# **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**

|  |
| --- |
| **Monster** |
| # name |
| + Monster(name)  + intro() |

|  |
| --- |
| **RockMonster** |
|  |
| + RockMonster(name)  + defend() |

|  |
| --- |
| **FireMonster** |
|  |
| + FireMonster(name)  + fire() |

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！

|  |
| --- |
| **Monster** |
| # name |
| + Monster(name)  + intro() |

|  |
| --- |
| **RockMonster** |
|  |
| + RockMonster(name)  + defend()  + intro() |

|  |
| --- |
| **FireMonster** |
|  |
| + FireMonster(name)  + fire()  + intro() |

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）**

|  |
| --- |
| **Rocket** |
| # fuel ： int  # height : int  # speed : int |
| + Rocket(fuel:int, height:int)  + getSpeed() : int  + getHeight() : int  + showData() : void  + useFuel() : void  + move() : void |

|  |
| --- |
| **Rocket2** |
|  |
| + Rocket2(fuel:int, height:int)  + useFuel() : void |

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。

|  |
| --- |
| **Fire** |
|  |
| + intro() : void |

|  |
| --- |
| **Darkness** |
|  |
| + intro() : void |

|  |
| --- |
| **Light** |
|  |
| + intro() : void |

**リスト1：妖精の召喚（J2Kad06Aクラス） 課題完成時の画面**

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();

…

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**

簡略化すること

# **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！

|  |
| --- |
| **BasePlayer** |
| # name ： String |
| + BasePlayer()  + intro() : void  + toString() : String  + takeStone(stone : int) : int  # myStrategy() : String |

|  |
| --- |
| **UserPlayer** |
| - in : Scanner |
| + UserPlayer()  + intro() : void  + takeStone(stone : int) : int  # myStrategy() : String |

# **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）**

MasterPlayerが先手の場合、

絶対に負けない。

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！