実装(implements)するインターフェイス：
  KeyListener, Runnable

- オーバーライドするAppletのメソッド
  public void init()
  public void paint(Graphics g)

- 実装しなければならないメソッド
  public void keyTyped(KeyEvent e)
  public void keyPressed(KeyEvent e)
  public void keyReleased(keyEvent e)
  public void run()

# クラス設計　Mainクラス

- フィールドは、
  再描画ごとに動くボール　Ball ball
  左右矢印キーで動くバー　Bar bar
  画面に配置されるブロックの2次元配列
  　　Block[][] blocks
  一定ミリ秒間隔でrepaintメソッドを呼び、再描画するスレッド
  　　Thread repaintThread
  描画するシーンを表す変数
  　　int scene

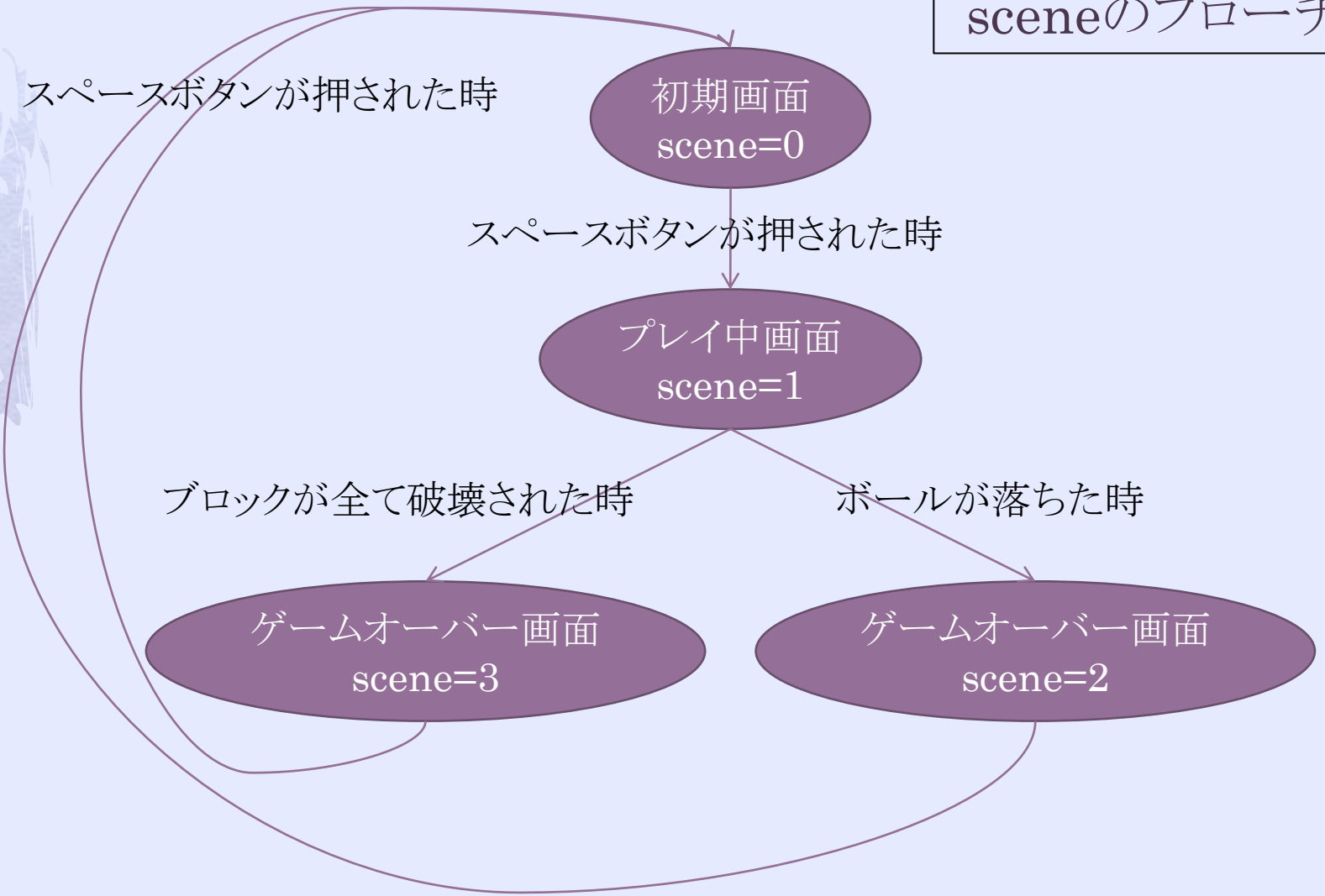
# クラス設計　Mainクラス

- sceneについて
  0:初期画面
  1:プレイ中画面
  2:ゲームオーバー画面
  3:ゲームクリア画面

- paint(Graphics g)メソッドの中でsceneの値に対応する画面を描画するメソッドを呼び出す。

# クラス設計　Mainクラス

sceneのフローチャート



# クラス設計　Mainクラス

- メソッドは、
  Appletクラスのinit, paintメソッドをオーバーライドする
  KeyListenerインターフェイスの抽象メソッド
  　　keyTyped(KeyEvent e)
  　　keyPressed(KeyEvent e)
  　　keyReleased(KeyEvent e)
  Runnableインターフェイスの抽象メソッド
  　　run()
  scene=0の時にpaintメソッドが呼ぶメソッド
  　　ready(Graphics g)
  scene=1　　〃
  　　playing(Graphics g)
  scene=2　　〃
  　　gameover(Graphics g)
  scene=3　　〃
  　　gameclear(Graphics g)
- それぞれ記述が長いので、処理は後で記述する(名前だけ宣言しておこう)

# クラス設計　Ballクラス

# クラス設計　Ballクラス

- フィールドは、
  中心座標　double x, y
  再描画ごとの座標の変化量　double dx, dy
  半径　int r
  ゲームオーバーかどうか
  boolean isGameOver
  が必要となるだろう。
  そのほかは書いていくうちに随時追加していく。

# クラス設計　Ballクラス

- メソッドは、
  ボールを座標x, yに描画するdrawメソッド
  ボールの座標を移動させるmoveメソッド
  (反射するべき場合は反射する)
  が必要となるだろう。
  そのほかは書いていくうちに随時追加していく。

