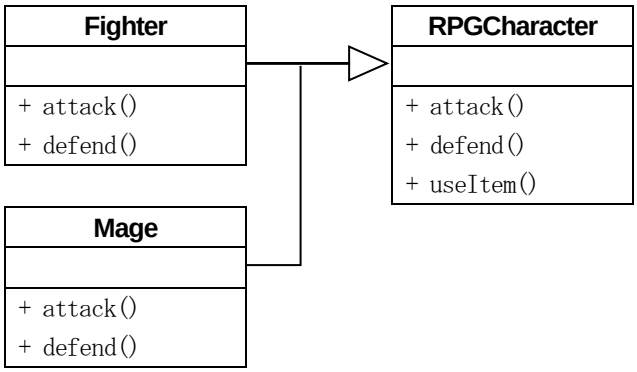
● J2Kad07「RPG キャラクター」

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クラス（Fighter）と魔法使いを表すクラス（Mage）、および動作チェックプログラム（J2Kad07 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！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

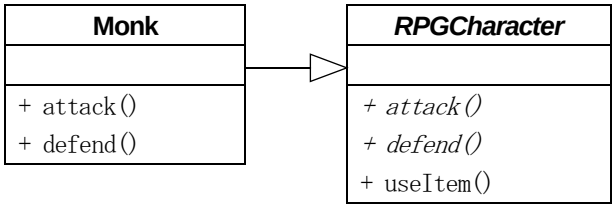
課題完成時の画面

```
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 の main メソッドをコピーすること

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 Monk クラスを追加し、Monk 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

Monk クラスの仕様（RPGCharacter クラスを継承する）

メンバ	仕様
attack メソッド	「素手で戦います！」と表示する。
defend メソッド	「素手で防御します！」と表示する。

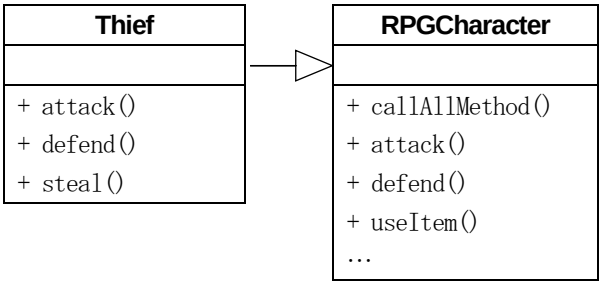
課題完成時の画面（戦士と魔法使いは J2Kad07D と同じ）

```
ジョブを選んでください（0：戦士、1：魔法使い、2：モンク、-1：終了）>2
素手で戦います！
素手で守ります！
何かのアイテムを使います！
```

● J2Kad07B「Template Method」※J2Kad07C の main メソッドをコピーすること

J2Kad07C において戦士・魔法使い・モンクともに動作チェックの手順（メソッドを呼び出す順序）は同じであった。

① 動作チェックの手順そのものを RPGCharacter クラスで実装し（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（Template Method）。
② 新たに盗賊（Thief クラス）を追加し、Thief クラスも選択できるようにせよ。なお、Thief クラスのみ新たに「盗む」（steal メソッド）が追加されるものとする。



attack、defend、useItem などの呼び出しは callAllMethod に集約する。
Thief の steal を呼び出すためには、RPGCharacter 側に工夫が必要。

Thief クラスの仕様（RPGCharacter クラスを継承する）

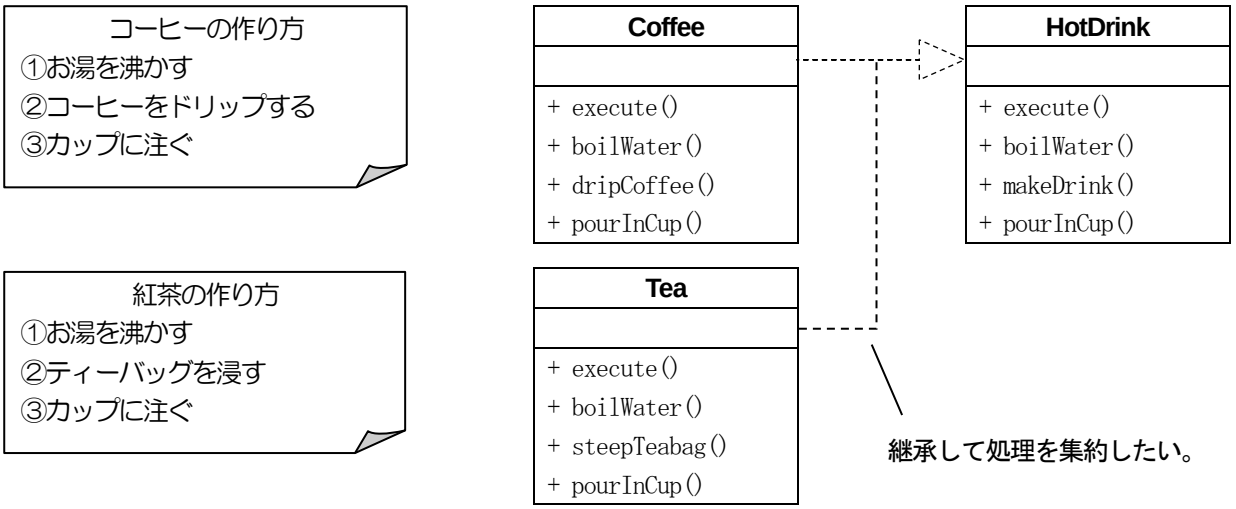
メンバ	仕様
attack メソッド	「素早く背後にまわって不意打ちします！」と表示する。
defend メソッド	「素早く攻撃をかわします！」と表示する。
steal メソッド	「何かを盗みます！」と表示する。

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：モンク、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！」

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、（J2Kad07D に続いて）とんでもないシロモノになってしまった（どこがとんでもないかは各自で考えること）！これでは新しいメニューの追加が面倒だ！スーパークラス（HotDrink）を追加して処理を簡略化せよ。



ヒント：
・コーヒー・紅茶ともに作り方の手順は同じだが、それぞれのクラスに手順に該当するメソッドがある。
・HotDrink をスーパークラスにして処理を集約したいが、Coffee と Tea でメソッドの仕様（名前や引数）が異なっている。

課題完成時の画面

```
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、-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、-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、-1：店を出る）＞-1
ありがとうございました！
```

● J2Kad07S「ECC コーヒー完成！」※J2Kad07A の main メソッドをコピーすること

新メニューとして「2：ココア」（Cocoa クラス）と「3：ゆず茶」（Yuzu クラス）を追加せよ。

ココアの作り方
①お湯を沸かす
②ココアパウダーを入れる
③カップに注ぐ

ゆず茶の作り方
①お湯を沸かす
②ゆずジャムを入れる
③カップに注ぐ
④はちみつを加える

課題完成時の画面

```
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、-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、-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、-1：店を出る）＞-1
ありがとうございました！
```

● J2Kad07X「円描画のアルゴリズム」

J2Kad06X 完成版相当の Canvas クラスが準備されている。Canvas クラスに円描画のメソッド（drawCircle）を追加し、円の描画を行え。なお、drawCircle 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：「ミッチェナーのアルゴリズム」「円描画のアルゴリズム」などで検索すること。

Canvas クラスに追加するメソッド

メンバ	仕様
`void drawCircle(int x, int y, int r);`	中心座標(x, y)、半径 r の円を描画する（「○」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（円の描画）

① drawCircle メソッドを使って円を描画する。中心座標と半径は以下の通り。

中心座標：(20, 10)、半径：8

課題完成時の画面（実装するアルゴリズムによっては微妙に形が異なることもあり）