多態性(ポリモルフィズム)について

```
// ButtonObject クラス(抽象クラス)
abstract class ButtonObject {
	String button;
	// 抽象クラスでもコンストラクタは記述できる
	// サブクラスのインスタンス生成に必要
	public ButtonObject(String btn) {
		this.button = btn;
	}
	// 通常のメソッド
	public void push() {
		System.out.println("電源ボタンが押されました。");
	}
	// 抽象メソッド(処理は継承したサブクラスで記述する)
	/*public*/abstract void push(int i);
}// ButtonObject

// Tv クラス
class Tv extends ButtonObject {
	// コンストラクタ
	public Tv() {
		super("チャンネル");
		System.out.println("TV を買いました。");
	}
	// 電源ボタンを押す(メソッドの上書き)
	@Override
	public void push() {
		// オーバーライド前のメソッドを呼び出す
		super.push();
		System.out.println("画面が写りました。");
	}
	// チャンネルボタンを押す(抽象メソッドの実装)
	@Override
	public void push(int a) {
		System.out.println(button + "ボタンが押されました。");
		System.out.println(a + "チャンネルに変わりました。");
	}
}// Tv

//CdPlayer クラス
class CdPlayer extends ButtonObject {
	// コンストラクタ
	public CdPlayer() {
		super("選曲");
		System.out.println("CD プレーヤーを買いました。");
	}
	// 電源ボタンを押す(メソッドの上書き)
	@Override
	public void push() {
```

```
                super.push();
                System.out.println("選曲待ちになりました。");
        }
        // 選曲ボタンを押す(抽象メソッドの実装)
        @Override
        public void push(int a) {
                System.out.println(button + "ボタンが押されました。");
                System.out.println(a + "曲目が再生されました。");
        }
}// CdPlayer

// 実行クラスの ButtonTest クラス
class ButtonTest {
        public static void main(String[] args) {
                // TV と CD プレイヤーを配列に格納
                ButtonObject[] bo = {new Tv(), new CdPlayer(),};
                // 二つのサブクラスにまとめて指示を出す
                for(int i = 0; I < bo.length; i++) {
                        bo[i].push();
                        bo[i].push(2);
                        System.out.println("");
                }
        }
}// ButtonTest
```

＜**多態性(ポリモルフィズム)**＞とは
異なるオブジェクトに対して、同じメッセージを送った場合、そのオブジェクトに合わせて異なる処理が適切になされること。
＜ポイント＞
・Tv クラスと CdPlayer クラスは ButtonObject クラスを継承し、各メソッドをオーバーライドしている。
・各インスタンスは main メソッドから push()という指示を受け取ると適切な処理を行う。
・利用者(main メソッド)は番号のボタンを押したい場合、push(番号)という形式でメッセージを送ればよい。

つまり、TV クラスと CdPlayer クラスの利用者は、push()というメソッド使い方さえ知っていれば、
それぞれの push()に応じて適切な処理をしてくれるのです。
家電製品をはじめとしてボタンは色々なものについていますが、「ボタンは押す」という操作は共通しています。
TV のボタンを押すと映像が画面に映り、CD プレイヤーの再生ボタンを押すと曲が再生されます。
私達は「ボタンを押されると何故番組が画面に映るのか？」「CD はどうやって曲を再生しているのか？」と
いったことは知らなくても、「ボタンは押す」ということさえ知っていればよいのです。

これを先ほどの多態性の定義に当てはめると、
異なるオブジェクトに対して【TV や CD プレーヤーに対して】
同じメッセージを送った場合【ボタンを押すというメッセージを送った場合】
そのオブジェクトに合わせて**異なる処理が適切になされる**。【TV は映像が映り、CD プレーヤーは曲が再生される】

となります。