基本情報技術者試験　平成 29 年春期　午後
問 11　ソフトウェア開発(Java)「料金プランの比較」 解説

クラス構成

TierTable クラス	段階的に変化する値のペアをテーブルとして表現するための抽象クラス
TierdRateTable クラス	料金表を表すクラス（TierTable クラスを継承）
RatePlan クラス	料金プランを表すクラス
Main クラス	プログラムを実行するためのクラス
DiscountTable クラス	割引率を表すクラス（TierTable クラスを継承）
DiscountPlan クラス	割引プランを表すクラス（RatePlan クラスを継承）

TierTable クラス

抽象クラス*1	TierTable	電力量と料金単価など値のペアをテーブルとして表現するための抽象クラス。 料金表や割引率の元となるクラス。
フィールド	double[ ][ ] pairs	電力量と料金単価などの値のペアを格納する配列。
コンストラクタ	TierTable(double... tiers)	引数 tiers から数値 2 個ずつをペアとして取り出し、フィールドの pairs に格納する。 引数 tiers の長さが奇数のときは、 IllegalArgumentException を投げる。 引数 tiers は double 型の可変長配列*3 で電力量の下限値と料金単価が格納されている。
抽象メソッド*2	double map(double amount)	引数で与えられた数値を別の数値に変換して返す抽象メソッド。 電力量から電力料金を求めるメソッドの元となるメソッド。

*1 抽象クラスとは抽象メソッド（処理の記述がないメソッド）を 1 つでも持っているクラスのこと。

*2 抽象メソッドとは、処理の記述がなく、メソッドの定義（戻り値の型、メソッド名、引数）のみを定義する。
実際の処理は継承したクラスでオーバーライドする。

*3 可変長配列とは配列の要素に 0 個以上任意の個数を指定可能な配列 「データ型. . .　配列名」で表す。
double... tiers は double 型の値を 0 個以上受け取ることができ、値は配列の要素として格納される。

〔プログラム1〕

```
abstract class TierTable {
    // フィールド　料金表(電力量下限値と料金単価のペア)を格納する配列
    final double[ ][ ] pairs;

    //コンストラクタ
    TierTable(double... tiers) {
        // 配列のサイズ/2 の余りが 1(つまり奇数)の場合
        // (電力量の下限値と料金単価のペアで格納されるため、偶数が正しいはず)
        if (tiers.length % 2 == 1) {
            // 例外を送出
            throw new IllegalArgumentException("不正な長さ:" + tiers.length);
        }
        // 料金表を格納する二次元配列を生成
        double[][] a = new double[tiers.length / 2][];
        // tiers(double の 1 次元配列)を double の 2 次元配列 a に格納
        for (int i = 0; i < tiers.length; i += 2) {
            // 電力量の下限値と料金単価のペア毎に配列を分割し、a の要素に格納
            a[ i / 2 ] = new double[ ] { tiers[i], tiers[i + 1] };
        }
        // 配列 a をフィールド pairs にコピー
        this.pairs = a;
    }

    //引数で与えられた数値を別の数値に
    //変換して返す抽象メソッド
    abstract double map(double amount);
}
```

a

b

ヒント：問題文「(1)抽象クラス TierTable は・・・」

抽象クラスの場合は、class の前に abstract 修飾子が必要

実行時の tires の値
料金プラン A の場合

0	19.62	120	26.10	300	30.12

料金プラン B の場合

0	18.17	120	24.17	300	27.77

tiers[0],[2],[4]は電力量の下限値
tiers[1],[3],[5]は電力量に対応する料金単価

ヒント：問題文「(1)① 可変長の引数で与えられた double の数値 2 個ずつをペアとして配列にし，更にその配列を要素とする配列を生成し，フィールド pairs に保持する」

1 行×6 列の tiers のデータを
3 行×2 列の a にコピーする
teirs[0]と[1]を a[0]に
teirs[2]と[3]を a[1]に
teirs[4]と[5]を a[2]に
コピーすればよい

i の初期値は 0、
i < 6 の条件を満たす間繰り返し、
処理後に i は 2 ずつ増加する

実行時の a(料金表)の値

プラン A の場合

0	19.62
120	26.10
300	30.12

プラン B の場合

0	18.17
120	24.17
300	27.77

a[0][0],[1][0],[2][0]は電力量の下限値
a[0][1],[1][1],[2][1]は電力量に対応する料金単価

抽象メソッドの場合、abstract が必要
処理は継承したサブクラス側で記述する

TierdRateTable クラス

クラス	TierdRateTable	料金表を表すクラス　TierTable クラスを継承。
コンストラクタ	TierdRateTable (double... tiers)	スーパークラス(TierTable　クラス)のコンストラクタを呼びだす。
メソッド	double map(double amount)	スーパークラス(TierTable クラス)の map メソッドをオーバーライドする。 引数の電力量から電力量料金を計算し、その値を返す。

