徹底攻略

試験番号 1Z0-814

Java SE 7/8 Bronze

[1Z0-814] 対応

問題集

志賀澄人／山岡敏夫［著］
株式会社ソキウス・ジャパン［編］

実績と信頼の黒本シリーズ

迷わずにこの一冊。

Java SE 7/8 対応

模擬問題2回分付き！ 紙面 1回分＋Web 1回分

「最重要ポイント」がひと目でわかる。 試験対策

シリーズ100万部突破!!

インプレス

本書は、Java SE 7/8 Bronzeの受験対策用の教材です。著者、株式会社インプレスは、本書の使用による同試験への合格を一切保証しません。

本書の内容については正確な記述につとめましたが、著者、株式会社インプレスは本書の内容に基づく試験の結果にも一切責任を負いません。

インプレスの書籍ホームページ

書籍の新刊や正誤表など最新情報を随時更新しております。

http://book.impress.co.jp/

はじめに

1996年に登場したJavaは、バージョンが上がるたびに改良が加わり、大きく成長してきました。Java SE 8では、さらなる言語仕様の拡張が行われ、より高い生産性を実現できる言語に進化しています。

本書は、Javaの認定資格のうち、Oracle Certified Java Programmer, Bronze SE 7/8認定資格を取得するための試験（1Z0-814）を受験される方を対象としています。

この試験は、これからプログラミングを始める初心者を対象に、Java言語の基礎的な知識を問うものです。Javaの特徴から基本的なプログラミングの制御、オブジェクト指向の基礎概念はもちろんのこと、Java SE 7で導入された新しい文法についても一部出題されます。

私と共著者の山岡氏は、長年の間、Javaをはじめとしたさまざまな技術を教える仕事をしています。これまでの技術者育成の経験をいかし、「理解するための問題と解説」となるよう心がけました。出題は、ポイントごとに分かれているため、何を問うているのかを予想してから解くことをお勧めします。また、問題ごとにバラバラの解説をするのではなく、章全体の解説を通して理解が深まるようにしています。ぜひ、分からない問題の解説だけでなく、その前後の解説も併せて読んでください。

資格は取得することも重要ですが、その過程はもっと重要です。有意義な試験対策の時間を過ごしていただきたいという思いから、初心者だからと情報を省略することはしていません。正解を探す「宝探し」のために問題を解いて終わりにするのではなく、理解するために本書を使っていただければ幸いです。本書が、読者の皆さんのお役に立つことを心から願っております。

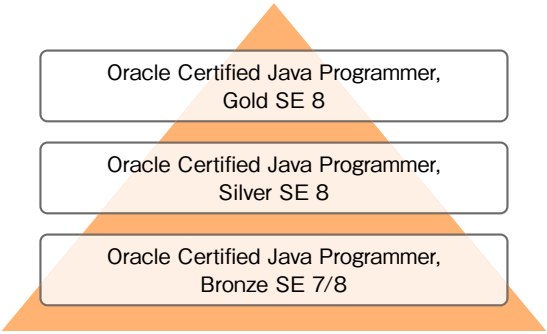
最後に、共著者である山岡氏と手厚いサポートをくださったソキウス・ジャパンの皆さんに、この場をお借りしてお礼申し上げます。

志賀 澄人

Java SE 8 認定資格について

Java SE 8認定資格は、米オラクル社がワールドワイドで提供している認定資格で、「Java Platform, Standard Edition 8」に対応しています。Java SE 8は、冗長的なコードの削減、コレクションやアノテーションの改善、並列処理プログラミング・モデルの簡素化、最新のマルチコア・プロセッサの効率的な活用により、企業システムやクラウド・サービス、スマート・デバイスなどで活用されるアプリケーション開発を加速させます。この資格を取得することで、業界標準に準拠した高度なスキルを証明します。

資格は、入門レベルから高度なスキルを証明できるプロフェッショナルレベルまで3段階に分かれています（以下の図を参照）。また、Java SE 6以前のバージョンに対応するJavaプログラマ認定資格の取得者を対象に、Oracle Certified Java Programmer, Gold SE 8（OCJ-P Gold SE 8）へのアップグレードパスが用意されています。



【Java SE 8 認定資格の種類】

認定資格名および対象	試験番号および試験名	受験前提条件
OCJ-P Bronze SE 7/8 言語未経験	1Z0-814 Java SE 7/8 Bronze	なし
OCJ-P Silver SE 8 開発初心者	1Z0-808 Java SE 8 Programmer I	なし
OCJ-P Gold SE 8 中上級者	1Z0-809 Java SE 8 Programmer II	OCJ-P Silver SE 8
	1Z0-810 Upgrade to Java SE 8 Programmer	OCJ-P Gold SE 7
	1Z0-813 Upgrade to Java SE 5 and 6 to Java SE 8 Programmer	Java SE 7よりも前のバージョンのSun認定Javaプログラマ

