● J2Kad06D「スーパークラスの参照」（入門編 P.209「クラスの継承と参照」）

J2Kad06D に Monster（名前：ピカチュウ）の各メソッドを呼び出す処理が作成されている。

① Monster を継承して FireMonster（名前：ヒトカゲ）と RockMonster（名前：カブト）を作成し、intro メソッドおよび各クラスで追加したメソッド（fire と defend）を呼び出す処理を追加せよ。
② FireMonster と RockMonster への参照のデータ型を Monster に変更し、動作確認せよ。

FireMonster クラスの仕様（Monster クラスを継承する）

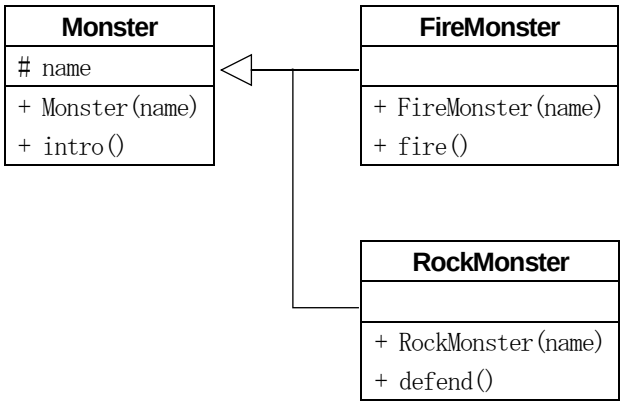
メソッド	仕様
FireMonster(String name)	Monster クラスのコンストラクタを呼び出す。引数 name をそのまま渡す。
void fire()	「～は炎をはいた！ゴオ～!!」（～は名前）と表示する。

RockMonster クラスの仕様（Monster クラスを継承する）

メソッド	仕様
RockMonster(String name)	Monster クラスのコンストラクタを呼び出す。引数 name をそのまま渡す。
void defend()	「～は防御している！ダメージを与えられない!!」（～は名前）と表示する。

①まで完成時の画面

```
おいらの名前はピカチュウ。
趣味は散歩。特技はどこでも眠れることだよ。

おいらの名前はヒトカゲ。
趣味は散歩。特技はどこでも眠れることだよ。
ヒトカゲは炎をはいた！ゴオ～！！

おいらの名前はカブト。
趣味は散歩。特技はどこでも眠れることだよ。
カブトは防御している！ダメージを与えられない！！
```



②まで完成時の画面

```
おいらの名前はピカチュウ。
趣味は散歩。特技はどこでも眠れることだよ。

おいらの名前はヒトカゲ。
趣味は散歩。特技はどこでも眠れることだよ。

おいらの名前はカブト。
趣味は散歩。特技はどこでも眠れることだよ。
```

スーパークラスの参照を使うとサブクラスのインスタンスも扱うことができる。ただしスーパークラスにないメソッドを呼び出すことはできない。

● J2Kad06C「ポリモーフィズム」（入門編 P.213「ポリモーフィズム」）

J2Kad06C に Monster（名前：ピカチュウ）が自己紹介する処理が作成されている。FireMonster と RockMonster に自己紹介（intro メソッド）を追加し、**Monster と同じ変数を使って**自己紹介する処理を作成せよ。