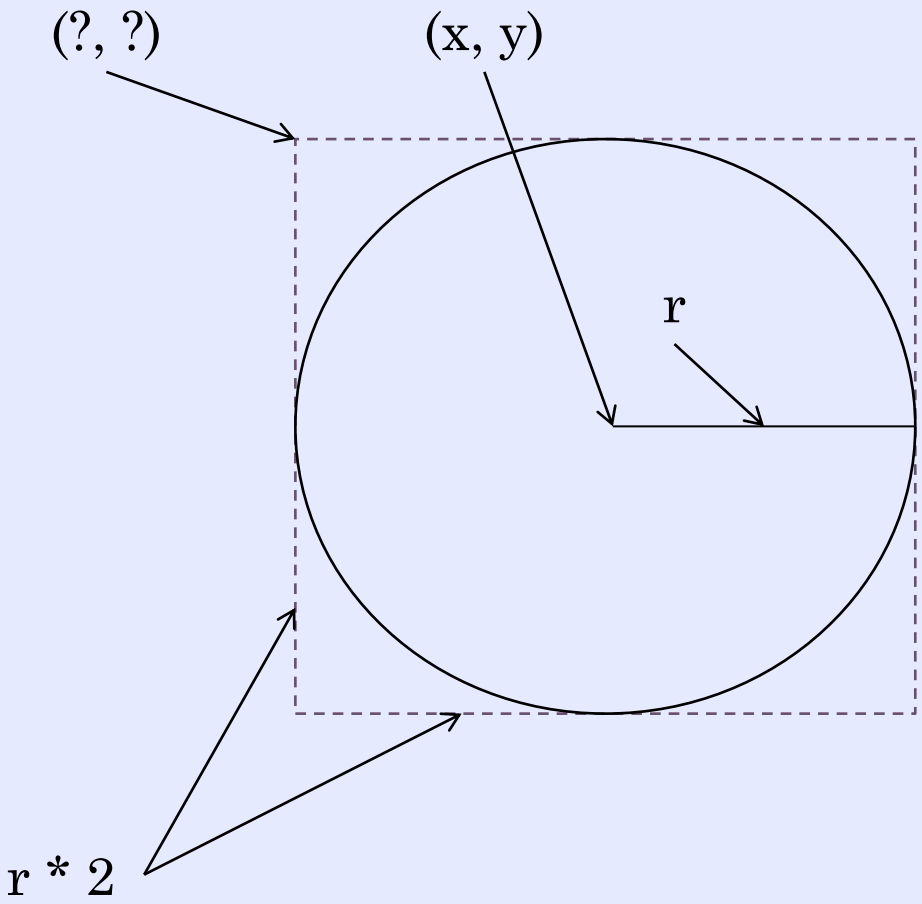
- 次項以降でdraw, moveメソッドを定義していく。

# クラス設計　Ballクラス

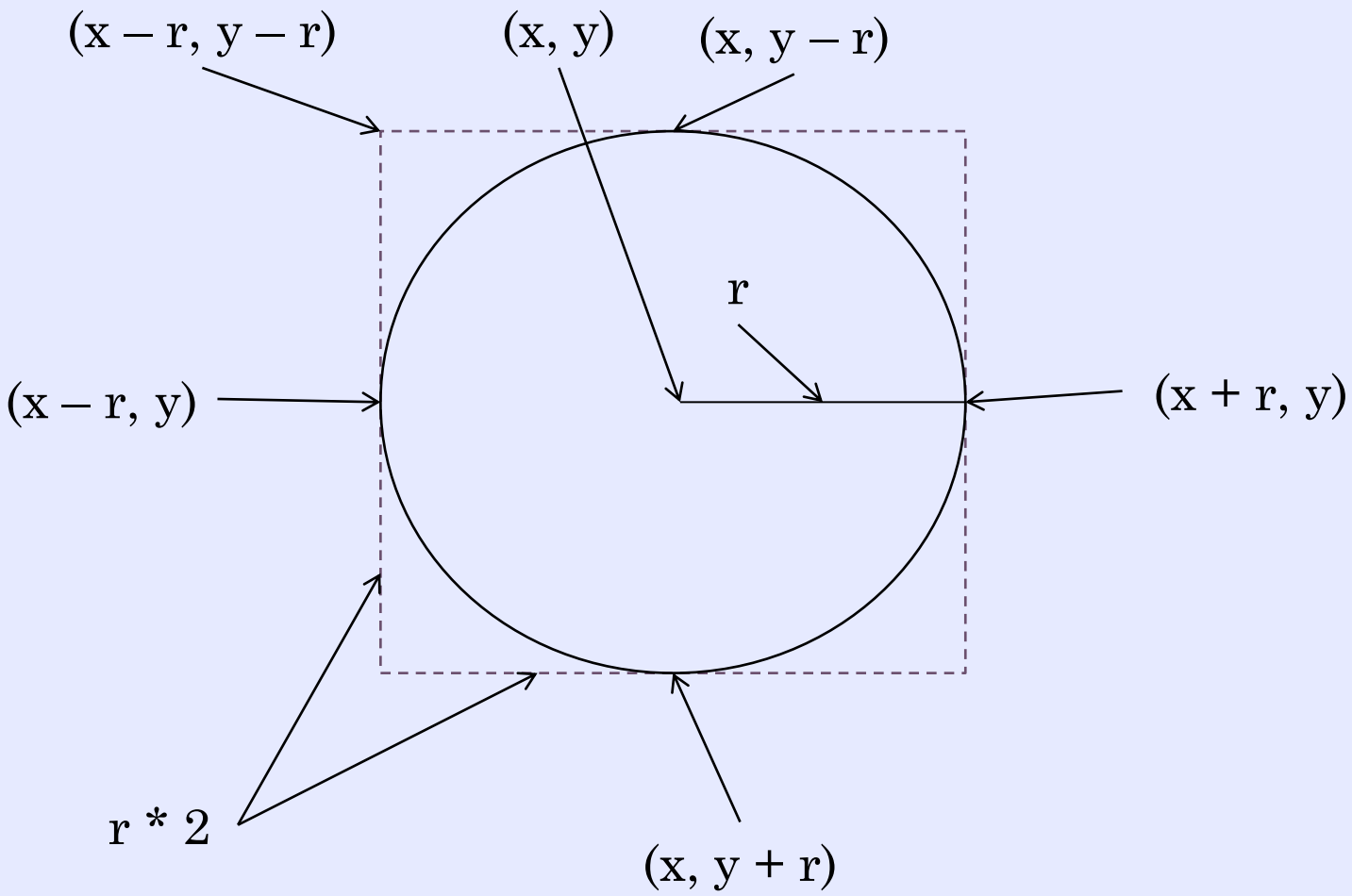
```
public void draw(Graphics g){
    g.drawOval((int)x – r,(int)y – r,r * 2,r * 2);
    //g.fillOval((int)x – r,(int)y – r,r * 2,r * 2);
}
```