OCJ-P Bronze SE 7/8 について

OCJ-P Bronze SE 7/8は、プログラミング経験1年未満の初心者向けの資格です。この資格は、日本でのみ認定されるものです。

OCJ-P Bronze SE 7/8の試験科目「Java SE 7/8 Bronze」では、プログラミング言語を学習し始めて間もない人が理解すべき基礎的な知識が身についているかが問われます。出題されるトピックは次のようなもので、Java言語とオブジェクト指向の基礎に対する理解度が問われます。より詳細な内容は、日本オラクルのWebサイトで確認することができます。

・Java言語のプログラムの流れ
・データの宣言と使用
・演算子と分岐文
・ループ文
・オブジェクト指向コンセプト
・クラスの定義とオブジェクトの生成、使用
・継承とポリモフィズム

Java SE 7/8 Bronze 試験について

本書では、OCJ-P Bronze SE 7/8の試験科目「Java SE 7/8 Bronze」を扱います。以下の試験概要は、2015年5月現在の内容です。また、合格ラインや試験料は今後変更される可能性があります。最新の試験情報は、必ず日本オラクルのWebサイトなどで確認してください。

●試験概要（2015 年 5 月現在）

・試験名　　:Java SE 7/8 Bronze
・試験番号　:1Z0-814
・試験時間　:65分
・問題数　　:60問
・合格ライン:60%
・試験方法　:CBTによる選択式
・試験料　　:13,600円（税抜き）

受験申し込み方法

「Java SE 7/8 Bronze」試験は、ピアソンVUE社の公認テストセンターまたはオンライン試験で受験可能です。

●公認テストセンターでの受験

申し込みは、ピアソンVUE社のコールセンターまたはWebサイトを利用して行ってください。いずれの場合も希望するテストセンター、日時を選択できます。予約状況によっては選択できない場合もありますので、必ず申し込み時に確認してください。なお、初めてピアソンVUE社に申し込む場合は、同社のWebサイトでアカウント情報を登録する必要があります。

●オンライン試験での受験

オンライン試験とは、インターネット経由で受験する試験監督不在の試験です。都合のよい時間にどこからでも受験できます。申し込みは、ピアソンVUE社のWebサイトから可能です。

ピアソン VUE 社

- ・URL　http://www.pearsonvue.com/japan/IT/oracle_index.html
- ・TEL　0120-355-583 または 0120-355-173
- ・FAX　0120-355-163
- ・Eメール　pvjpreg@pearson.com

●オラクル認定資格に関する問い合わせ先

オラクル認定資格に不明な点がある場合は、オラクル認定資格事務局のメールアドレスに問い合わせることができます。

- ・Eメール　oraclecert_jp@oracle.com

本書の活用方法

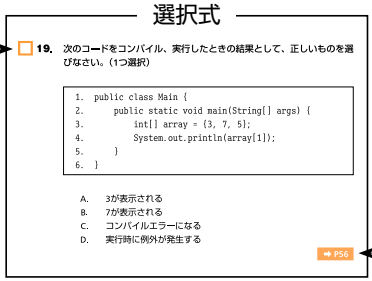
本書は、カテゴリ別に分類された、問題と解答で構成されています。試験の出題範囲に沿った問題に解答したのち、解説を読んで学習すると、合格レベルの実力が身に付きます。また、実際の試験に近い形式になっていますので、より実戦的に学習できます。

問題

試験の出題形式は選択式です。正答を1つだけ選ぶものと複数選ぶものがあります。

問題や解説のソースコードには全文を掲載したもの（行番号が1行目からのもの）と抜粋のみ（行番号が11行目からのもの）があります。ただし、適切なソースコードを選択する問題の一部は、選択肢のコードが複数行になるものがあるため、必ずしも行番号を付与していない場合もあります。

また、抜粋のソースコードでは、mainメソッド、例外処理、import文などを省略し、プログラムを動作させるのに不完全な形で記述されているものもあります。このようなソースコードは、実際の試験でも同様に出題されることが考えられます。本書の問題に解答する際には、省略されている記述にかかわらず、問題の趣旨を的確にとらえるようにしてください。



チェックボックス

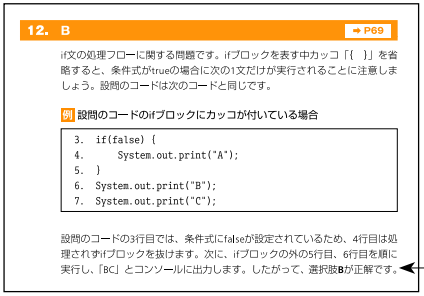
確実に理解している問題のチェックボックスを塗りつぶしながら問題を解き進めると、2回目からは不確かな問題だけを効率的に解くことができます。すべてのチェックボックスが塗りつぶされれば合格は目前です。

