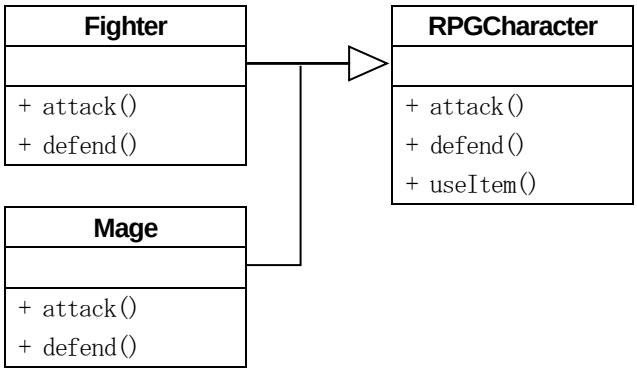
● J2Kad07D「ポリモーフィズムの復習」

ECC がゲームを作ることになった！その名も「ドラゴンファンタジー」。戦士と魔法使いがドラゴン退治に行くという画期的なストーリーだ。とりあえず戦士を表す Fighter クラスと魔法使いを表す Mage クラス、および動作チェックプログラム（J2Kad07D クラス）を業者に依頼して作ってもらったが、今後様々なジョブを追加するととんでもないことになるシロモノだった！ポリモーフィズムを使って処理を簡略化せよ。



スーパークラスとして
RPGCharacter クラスを作成する。

リスト１：業者のプログラム（J2Kad07D クラス）

```
public class J2Kad07D {
    public static void main(String[] args) {
            ⋮
        while(true) {
            System.out.print("ジョブを選んでください（0：戦士、1：魔法使い、-1：終了）>");
            int job = in.nextInt();
            if (job < 0) break;
            switch(job) {
                default:
                case 0: // Fighter
                    Fighter f = new Fighter();
                    f.attack();
                    f.defend();
                    f.useItem();
                    break;
                case 1: // Mage
                    Mage m = new Mage();
                    m.attack();
                    m.defend();
                    m.useItem();
                    break;
            }
            System.out.println();
        }
    }
}
```

恐ろしい処理

課題完成時の画面

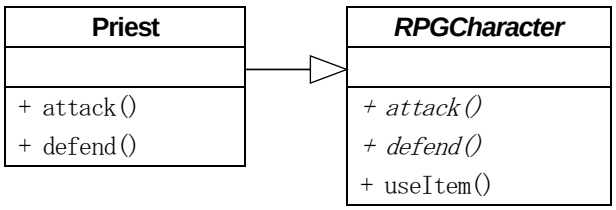
```
ジョブを選んでください（0：戦士、1：魔法使い、-1：終了）>0
武器で攻撃します！
盾で防御します！
何かのアイテムを使います！

ジョブを選んでください（0：戦士、1：魔法使い、-1：終了）>1
攻撃の魔法を唱えます！
防御の魔法を唱えます！
何かのアイテムを使います！

ジョブを選んでください（0：戦士、1：魔法使い、-1：終了）>-1
```

● J2Kad07C「抽象メソッドと抽象クラス」※J2Kad07D をコピーして作成

（入門編 P.240「実体のない抽象メソッド」、P.237「インスタンスを作れないクラス」）

J2Kad07D で追加した RPGCharacter クラスの attack メソッドと defend メソッドにはそもそもプログラムコードが必要ないことがわかった。

① RPGCharacter クラスの attack メソッドと defend メソッドを抽象メソッドにせよ。
② 新たに僧侶（Priest クラス）を追加し、Priest クラスの動作チェックも追加せよ。動作チェックの手順（メソッドを呼び出す順序）は戦士・魔法使いと同じとする。



抽象クラスおよび抽象メソッドは
*斜体*で記述する

Priest クラスの仕様（RPGCharacter クラスを継承する）

メソッド	仕様
void attack()	「みんなを励まします！」と表示する。
void defend()	「神に祈ります！」と表示する。

課題完成時の画面（戦士と魔法使いは J2Kad07D と同じ）

```
ジョブを選んでください（0：戦士、1：魔法使い、2：僧侶、-1：終了）>2
みんなを励まします！
神に祈ります！
何かのアイテムを使います！
```