FireMonster クラスに追加するメソッド

メソッド	仕様
void intro()	Monster クラスの intro メソッドを呼び出したのち、「炎も出せるよ！」と表示する。

RockMonster クラスに追加するメソッド

メソッド	仕様
void intro()	Monster クラスの intro メソッドを呼び出したのち、「とても硬いぜ！」と表示する。

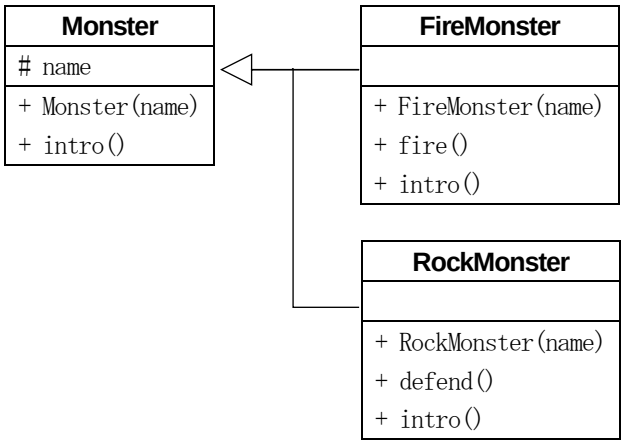
リスト１：ポリモーフィズム（ファイル「J2Kad06C.java」）

```
public class J2Kad06C {
    public static void main(String[] args) {
        // Monster
        Monster m = new Monster("ピカチュウ");
        m.intro();
        System.out.println();

        // FireMonster
        変数 m を使って FireMonster の自己紹介をする
        System.out.println();

        // RockMonster
        変数 m を使って RockMonster の自己紹介をする
    }
}
```

課題完成時の画面

```
おいらの名前はピカチュウ。
趣味は散歩。特技はどこでも眠れることだよ。

おいらの名前はヒトカゲ。
趣味は散歩。特技はどこでも眠れることだよ。
炎も出せるよ！

おいらの名前はカブト。
趣味は散歩。特技はどこでも眠れることだよ。
とても硬いぜ！
```

同じ参照（変数）を使って intro メソッドを呼び出しても
参照先が異なると動作も異なる。

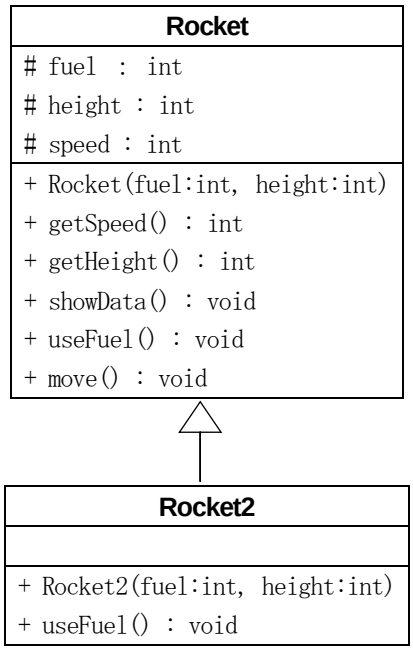


● J2Kad06B「月面着陸ゲーム！③（ロケット選択）」

Rocket クラス（旧型ロケット）を使った月面着陸ゲームの完成版が準備されている。Rocket2 クラス（新型ロケット）も準備されている。旧型ロケットと新型ロケットを選択して遊べるように修正せよ。

課題完成時の画面①（旧型ロケットを選択）

```
使用するロケットを選択してください（1：旧型、それ以外：新型）＞1
ロケットを作った！
―――――――――――――――――――――
［燃料］：15　　［落下速度］：0　　［高度］：300
―――――――――――――――――――――
逆噴射しますか？（1：する、それ以外：しない）＞0
　　⋮
　（中略）
　　⋮
―――――――――――――――――――――
［燃料］：1　　［落下速度］：10　　［高度］：5
―――――――――――――――――――――
逆噴射しますか？（1：する、それ以外：しない）＞1
燃料を使った！

―――――――――――――――――――――
［燃料］：0　　［落下速度］：9　　［高度］：-4
―――――――――――――――――――――
おめでとう！着陸成功！！
```

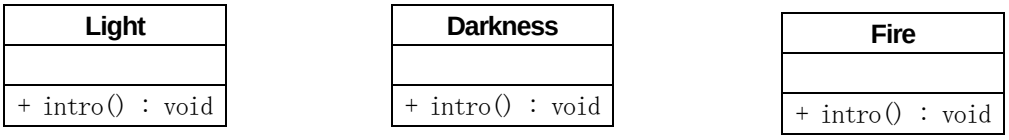


課題完成時の画面②（新型ロケットを選択）

```
使用するロケットを選択してください（1：旧型、それ以外：新型）＞0
ロケットを作った！
新型エンジンだ！
―――――――――――――――――――――
［燃料］：15　［落下速度］：0　　［高度］：300
―――――――――――――――――――――
逆噴射しますか？（1：する、それ以外：しない）＞0
　　⋮
　（中略）
　　⋮
―――――――――――――――――――――
［燃料］：6　　［落下速度］：9　　［高度］：1
―――――――――――――――――――――
逆噴射しますか？（1：する、それ以外：しない）＞1
燃料を使った！

―――――――――――――――――――――
［燃料］：5　　［落下速度］：6　　［高度］：-5
―――――――――――――――――――――
おめでとう！着陸成功！！
```

