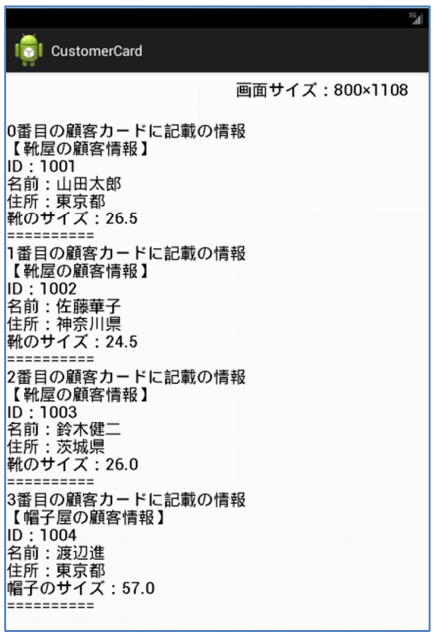
８　継承の活用

8.1　継承の活用のソースリスト

8.1.1　テンプレート(MainActivity.java)の①の箇所に　Main060View　を入力してください。

8.1.2　ClassGroup.java　を書き換えてください。

8.1.3　Main050View.java をコピーして
Main060View.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
{
    ・・・

    public CustomerCard() {}

    public CustomerCard(String name)
    {
        ・・・
    }

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

    public CustomerCard(int id, String name, String address, double shoeSize)
    {
        ・・・
    }

    public int printInfo(Canvas canvas, Paint paint, int y, int INTERVAL)
    {
        ・・・
    }
}
```

ファイル名：　src/jp/edu/mie/ Main060View.java

```
package jp.edu.mie;
・・・
public class Main060View extends View    //①
{
    public Main060View(Context context)
```

```
    {
        ・・・
    }
    protected void onDraw(Canvas canvas)    //②
    {
        ・・・
        CustomerCard[] cards = new CustomerCard[100];//配列の作成
        cards[0] = new ShoeShopCustomerCard("山田太郎", "東京都", 26.5);
        cards[1] = new ShoeShopCustomerCard("佐藤華子", "神奈川県", 24.5);
        cards[2] = new ShoeShopCustomerCard("鈴木健二", "茨城県", 26.0);
        cards[3] = new HatShopCustomerCard("渡辺進", "東京都", 57.0);
        ・・・
    }
}

class ShoeShopCustomerCard extends CustomerCard
{
    private double shoeSize;

    ShoeShopCustomerCard(String name, String address, double shoeSize)
    {
        super(name, address);
        if(shoeSize > 5 && shoeSize < 50){ this.shoeSize = shoeSize; }
    }
    ShoeShopCustomerCard(String name)
    {
        super(name);
    }
    public int printInfo(Canvas canvas, Paint paint,
                                                    int y, int INTERVAL)
    {
        canvas.drawText("【靴屋の顧客情報】",
                                                10, y += INTERVAL, paint);
        y = super.printInfo(canvas, paint, y, INTERVAL);
        canvas.drawText("靴のサイズ："+ shoeSize,
                                                10, y += INTERVAL, paint);
        return y;
    }
}

class HatShopCustomerCard extends CustomerCard
{
    private double hatSize;

    HatShopCustomerCard(String name, String address, double hatSize)
    {
        super(name, address);
        if(hatSize > 5 && hatSize < 50){ this.hatSize = hatSize; }
    }
    HatShopCustomerCard(String name)
```

```
    {
        super(name);
    }
    public int printInfo(Canvas canvas, Paint paint,
                                                    int y, int INTERVAL)
    {
        canvas.drawText("【帽子屋の顧客情報】",
                                                10, y += INTERVAL, paint);
        y = super.printInfo(canvas, paint, y, INTERVAL);
        canvas.drawText("帽子のサイズ："+ hatSize,
                                                10, y += INTERVAL, paint);
        return y;
    }
}
```

8.2　継承の活用のソースリストの解説

①public class Main060View extends View

Viewクラスには多くのサブクラスがあります。ビューはViewクラスで定義されています。

Viewクラスを「継承」してAndroidクラスを作成するという意味です。Viewクラスはソースのどこにも書いていませんが、Android SDKがあらかじめ用意しているクラスであり、Androidアプリケーションの画面を担当するクラスです。Androidクラスは、Viewクラスのメソッドをそのまま引き継ぐだけでなく、書き換えることができます。これをメソッドのオーバーライドと呼びます。

②protected void onDraw(Canvas canvas)

onDraw()メソッドはコールバックメソッドで、システムから呼び出されます。呼び出されるタイミングはアプリケーションが直接決めるものではなく、もちろんアプリケーションが自分で呼び出すものでもありませんが、Activityが起動した直後には確実に呼び出されます。

8.3　アクセス修飾子

他のクラスや他のパッケージからのアクセスを許可するかどうかを指定するものです。[public][指定なし][protected][public]の４種類があります。

呼び出し元クラス	private	指定なし	protected	public
同じクラス	○	○	○	○
同じパッケージ内のサブクラス	×	○	○	○
同じパッケージ内の非サブクラス	×	○	×	○
他のパッケージ内のサブクラス	×	×	○	○
他のパッケージ内の非サブクラス	×	×	×	○

8.4　カプセル化

```
class ShoeShopCustomerCard extends CustomerCard
{
    private double shoeSize;

    ShoeShopCustomerCard(String name, String address, double shoeSize)
    {
        super(name, address);
        if(shoeSize > 5 && shoeSize < 50){ this.shoeSize = shoeSize; }
    }
```

ShoeShopCustomerCard クラスの外から、private メンバの shoeSize を直接設定することはできません。しかし、その代わりに、public メンバの ShoeShopCustomerCard ()コンストラクタドならば呼び出すことができます。

このコンストラクタを使うと、正しい値であるかどうかを必ずチェックしてから、靴のサイズが設定されます。つまり、誤った靴のサイズが設定されることがなくなり、誤りのおきにくいプログラムを作成できます。

このように、クラスの中にデータ（フィールド）と機能（メソッド）をひとまとめにし、保護したいメンバに private を付けて勝手にアクセスできなくする機能を**カプセル化**といいます。一般的には、フィールドは、private メンバ、メソッドは、public メンバを指定します。