● J2Kad07B「フック」※J2Kad07C をコピーして作成

僧侶（Priest クラス）独自の行動として「治療する」（heal メソッド）を追加することになった。また新たに盗賊（Thief クラス）も追加することになった。

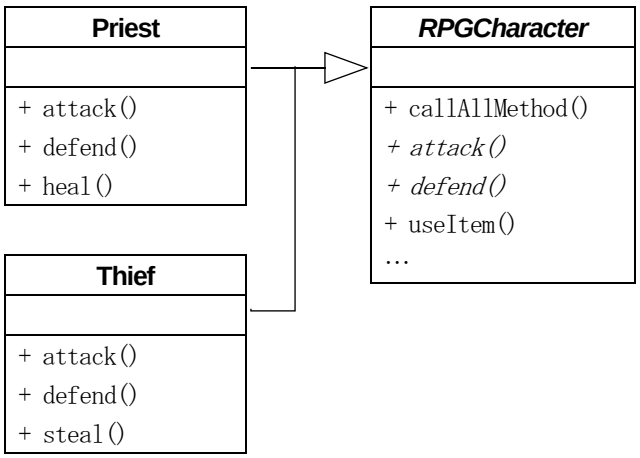
① Priest に heal メソッドを追加し、動作チェックできるようにせよ。
② 動作チェックの手順そのものを RPGCharacter で実装し（checkAllAction メソッド）、main メソッドからは checkAllAction メソッドのみを呼び出すように修正せよ。
③ Thief を追加し、Thief も選択できるようにせよ。なお、Thief のみ新たに「盗む」（steal メソッド）が追加されるものとする。

Priest クラスの追加メソッド

メンバ	仕様
void heal()	「みんなの傷を治します！」と表示する。

Thief クラスの仕様（RPGCharacter クラスを継承する）

メンバ	仕様
void attack()	「素早く背後にまわって不意打ちします！」と表示する。
void defend()	「素早く攻撃をかわします！」と表示する。
void steal()	「何かを盗みます！」と表示する。



attack、defend、useItem などの呼び出しは callAllMethod から行う。
Priest の heal や Thief の steal を呼び出すためには、RPGCharacter 側に工夫が必要。

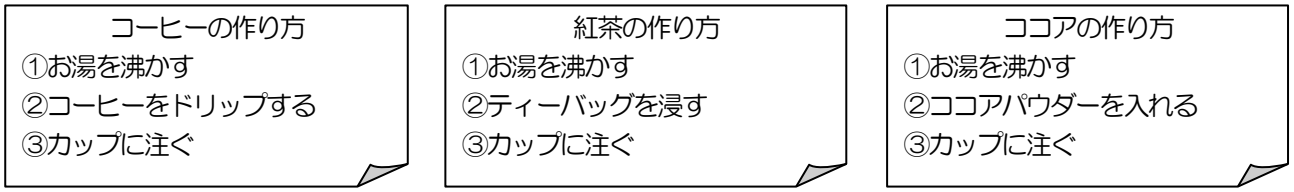
課題完成時の画面（戦士と魔法使いは J2Kad07D と同じ）

```
ジョブを選んでください（0：戦士、1：魔法使い、2：僧侶、3：盗賊、-1：終了）＞2
みんなを励まします！
神に祈ります！
何かのアイテムを使います！
みんなの傷を治します！

ジョブを選んでください（0：戦士、1：魔法使い、2：僧侶、3：盗賊、-1：終了）＞3
素早く背後にまわって不意打ちします！
素早く攻撃をかわします！
何かのアイテムを使います！
何かを盗みます！
```

