第 49 回検定（平成 25 年 9 月 22 日）全商情処プロ部門 Java 解説

２級【７】

解答　（１）nen＜　　（２）saiko　　（３）gokei / ninzu
　　（４）i = hajime　　（５）kyoriKei

問１は、単なる if～else if の問題である。基本的な文法が分かっていれば容易に解ける。 問２は、最大値を求めるアルゴリズムである。この問題も、基本的なアルゴリズムを理解していれば比較的簡単に解ける。 問３は、一次元配列を使用した問題である。変数が多く悩むかもしれないが、実際のデータを当てはめて考えると解き易い。特別難しい問題でもない。 問１～問３全て基本的な問題であり、基本的な文法を理解していれば十分解けるであろう。

（１）は、処理条件の「7 歳以上 12 歳未満」を表せばよい。処理が（１）の else if まで降りてきている段階で、すでに 7 歳以上という条件を満たすことになる（厳密には nen <= 6 の逆となるので、7 歳以上ではなく 6 歳より上)。よって、7 歳以上という条件は入れる必要がなく、12 歳未満という条件を答えればよい。「未満」なのでイコールは付かないことに注意が必要である。

（２）は、最大値を更新する処理を記入する。データを読み込む都度、読み込んだ onryo が今までの最大値の saiko と同じかどうかを比較する。そして onryo は saiko を超えていれば、（２）で最大値を更新する、というものである。

（３）は、変数名が表す通り平均値を計算する。平均の計算は合計÷件数である。合計を表す変数は gokei、件数を表す変数は ninzu であり、これらも変数名から容易に役割を想像できる。なおこの計算は、heikin、gokei、ninzu 共に int 型変数なので、計算結果は端数を切り捨てらた int 型となる。

（４）では、for で使用する変数 i の初期値を設定する必要がある。なお、変数 i はプログラムの前半でローカル変数としてすでに定義されている。よって（４）では、定義する必要はなく初期値のみ代入すればよい。なお（４）で「int i = hajime」と答えると、ローカル変数が重複するのでコンパイルエラーとなる。変数 hajime と owari は、乗車駅番号と降車駅番号の 2 つのうち数値の小さい方を hajime とし、数値の大きい方を owari とするためのものである。これらを設けて hajime は owari より小さいという風にしないと、for の中で「hajime から owari まで表示する」という命令ができない。

（５）は駅の間の距離をプラスしていく処理である。右辺にも kyoriKei とあるので、まさか左辺の問題個所の（５）に同じ変数がくるかと驚くかもしれないが、正解は kyoriKei である。大文字小文字に中が必要である。

１級【７】

解答　（１）sortMonth[i] = i　　（２）sellTotal[month]
（３）max = idxTotal　　（４）serviceCode / 10
（５）goods[code].printSellTotal()

オブジェクト指向を取り入れたオーソドックスな問題である。しかし、Service クラスのインスタンスを参照型の配列 goods で管理していることを理解していなければ、全体として何をしているのかつかめないだろう。インスタンス変数やコンストラクタ、インスタンスメソッド等が網羅されており、模範的な良問と言えるであろう。 なお、基本的なことであるが Java は大文字小文字の区別を行う言語である。例えば、（１）の解答の「sortMonth[i] = i」は、「sortmonth[i] = i」では誤答となるので注意すること。

（１）が含まれている public Service(…)はコンストラクタである。コンストラクタで初期値を設定しているのが（１）であり、ここでは配列 sortMonth に 1～12 の数値を代入する必要がある。なお、サービス商品コードは 1～20 の 20 種類あり、その 20 種類の 1 つ 1 つがそれぞれ配列 sortMonth や配列 sellTotal を保持することとなる。これらインスタンスが保持するデータは class Service のすぐ下を見ると理解できる。private String name;から下 4 行がインスタンス変数であり、インスタンス毎に持つデータである。

（２）はインスタンスメソッドである。また、戻り値の型が何もないという void となっているので単純に○○.total()という風に呼び出せる。このメソッドで計算した money は sellTotal の 13 番目だけでなく、各月にもプラスする必要がある。添字として引数で受け取った month を使用すればよい。なお、呼び出し元の goods[idx].total(month,…)と、このメソッドの public void total(int month,…)が、同じ month という変数名になっているが、メソッド側は別に int month でなくても良い。変数名が同じ方がメソッドで行っていることが理解し易いのでこのような変数名のつけ方をしている。慣れるまではどの変数かつかむのが難しいが、メソッドの引数として書かれている int month 等は、単なる引数でありそのメソッドでしか使用されないものだ、と理解しておくとよい。同様の理由で、total(…)メソッド内で使われている optionUnitPrice と Cleaning クラスにある static なクラス変数の optionUnitPrice は別物である。

（３）は古典的な選択法による並べ替えの問題である。メソッド名が sortTotal()となっているので並べ替えをしているメソッドだということが容易に想像し易い。

（４）で計算している idx はその 2 行下で参照型配列 goods の添字として使用されている。なお、（４）が含まれている while 文は「Selling.csv を読み込み金額を計算する」という役割である。その上の while 文は「goods.csv を読み込みインスタンスを生成する」という役割である。インスタンス生成個所は、goods[code] = new Service(name, unitPrice);であり、goods の添字としてサービス商品コードが使用されていることが分かる。よって（４）でもサービス商品コードを使用する必要がある。

しかしファイル Selling.csv にはサービスコードしかないので、サービスコードからサービス商品コードを割り出す必要がある。よって解答は上記のようになるが、この問題は Java とは直接関係のない問題であり、普段から接している問題量や実際にプログラミングの場面での応用力等が試される。また、（４）の下の行の「%」は剰余を表している。

（５）は出力である。出力は配列 goods から参照される 20 個の Service インスタンスが 1 つずつメソッドとして保持している。メソッドは printSellTotal()である。よって、配列 goods から 1 つ 1 つ呼び出してあげればよい。問題としては簡単である。しかし、クラスやインスタンスの概念、また goods には参照が記憶されておりそれらがインスタンスを指し示している、そしてそのインスタンスは生成された時点でインスタンスメソッドを内包する、という様な点を理解していなければ解けないだろう。