〔プログラム2〕

```
class TierdRateTable extends TierTable {
        // コンストラクタ
        TierdRateTable(double... tiers) {
                //スーパークラス(TierTable クラス)のコンストラクタを実行
                super(tiers);
        }
         //引数の電力量から電力量料金を計算し、その値を返す
        double map(double amount) {
                // 電力料金を初期化
                double charge = 0;
                // スーパークラス(TierTable クラス)の
                // pairs(料金表)のサイズ分(3 回)繰り返す
                for (int i = 0; i < pairs.length; i++) {
                        // 引数の電力量が pairs の i+1 の行の電力量の下限値より多い場合
                        //つまり引数の電力量が pairs の i 行の上限値より多い(i 行に該当しない)
                        if (i + 1 < pairs.length && amount > pairs[i + 1][0]) {
                                //電力料金＋
                                //(i+1 の行の電力量の下限値- i 行の電力量の下限値)*料金単価
                                charge += (pairs[i + 1][0] - pairs[i][0] ) * pairs[i][1];
                                           (c)             (d)
                        } else {// 引数の電力量が pairs の i の行の電力量の上限値に収まる場合
                                //(引数の電力量- i の行の電力量の下限値)*料金単価
                                charge += (amount - pairs[i][0]) * pairs[i][1];
                                break;
                        }
                }
                // 電力料金を返す
                return charge;
        }
}
```

0	19.62
120	26.10
300	30.12

pairs[0][0],[1][0],[2][0]には電力量の下限値が格納されている。pairs[0][0]は電力量 0～120 を表しているため **pairs[0][0]に該当するかを判断するためには、引数の電力量が pairs[1][0](120)より大きいかどうかで判定する。**

①

②

0	19.62
120	26.10
300	30.12

プラン A で amount(電力量)が 120 の場合
i=0 の時②が実行され、charge=(120-0)*19.62=2354.4 を返す
プラン A で amount(電力量)が 313 の場合
i=0 の時①が実行され、charge=(120-0)*19.62=2354.4
i=1 の時①が実行され、charge=2354.4+(300-120)*26.10=7052.4
i=2 の時②が実行され、charge=7052.4+(313-300)*30.12=7443.96
を返す

RatePlan クラス

クラス	RatePlan	料金プランを表すクラス
フィールド	private final String name	料金プラン名
	private final double basicCharge	基本料金
	private final TierTable pricingTiers	料金表
コンストラクタ	RatePlan(String name, double basicCharge, TierTable pricingTiers)	引数で指定された料金プラン名、基本料金、料金表のインスタンスを生成。
メソッド	String getName()	料金プラン名を返す。
	int getPrice(double amount)	引数の電力量から電力量料金を計算し、基本料金との合計を型 int の数値で返す。このとき，小数点以下は切り捨てられる。

〔プログラム3〕

```
class RatePlan {
        //料金プラン名
        private final String name;
        //基本料金
        private final double basicCharge;
        //料金単価の階層
        private final TierTable pricingTiers;

        //コンストラクタ（引数の値をフィールドに設定）
        RatePlan(String name,double basicCharge,TierTable pricingTiers ){
                this.name = name;
                this.basicCharge = basicCharge;
                this.pricingTiers = pricingTiers;
        }
        //プラン名を返す
        public String getName() {
                return name;
        }
        //電力量から電力量料金を計算し、基本料金との合計を数値で返す。
        //このとき，小数点以下は切り捨てられる。
        int getPrice(double amount){
                //フィールド pricingTiers の電力量に応じた電力料金を計算するメソッドを実行し、
                //基本料金を加算した値を返す
                return (int)(basicCharge + pricingTiers.map(amount));
        }
}
```

Main クラス

クラス	Main	プログラムを実行するためのクラス
メソッド	public static void main(String[] args)	表 1 のプラン A 及びプラン B を表す RatePlan のインスタンスをそれぞれ生成し、電力量が 543.0kWh のときの電気料金を比較する。