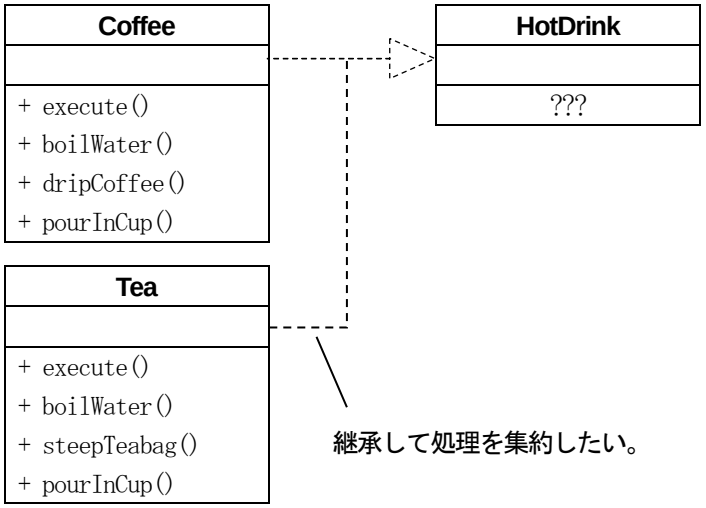
● J2Kad07A「ECC コーヒー再び！（Template Method）」

世界に羽ばたく ECC がカフェを経営することになった。名付けて「ECC コーヒー」。メニューはコーヒーと紅茶。ところがドリンク生成プログラムを業者に依頼して作ってもらったところ、とんでもないシロモノになってしまった！これでは新しいメニューの追加が面倒だ！

① スーパークラス（HotDrink）を導入して main メソッドを簡略化せよ。
② Coffee クラスと Tea クラスも簡略化せよ。
③ ココア（Cocoa）を追加せよ。



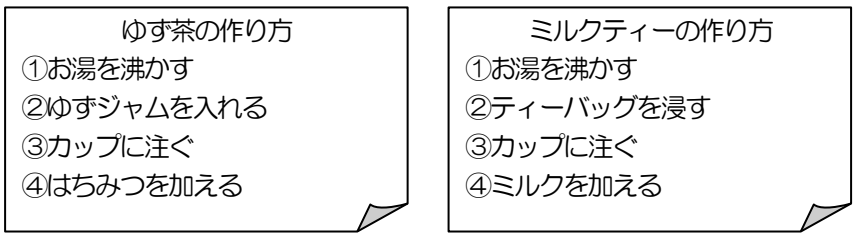
ヒント：
サブクラスのメソッドをスーパークラスへ集約するためにはメソッドの仕様（戻り値、名前、引数）を同じにする必要がある。



課題完成時の画面
（J2Kad07S を参照）

● J2Kad07S「ECC コーヒー完成！」※J2Kad07A をコピーして作成

新メニューとして「3：ゆず茶」（Yuzu クラス）と「4：ミルクティー」（MilkTea クラス）を追加せよ。



課題完成時の画面（J2Kad07A＆J2Kad07S）※「3：ゆず茶」と「4：ミルクティー」は J2Kad07S

```
ECC コーヒーへようこそ！
門外不出のレシピで作るから、おいしいよ！！

ご注文は？（0：コーヒー、1；紅茶、2：ココア、3：ゆず茶、4：ミルクティー、-1：店を出る）＞0
お湯を沸かしました！
コーヒーをドリップしました！
カップに注ぎました！
お待たせしました！ごゆっくりどうぞ！

ご注文は？（0：コーヒー、1；紅茶、2：ココア、3：ゆず茶、4：ミルクティー、-1：店を出る）＞1
お湯を沸かしました！
ティーバッグを浸しました！
カップに注ぎました！
お待たせしました！ごゆっくりどうぞ！

ご注文は？（0：コーヒー、1；紅茶、2：ココア、3：ゆず茶、4：ミルクティー、-1：店を出る）＞2
お湯を沸かしました！
ココアパウダーを入れました！
カップに注ぎました！
お待たせしました！ごゆっくりどうぞ！

ご注文は？（0：コーヒー、1；紅茶、2：ココア、3：ゆず茶、4：ミルクティー、-1：店を出る）＞3
お湯を沸かしました！
ゆずジャムを入れました！
カップに注ぎました！
はちみつを加えました！
お待たせしました！ごゆっくりどうぞ！

ご注文は？（0：コーヒー、1；紅茶、2：ココア、3：ゆず茶、4：ミルクティー、-1：店を出る）＞4
お湯を沸かしました！
ティーバッグを浸しました！
カップに注ぎました！
ミルクを加えました！
お待たせしました！ごゆっくりどうぞ！
```

