● J2Kad06D「スーパークラスの参照」

J2Kad06D クラスに Monster クラス（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 クラスを継承して FireMonster クラス（名前：ヒトカゲ）と RockMonster クラス（名前：カブト）を作成し、それぞれのクラスの各メソッド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 クラスに変更し、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 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。
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）

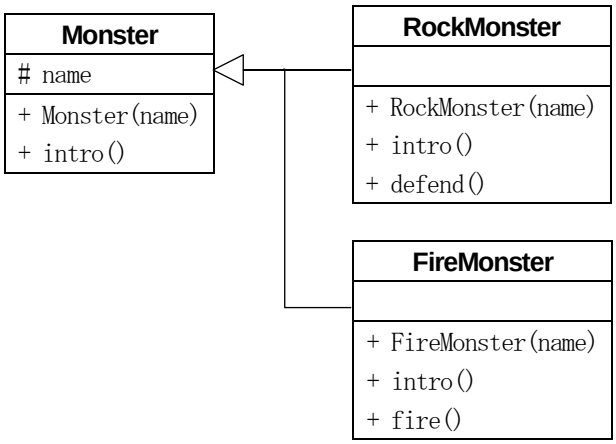
メソッド	仕様
RockMonster(String name)	Monster クラスのコンストラクタを呼び出す。引数 name をそのまま渡す。
void intro()	Monster クラスの intro メソッドを呼び出したのち、「とても硬いぜ！」と表示する。
void defend()	「～は防御している！ダメージを与えられない!!」（～は名前）と表示する。

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。
炎も出せるよ！
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。
とても硬いぜ！
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。
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！
```

● J2Kad06C「ポリモーフィズム」

Monster クラスの配列（要素数 3）を使って、Monster クラス（名前：ピカチュウ）・FireMonster クラス（名前：ヒトカゲ）・RockMonster クラス（名前：カブト）が自己紹介する処理を作成せよ。なお、自己紹介は for 文を使って作成すること。

```
Monster[] m = new Monster[3];     // 0 に Monster、1 に FireMonster、2 に RockMonster を設定する
```

課題完成時の画面

（J2Kad06D の**②まで完成時の画面**と同じ）

● J2Kad06B「モンスターを探せ！」

モンスターを探す処理を作成せよ。「0：探す」を選択すると Monster・FireMonster・RockMonster のどれか（乱数で決定）が自己紹介するものとする。

リスト 1「モンスターを探せ！」（J2Kad06B クラス）

```
Import java.util.Scanner;

public class J2Kad06B {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        System.out.println("モンスターを探します！");

        while(true) {
            System.out.print("どうしますか？（0：探す、-1：やめる）>");
            int cmd = in.nextInt();
            if (cmd < 0) break;

            // Monster の自己紹介
            乱数で決定した Monster に自己紹介させる
            System.out.println();
        }
    }
}
```

課題完成時の画面

```
モンスターを探します！
どうしますか？（0：探す、-1：やめる）>0
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！

どうしますか？（0：探す、-1：やめる）>0
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！

どうしますか？（0：探す、-1：やめる）>0
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。

どうしますか？（0：探す、-1：やめる）>-1
```

● J2Kad06A「妖精の召喚」

光の妖精（Light）・闇の妖精（Darkness）・炎の妖精（Fire）が自己紹介するクラスが準備されている。J2Kad06Bを参考に妖精（Light・Darkness・Fire）を召喚する処理を作成せよ。なお、ポリモーフィズムが使えるように必要であればクラス（例えばFairyクラス）を追加すること。

Light
+ intro() : void

Darkness
+ intro() : void

Fire
+ intro() : void

課題完成時の画面

