# 基本情報技術者　Java 言語対策

## 補足資料

株式会社アイテック　IT 人材教育研究部

# 午後の言語問題の解法

本書を通じて，基本情報技術者試験に出題される Java の問題を解くのに十分な知識が備わったはずです。しかし，実際の試験で出題される午後問題を解くには，どうしてもその分量からくるプレッシャーがつきものです。 こうしたプレッシャーを跳ね除けるためには，問題を解くためのコツを知っていることが有効になります。

ここでは，こうしたコツを紹介します。

### （1）クラスやインタフェースを抽出する

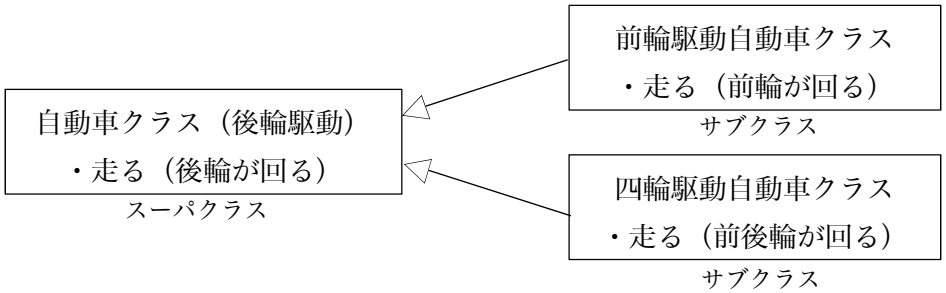
Java のプログラムコードは，これまで学習してきたように，クラスやインタフェースなど，オブジェクト単位で区分けされます。これが実務としてはファイルが分割されることになるのですが，**試験問題の中では〔プログラム 1〕，〔プログラム 2〕… といったように問題文の中のセクションが分けられますので，これを目印にして，登場するクラスやインスタンスの名前を列挙しましょう。**これはメモでも良いですし，頭の中で行っても構いません。

### （2）クラスやインタフェースの継承や実装関係を整理する

その次に，各クラス，インタフェース間の継承や実装関係性を整理しましょう。多くの場合，次の種類に分類できます。

① 他のクラスの継承元，つまりスーパクラスになっているクラス
② 他のクラスの継承先，つまりサブクラスになっているクラス
③ インタフェース
④ インタフェースを実装するクラス
⑤ 継承や実装の関係をもたないクラス

このうち，スーパクラスとサブクラスの間では，同じ型，同じ名前のメソッドが定義されていないか確認する必要があります。 もし，サブクラスがスーパクラスと同じ型，同じ名前のメソッドをもっている場合，サブクラスのもつメソッドはスーパクラスのメソッドをオーバーライドしたメソッドになります。**メソッドをオーバーライドしている場合は，まさにそこがそのサブクラスの特徴**ともいえる部分になります。



### （3）クラスの包含関係を整理する

問題に出てくるクラスやインスタンスの間には，継承関係の他，包含の関係も存在します。一番良い例は**プログラムのエントリポイントである `static void main` メソッド**をもつ最上位のクラスです。 `static void main` メソッドをもつクラスは，テスト用の上位モジュール，つまり“ドライバ”に該当します。

このドライバは，あくまでもテスト用なので簡単なプログラムである一方，モジュールの呼出し関係上は最上位に位置するため，このクラスのメンバや，`static void main` メソッドの内容はテスト用プログラムの処理概要を具体的に表しています。その他のクラスの間にも包含の関係が多くみられるでしょう。こうした点を整理しておくことが重要です。

### （4）ジェネリクス利用の有無を確認する

最近の Java プログラミングではジェネリクス（特に `List`， `Map`， `Set`）は必須と言っても過言ではないほど利用されています。試験問題にも頻繁に登場しますので，ジェネリクスが使われている部分を抽出して，整理しておきましょう。ジェネリクスが使われている場合は，`List`，`Map`，`Set` といったインスタンスの**生成，初期化，追加，削除**のタイミングなどを気にしながらプログラムコードを追うことで，プログラムの流れをより簡単に把握できるはずです。

また，インタフェースである `List`，`Map`，`Set` 型の変数に対して，実際のクラスである `ArrayList`， `HashMap`， `HashSet` のインスタンスを生成して代入する箇所は次のような書式になりますが，このうちのいずれかの部分が空欄になっている様な問題も出題されます。

```
List<Integer> data = new ArrayList<Integer>();
```

### （5）例外定義の有無を確認する

Java において“例外”とは，結局のところ例外処理（エラー処理）のために用意されたオブジェクトです。問題によって，この例外処理を記述している場合と，していない場合があります。**例外処理があるプログラムコードの方が，より実践に近い**ものともいえます。この例外処理がある場合，次の点を整理すると良いでしょう。

① 独自例外クラスを定義しているか
② `throw` による例外発生箇所があるか
③ `throws` によって，呼出し元に例外を上げるメソッド定義がないか
④ `catch` によって，例外をキャッチしているエラー処理ロジックがないか

こうした点を整理することで，エラーが発生した際の処理内容が把握できます。逆にこの例外が登場する問題の場合，**”`throw new ○○Exception()`” の様に，例外を発生させる**

**部分**の穴埋め問題になっていたり，**”catch（〇〇Exception ex）” の様に，発生した例外をキャッチする部分**，そして**”void 〇〇function() throws 〇〇Exception” の様に例外の発生可能性について宣言したりする部分**で，文法的な空欄穴埋めを行う問題も多く出題されます。

### （6）スレッドや排他制御の有無を確認する

プログラム内に，**“Thread”**や**“Runnable”**が登場しないか，確認しましょう。これらが登場する場合，スレッドによる並行処理がテーマになっているため，複数のスレッド間で行われる排他制御も話題になることが想定されます。具体的には **synchronized** キーワードが登場すると思って間違いないでしょう。

以　上