- Graphicsオブジェクトを引数として受け取り、それを使って円を描画する。
- 今回はfillOvalではなく、drawOvalを使用する。
- 次項で座標指定の解説をする

# クラス設計　Ballクラス



# クラス設計　Ballクラス



# クラス設計　Ballクラス

```
public void move(){
        x += dx;
        y += dy;

        /*壁との当たり判定を記述*/
        /*バーとの当たり判定を記述*/
        /*ブロックとの当たり判定を記述*/
}
```

- 壁、バー、ブロックとの当たり判定は、後に解説する

# クラス設計　Barクラス

# クラス設計　Barクラス

- フィールドは、
  左上の座標　double x, y
  幅　int width
  高さ　int height
  横移動する速さ(変化量)　double dx

# クラス設計　Barクラス

- メソッドは、
  バーを描画するdrawメソッド
  バーを移動するmoveメソッド

- 次項以降でdraw, moveメソッドを定義していく。

# クラス設計　Barクラス

```
public void draw(Graphics g){
    g.drawRect((int)x, (int)y, width, height);
}
```

- Ballクラスと同様にGraphicsオブジェクトを引数として受け取り、それを使って長方形を描画する。

# クラス設計　Barクラス

```
public void move(boolean isRight){
        if(isRight)
                x += dx;
        else
                x -= dx;
}
```

- 引数isRightがtrueの場合、バーを右に移動させ、falseの場合、バーを左に移動させる。
- すなわち、isRightは矢印キーが「→」の場合、true「←」の場合、falseとする。

# クラス設計　Blockクラス

# クラス設計　Blockクラス

- 同じ長方形なので、Barクラスとほぼ同じ

- フィールドは、
  左上の座標　double x, y
  幅　int width
  高さ　int height
  既に壊されたかどうか　boolean isBroken

# クラス設計　Blockクラス

- メソッドは、
  ブロックを描画するdrawメソッド

- 次項以降でdrawメソッドを定義していく。

# クラス設計　Blockクラス

```
public void draw(Graphics g){
    if(isBroken){
        g.drawRect((int)x, (int)y, width, height);
    }
}
```

- Ball, Barクラスと同様にGraphicsオブジェクトを引数として受け取り、それを使って長方形を描画する。
- フィールドisBrokenがtrueの場合は描画し、falseの場合は描画しない。

# 当たり判定

# 実行クラスを完成させる

# 終わり