```
妖精を召喚して自己紹介させます！
誰を召喚しますか？（0：光の妖精、1：闇の妖精、2：炎の妖精、-1：やめる）＞0
わたしは光の妖精！この者に祝福を！！

誰を召喚しますか？（0：光の妖精、1：闇の妖精、2：炎の妖精、-1：やめる）＞1
わたしは闇の妖精だ！闇の力を思い知れ！！

誰を召喚しますか？（0：光の妖精、1：闇の妖精、2：炎の妖精、-1：やめる）＞2
わたしは炎の妖精さ！炎の力は気まぐれなのさ！！

誰を召喚しますか？（0：光の妖精、1：闇の妖精、2：炎の妖精、-1：やめる）＞-1
```

● J2Kad06S「石取りゲーム完成！」

J2Kad05S 完成版相当のプログラムが準備されている。BasePlayer・UserPlayer・CompPlayer だけでなく MasterPlayer も作成済みだ！

① ポリモーフィズムを使って先手と後手の処理をひとつにまとめよ（J2Kad05X とほぼ同じ、ただし「～の番です」と表示するときに「先手：」「後手：」を付けること）。
② 先手と後手を 0：BasePlayer、1：UserPlayer、2：CompPlayer、3：MasterPlayer から選択できるようにせよ。

課題完成時の画面（先手：UserPlayer、後手：MasterPlayer 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：Base、1：User、2：Comp、3：Master）＞1
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：Base、1：User、2：Comp、3：Master）＞3

名前：ECC・・・あなたが操作するプレイヤーです。
名前：Master・・・最強です！！

残り 20 個：●●●●●●●●●●●●●●●●●●●●
先手：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：●●●●●●●●●●●●●●●●●
後手：Master の番です。
1 個取りました！
      ⋮
   （中略）
      ⋮
残り 4 個：●●●●
先手：ECC の番です。
何個取りますか？（1-3）＞3
3 個取りました！

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

先手も後手も、どのプレイヤーを選んでも、同じコードで動作する。

● J2Kad06X「直線描画のアルゴリズム」※実行画面を別ウインドウにしてください

画面に画像を描画するための Canvas クラスが準備されている。Canvas クラスに直線描画のメソッド（drawLine メソッド）を追加し、三角形の描画を行え。なお、drawLine メソッド内において変数を使う場合は、データ型は int 型のみ OK とする。

ヒント：「ブレゼンハムのアルゴリズム」「直線描画のアルゴリズム」などで検索すること。

Canvas クラスの仕様（drawLine メソッドを追加する）

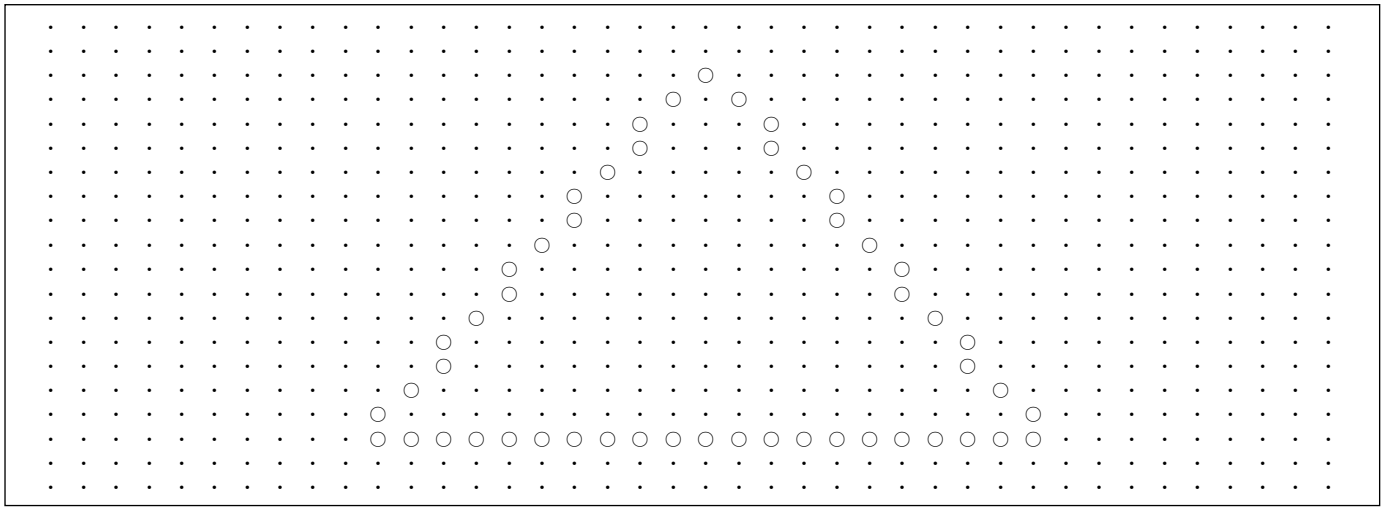
メンバ	仕様
boolean[][] pixel	仮想画面に該当する 2 次元配列。値が true のとき「○」false のとき「・」を表示する。
Canvas(int width, int height)	コンストラクタ。高さ height×横幅 width で pixel を生成する（初期値は false)。
void show()	仮想画面 pixel を画面に表示する。
boolean inBound(int x, int y)	座標(x, y)が pixel の範囲内なら true、範囲外なら false を返す。
void set(int x, int y)	座標(x, y)が pixel の範囲内なら点を打つ「○」。
void reset(int x, int y)	座標(x, y)が pixel の範囲内なら点を消す「・」。
void drawLine(int x1, int y1, int x2, int y2)	座標(x1, y1)から座標(x2, y2)まで直線を引く（「○」を打つ)。 ただし pixel の範囲外には打たない。**※変数を使う場合は int 型のみ OK。**

main メソッドの仕様（三角形の描画）

① drawLine メソッドを使って三角形を描画する。三角形の頂点は以下の通り。

(10, 17)、(30, 17)、(20, 2)

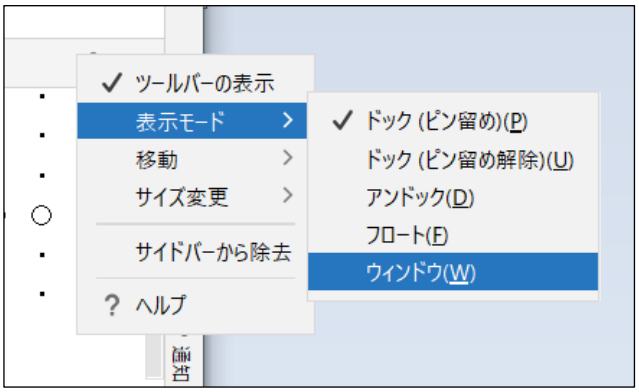
課題完成時の画面（実装するアルゴリズムによっては 1 マス程度「○」がずれることもあり）



● 実行画面を別ウインドウにする方法

① プログラムを実行する。
② 実行画面右上の⚙をクリックする。

③ ポップアップメニューが表示されるので、[表示モード]→[ウインドウ]を選択する。



④ 実行画面が別ウインドウになるので好みの大きさ（例えば全画面表示）に設定する。