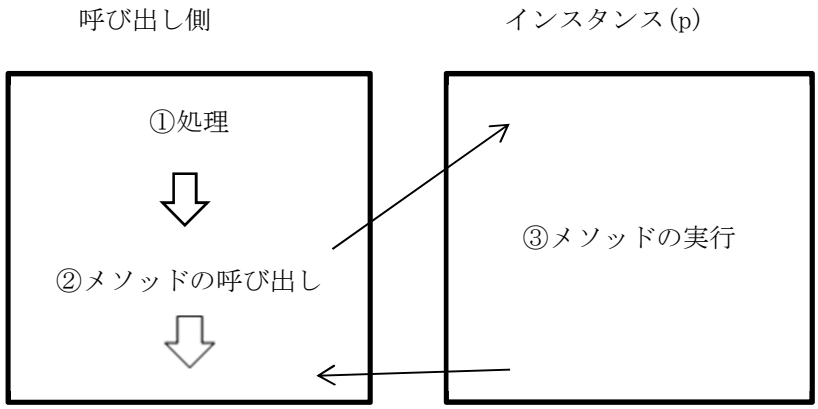
６　メソッドの活用

例

```
class Point
{
    int x;
    int y;
 ③ void printPosition
    {
        canvas.drawText("座標値は（"  + this.x + "," + this.y + "）
                                                    です", 10, 20, paint);
    }
}
class MethodCallExample
{
    point p = new Point();  //インスタンスの生成
①   p.x = 10;
    p.y = 5;
②   p.printPosition();  //メソッドの呼び出し
}
```

6.1 メソッドを呼び出したときの処理の流れ



6.2 メソッドの活用のソースリスト

6.2.1　テンプレート(MainActivity.java)の①の箇所に　Main040View　を入力してください。

6.2.2　ClassGroup.java　を書き換えてください。

6.2.3　Main030View.java をコピーして　Main040View.java　を新規作成してください。

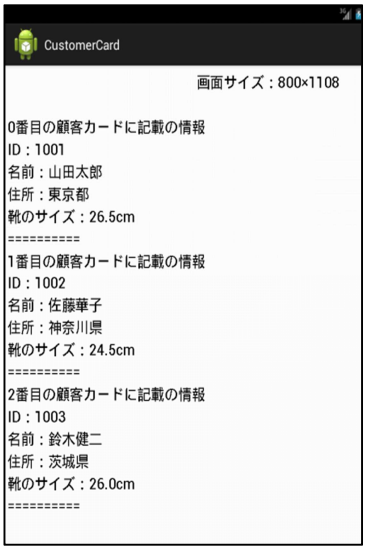
ファイル名：　src/jp/edu/mie/ClassGroup.java

```
package jp.edu.mie;
・・・
class CustomerCard
{
    ・・・
    double shoeSize;//靴のサイズ(cm)

    public CustomerCard() {}
```

```
    public CustomerCard(int id, String name, String address, double shoeSize)
    {
        this.id = id;
        this.name = name;
        this.address = address;
        this.shoeSize = shoeSize;
    }

    public int printInfo(Canvas canvas, Paint paint, int y, int INTERVAL)
    {
        canvas.drawText("ID：" + this.id, 10, y += INTERVAL, paint);
        canvas.drawText("名前：" + this.name, 10, ye += INTERVAL, paint);
        canvas.drawText("住所：" + this.address, 10, y+= INTERVAL, paint);
        canvas.drawText("靴のサイズ：" + this.shoeSize + "cm"
                                                , 10, y += INTERVAL, paint);
        return y;
    }
}
```



ファイル名：　src/jp/edu/mie/Main040View.java

```
package jp.edu.mie;
・・・
public class Main040View extends View
{
    public Main040View(Context context)
    {
        ・・・
    }
    protected void onDraw(Canvas canvas)
    {
        ・・・
        canvas.drawText("画面サイズ：" + getWidth()
                + "×" + getHeight(), getWidth()/2, 40, paint);
        CustomerCard[] cards = new CustomerCard[100]; //配列の作成

        int y;//y 座標
        final int INTERVAL = 30;//y 座標の間隔

        cards[0] = new CustomerCard(1001, "山田太郎", "東京都", 26.5);
        cards[1] = new CustomerCard(1002, "佐藤華子", "神奈川県", 24.5);
        cards[2] = new CustomerCard(1003, "鈴木健二", "茨城県", 26.0);

        y = 80;
        for(int i = 0; i < cards.lenght; i++)
        {
            if(cards[i] == null){
                break;//参照のない要素があったらループ処理を終了
            }
            canvas.drawText(Integer.toString(i)
                                        + "番目の顧客カードに記載の情報"
```

```
                                                , 10, y += INTERVAL, paint); //①
                y = cards[i].printInfo(canvas, paint,
                                        y, INTERVAL);//②メソッドの呼び出し（引数あり）
                canvas.drawText("====================", 10,
                                                        y += INTERVAL, paint);
            }
        }
    }
```

6.3　メソッドの活用のソースリストの解説

Java のプログラムを構成しているのがメソッドです。

このプログラムで一番外側にある class と書かれたブロックのことをクラスといい、その内側にある printInfo ()などのブロックのことをメソッドといいます。

this とは自分自身を表すキーワードです。メソッドの中でインスタンス変数を参照するために使用します。

CustomerCard クラスに、情報を画面に表示できる printInfo メソッドが追加されたことで、情報の確認が簡単にできるようになりました。また、引数のあるコンストラクタが宣言されたことで、インスタンスの生成と情報の設定を一行で行えるようになりました。これにより、情報の設定忘れを防ぐことができます。

①canvas.drawText(Integer.toString(i) + "番目の顧客カードに記載の情報"
                                                , 10, y += INTERVAL, paint);

「y += INTERVAL」は「y = y + INTERVAL」 のことであり、y 座標の計算式です。

②y = cards[i].printInfo(canvas, paint, y, INTERVAL);

cards[i].printInfo;という命令文によって、配列に入っている要素の種類に応じた printInfo メソッドが実行されていることが確認できます。ポリモーフィズムが有効に働いていることがわかります。