● J2Kad06A「妖精の召喚」

J2Kad06A に妖精を召喚する処理が作成されている。「0：召喚する」を選択すると光の妖精（Light）・闇の妖精（Darkness）・炎の妖精（Fire）のどれかを召喚して自己紹介させる。ポリモーフィズムを使って、自己紹介部分の処理（switch 文のところ）を簡略化せよ。



リスト１：妖精の召喚（J2Kad06A クラス）

```
public class J2Kad06A {
    public static void main(String[] args) {
            ⋮
        // 妖精の召喚
        int n = (int)(Math.random() * 3);
        switch(n) {
            default:
            case 0:
                Light light = new Light();
                light.intro();
                break;
            case 1:
                Darkness dark = new Darkness();
                dark.intro();
                break;
            case 2:
                Fire fire = new Fire();
                fire.intro();
                break;
        }
        System.out.println();
            ⋮
```

簡略化すること

課題完成時の画面

```
妖精を召喚して自己紹介させます！
どうしますか？（0：召喚する、-1：やめる）＞0
わたしは光の妖精！この者に祝福を！！

どうしますか？（0：召喚する、-1：やめる）＞0
わたしは闇の妖精だ！闇の力を思い知れ！！

どうしますか？（0：召喚する、-1：やめる）＞0
わたしは炎の妖精さ！炎の力は気まぐれなのさ！！

どうしますか？（0：召喚する、-1：やめる）＞-1
```

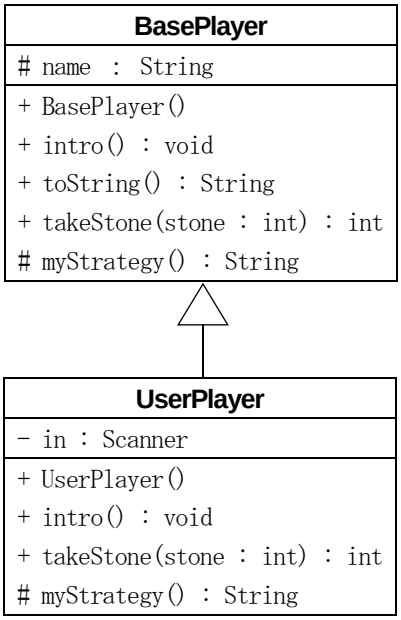
● J2Kad06S1「石取りゲームの簡略化」

「石取りゲーム」のプログラムが準備されている。20 個ある石から先手と後手、交互に 1～3 個ずつ石を取っていき、最後の 1 つを取った方が負けになる。先手は「あなた」が操作する UserPlayer、後手は乱数で取る石の数を決める BasePlayer（名前：CPU）となっている。ポリモーフィズムを使って石取りゲームの処理を簡略化せよ。

ヒント：先手と後手の処理をひとつにまとめる

課題完成時の画面

```
20 個ある石を交互に取っていきます。一度に取れる石の数は 1-3 個です。
最後の 1 つを取った方が負けです。

あなたの名前を入力してください>ECC

名前：ECC・・・あなたが操作するプレイヤーです。
名前：CPU・・・取る石の数を乱数で決めます。

残り 20 個：●●●●●●●●●●●●●●●●●●●●
ECC の番です。
何個取りますか？（1-3）>3
3 個取りました！

残り 17 個：●●●●●●●●●●●●●●●●●
CPU の番です。
3 個取りました！
      ⋮
   (中略)
      ⋮
残り 7 個：●●●●●●●
ECC の番です。
何個取りますか？（1-3）>2
2 個取りました！

残り 5 個：●●●●●
CPU の番です。
2 個取りました！

残り 3 個：●●●
ECC の番です。
何個取りますか？（1-3）>2
2 個取りました！

残り 1 個：●
CPU の番です。
1 個取りました！
CPU の負けです！
```



● J2Kad06S2「石取りゲーム完成！」※J2Kad06S1 をコピーして作成

J2Kad06S1 に「少し強い」CompPlayer を追加し、先手と後手を UserPlayer、BasePlayer、CompPlayer から選択できるようにせよ。