〔プログラム4〕

```
public class Main{
        public static void main(String[] args){
                //プラン A の料金表を作成
                RatePlan planA = new RatePlan("プラン A",1123.30,
                                        new TierdRateTable(0,19.62,120,26.10,300,30.12));
                //プラン B の料金表を作成
                RatePlan planB = new RatePlan("プラン B",1040.10,
                                        new TierdRateTable(0,18.17,120,24.17,300,27.77));
                //電力量を 543.0 に指定
                double amount = 543.0;
                //プラン A の料金からプラン B の料金を減算
                int d = planA.getPrice(amount) - planB.getPrice(amount);
                //d が負の値（プラン A のほうが安い）
                if(d < 0 ){
                        System.out.printf("%s が%d 円安い%n",planA.getName(),-d);
                //d が正の値（プラン B のほうが安い）
                }else if(d > 0){
                        System.out.printf("%s が%d 円安い%n",planB.getName(),d);
                }else{
                        System.out.println("両プランで同額");
                }
        }
}
```

%s に planA.getName()で取得した料金プラン名を、%d にはプラン A とプラン B の差額を埋め込んで文字列を生成して出力

DiscountTable クラス

クラス	DiscountTable	割引率を表すクラス TierTable クラスを継承。
コンストラクタ	DiscountTable (double... tiers)	スーパークラス（TierTable クラス）のコンストラクタを呼びだす。
メソッド	double map(double amount)	引数の電気料金から割引率を求め，その値を返す。割引率は小数で与えるものとする。例えば，1%は 0.01 である。

〔プログラム5〕

```
class DiscountTable extends TierTable{
        //コンストラクタ
        public DiscountTable(double... tiers) {
                // スーパークラス（TierTable クラス）のコンストラクタを実行
                super(tiers);
        }

        //引数の電気料金から割引率を求める
        double map(double amount) {
                //割引率テーブルの最後から検索
                for(int i = pairs.length -1; i>= 0; i--){
                        //引数の電気料金のほうが割引率テーブルの電気料金より高い
                        if(amount > pairs[i][0]){
                                //電気料金に応じた割引率を整数で返す
                                return pairs[i][1];
                        }
                }
                //電気料金が割引率テーブルに該当しない（負の値の場合）例外を送出
                throw new IllegalArgumentException("amount = "+amount);
        }
}
```

割引率テーブルの例

料金	割引
0	0.01
5000	0.03
8000	0.05

表 2 から判断すると割引率テーブルには下限値が格納されている。電気料金が 10,000 の場合、テーブルの先頭から「下限値より大きい」条件で検索すると、1 行目の 0 が該当してしまうため、テーブルの最後から検索すると、8,000 以上の 3 行目が該当し、0.05 が返される。

DiscountPlan クラス

クラス	DiscountPlan	割引率を表すクラス RatePlan クラスを継承。
フィールド	private final TierTable discountTiers	割引率テーブル
コンストラクタ	DiscountPlan (String name, double basicCharge, TierTable pricingTiers, TierTable discountTiers)	引数の料金プラン名、基本料金、料金表、割引表をフィールドに設定。
メソッド	int getPrice(double amount)	割引を適用した金額を電気料金として返す。

〔プログラム6〕

```
class DiscountPlan extends RatePlan{
        //割引率テーブル
        private final TierTable discountTiers;

        // コンストラクタ
        public DiscountPlan(String name, double basicCharge, TierTable pricingTiers, TierTable discountTiers) {
                //スーパークラス（RatePlan クラス）のコンストラクタを実行
                super(name, basicCharge, pricingTiers);
                //引数の割引率テーブルをフィールドに設定
                this.discountTiers = discountTiers;
        }
        //割引を適用した金額を電気料金として返す
        int getPrice(double amount) {
                //スーパークラス（RatePlan クラス）の getPrice メソッドで電気料金を求める
                int price = super.getPrice(amount);
                //上記で求めた金額に割引を適用した金額を返す
                return (int)(price * (1.0- discountTiers.map(price)));
        }
}
```

e

ヒント：設問 2 プログラム 6 の説明「DiscountPlan は、クラス RatePlan を拡張し」

RatePlan を拡張＝RatePlan を継承
継承の場合は、
class サブクラス名 extends スーパークラス名

f

ヒント：設問 2 プログラム 6 の説明
「上位クラスである RatePlan のメソッド getPrice で求めた電気料金」

サブクラスのメソッドから上位クラス（スーパークラス）の同一名のメソッドを呼び出す場合、「super . メソッド名( )」と記述する。

g

ヒント：設問 2 プログラム 5 の説明
「メソッド map は、引数で与えられた電気料金から割引率を求め、その値を返す。ここで、割引率は小数で与えるものとする。例えば、1%は 0.01 である。」

DiscountTable クラスの map メソッドは、電気料金から割引率を求めるため、引数には直前で求めた price を渡す。map メソッドの戻り値（割引率）は double になるため、1.0 からマイナスしたものを price にかけて割引後の料金を求める。