● J2Kad07X「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（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）

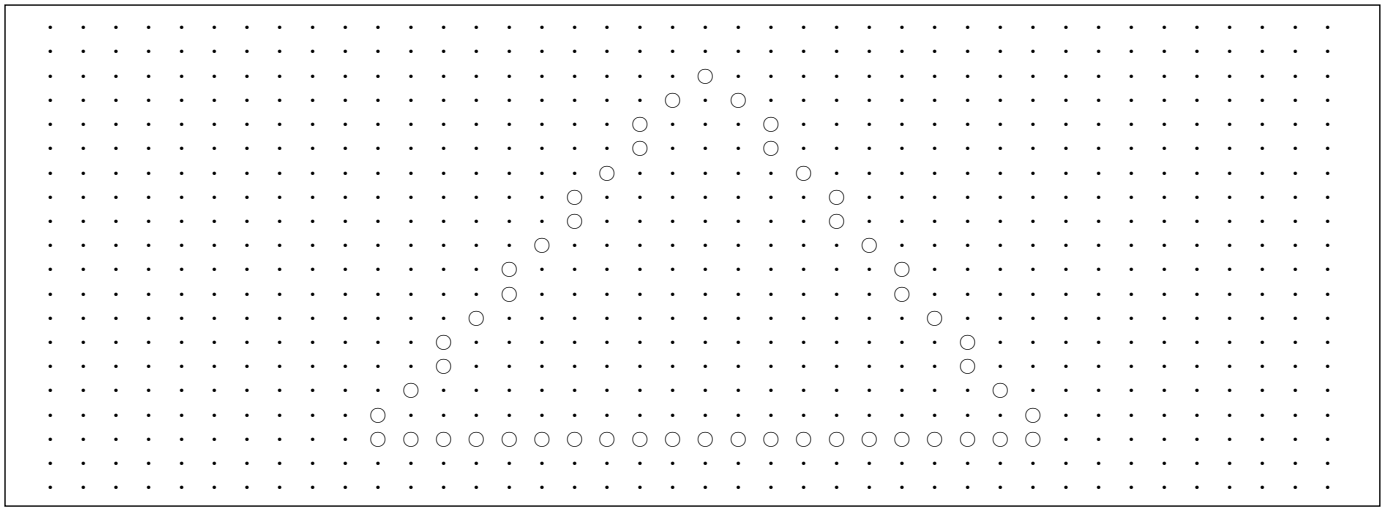
メンバ	仕様
boolean[][] pixel	仮想画面に該当する 2 次元配列。値が true のとき「○」false のとき「・」を表示する。
Canvas(int width, int height)	コンストラクタ。高さ height×横幅 width で pixel を生成する（初期値は false)。
void show()	仮想画面 pixel を画面に表示する。
boolean inBound(int x, int y)	座標(x, y)が pixel の範囲内なら true、範囲外なら false を返す。
void set(int x, int y)	座標(x, y)が pixel 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)。 **※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)

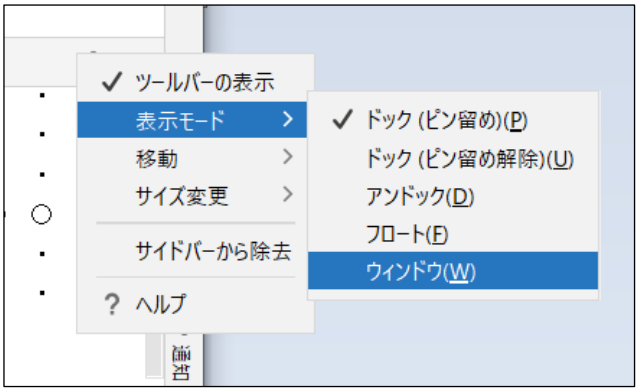
課題完成時の画面（実装するアルゴリズムによっては 1 マス程度「○」がずれることもあり）



● 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。