CompPlayer の仕様（BasePlayer を継承）

メソッド	仕様
コンストラクタ	name に「HAL」を代入する。
int takeStone(int stone)	stone が 1： 1 を返す。 stone が 2～4： stone-1 を返す。 stone が 6～8： stone-5 を返す。 それ以外： 乱数で 1～3 を返す（BasePlayer の takeStone と同じ）。
String myStrategy()	文字列「少し強いです！」を返す。

課題完成時の画面（先手：UserPlayer、後手：CompPlayer を選択した場合）

```
20 個ある石を交互に取っていきます。一度に取れる石の数は 1-3 個です。
最後の 1 つを取った方が負けです。

先手を選んでください（0：User、1：Base、2：Comp）＞0
あなたの名前を入力してください＞ECC
後手を選んでください（0：User、1：Base、2：Comp）＞2

名前：ECC・・・あなたが操作するプレイヤーです。
名前：HAL・・・少し強いです！

残り 20 個：●●●●●●●●●●●●●●●●●●●●
ECC の番です。
何個取りますか？（1-3）＞3
3 個取りました！
      ⋮
   （中略）
      ⋮
残り 5 個：●●●●●
ECC の番です。
何個取りますか？（1-3）＞1
1 個取りました！

残り 4 個：●●●●
HAL の番です。
3 個取りました！

残り 1 個：●
ECC の番です。
何個取りますか？（1-3）＞1
1 個取りました！
ECC の負けです！
```

先手も後手も UserPlayer を選ぶと人 VS 人の対戦になる。
先手も後手も BasePLayer または CompPlayer を選ぶとコンピュータ同士の対戦になる。

● J2Kad06X「最強！MasterPlayer！！」※J2Kad06S2 をコピーして作成

CompPlayer より強い MasterPlayer（名前：ECC）を作成し、選択できるようにせよ。

MasterPlayer の仕様（BasePlayer を継承）

メソッド	仕様
コンストラクタ	name に「ECC」を代入する。
int takeStone(int stone)	↓の表と CompPlayer のアルゴリズムを参考に各自で考えること。
String myStrategy()	文字列「最強です！！」を返す。

ヒント：
相手の番のときの石の数が 5 つになるようにすれば必ず勝てる（相手が 1 つ取れば自分は自分は 3 つ、2 つ取れば自分も 2 つ、3 つ取れば自分は 1 つというように合わせて 4 つになるように取れば、相手の番では必ず 1 つにできる。
↓
そのためには相手の番のときの石の数が 9 つになるようにする。
↓
そのためには…

相手の番のときの石の数	相手が取れる石の数	勝つために自分が取る石の数	残った石の数（相手の番）
⋮	⋮	⋮	⋮
9	1	3	5
	2	2	5
	3	1	5
5	1	3	1
	2	2	1
	3	1	1
1	1	←自分の勝ち	

課題完成時の画面（先手：MasterPlayer、後手：CompPlayer を選択した場合）

```
先手を選んでください（0：User、1：Base、2：Comp、3：Master）＞3
後手を選んでください（0：User、1：Base、2：Comp、3：Master）＞2

名前：ECC・・・最強です！！
名前：HAL・・・少し強いです！

残り 20 個：●●●●●●●●●●●●●●●●●●●●
ECC の番です。
3 個取りました！

残り 17 個：●●●●●●●●●●●●●●●●●
HAL の番です。
1 個取りました！

残り 16 個：●●●●●●●●●●●●●●●●
ECC の番です。
3 個取りました！

残り 13 個：●●●●●●●●●●●●●
HAL の番です。
1 個取りました！

残り 12 個：●●●●●●●●●●●●
ECC の番です。
3 個取りました！

残り 9 個：●●●●●●●●●
HAL の番です。
3 個取りました！

残り 6 個：●●●●●●
ECC の番です。
1 個取りました！

残り 5 個：●●●●●
HAL の番です。
3 個取りました！

残り 2 個：●●
ECC の番です。
1 個取りました！

残り 1 個：●
HAL の番です。
1 個取りました！
HAL の負けです！
```

MasterPlayer が先手の場合、
絶対に負けない。