解答ページ

問題の右下に、解答ページが表示されています。ランダムに問題を解くときも、解答ページを探すのに手間取ることがありません。

解答

解答には、問題の正解および不正解の理由だけでなく、用語や重要事項などが詳しく解説されています。また、解説のコード例を試して学習することでさらに理解が深まります。



解説（用語）

重要な用語やキーワードと、正解の選択肢は「選択肢**B**」のように太字で示しています。

目次

第1章

Java言語のプログラムの流れ

- Javaの特徴
- Javaプログラムの作成と実行
- 各種エディションの特徴

1. Javaに関する説明として、正しいものを選びなさい。(2つ選択)

A. すべての変数は、型の宣言をしなければならない
B. すべての式は、型を持たない
C. 式の型はコンパイル時に解釈される
D. 実行時にエラーを発生させることで、型の整合性を確認する

➡ P15

2. Javaに関する説明として、正しいものを選びなさい。(2つ選択)

A. マルチスレッドによる並行処理をサポートする
B. マルチプロセスによる並行処理をサポートする
C. シングルスレッドアプリケーションのみをサポートする
D. 並行処理を完全に制御できる
E. 並行処理を部分的に制御できる

➡ P16

3. Javaに関する説明として、正しいものを選びなさい。(2つ選択)

A. 特定のOSに特化したプログラミング言語である
B. ほかの言語に比べて高速に実行できる
C. あらかじめ機械語にコンパイルされる
D. メモリ管理が自動化される
E. セキュリティが向上する実行方式を取り入れている

➡ P17

4. Javaプログラムの作成から実行までの流れに関する説明として、正しいものを選びなさい。(2つ選択)

A. コンパイラによって機械語にコンパイルされる
B. コンパイラによって中間コードにコンパイルされる
C. 実行可能ファイルを作成する
D. JVMにクラスファイルを読み込ませる

➡ P19

5. Javaのクラスファイルに関する説明として、正しいものを選びなさい。（1つ選択）

A. プラットフォームに特化したネイティブコードが記述されている
B. プラットフォームに依存しないネイティブコードが記述されている
C. JVMだけが理解できるコードが記述されている
D. 人間が理解できるコードが記述されている

➡ P21

6. Javaに関する説明として、正しいものを選びなさい。（2つ選択）

A. 自動的にメモリを解放する
B. メモリを任意のタイミングで解放できる
C. ポインタを使ってメモリを自由に操作できる
D. メモリの効率的な利用を自動化する

➡ P21

7. Javaのソースファイルに関する説明として、正しいものを選びなさい。（2つ選択）

A. 1つのソースファイルにpublicなインタフェースを複数記述できる
B. ソースファイルの名前はpublicなクラス名と一致させなくてはいけない
C. 1つのソースファイルには、1つのクラスだけを記述できる
D. 1つのソースファイル内に、デフォルトのアクセス修飾子で修飾したインタフェースとpublicなクラスの両方を記述できる
E. 1つのソースファイルにpublicなクラスを複数記述できる

➡ P22

8. JavaのエディションのうちJava SEに関する説明として、正しいものを選びなさい。（2つ選択）

A. JVMが含まれる
B. GUIアプリケーション開発に向いている
C. 大規模システム向けの機能をセットにして提供している
D. 仕様のみを提供している

➡ P23

9. Javaのエディションのうち**Java EE**に関する説明として、正しいものを選びなさい。（2つ選択）

A. 各社がオラクル社の認定を受け、実装を提供している
B. Java SEの範囲は含まない
C. エンタープライズ用途向けの多くの機能をセットにしたものである
D. 携帯電話のような、リソースが制限されたデバイス向けの機能を提供している

➡ P25

10. Javaのエディションのうち**Java ME**に関する説明として、正しいものを選びなさい。（2つ選択）

A. Java SEの機能の一部を抜き出して定義したエディションである
B. 基本的なライブラリとJVMを組み合わせたものを「プロファイル」と呼ぶ
C. 特定のデバイスに対応するAPIだけを抽出したものを「コンフィギュレーション」と呼ぶ
D. Java MEでは、JVMではなく「KVM」と呼ばれる仮想マシンを使うことがある

➡ P26

第 1 章　Java言語のプログラムの流れ
解　答

1. A、C

➡ P12

変数や式の型の扱い方についての問題です。
プログラミング言語は、**静的言語**と**動的言語**の2つに分かれます。これらの違いは、変数や式の**型情報の扱い方**にあります。**型**とは、データの種類を表すための情報です。変数や式の結果作られるデータの種類を、コンパイル時に決定するのが「静的言語」、実行時に決定するのが「動的言語」です。型を決めるタイミングがプログラムの実行前か実行時かによって静的か動的かに分かれます。

静的言語の場合は、**変数**や**式**が型情報を持つのが特徴です。式の型は、**コンパイル時**に決定されます。期待されている型と式の結果の型が合わない場合には、コンパイルエラーが発生します。
一方の動的言語は、変数や式が型情報を持たず、変数宣言時にもデータ型を指定しません。言語によっては、変数宣言そのものがないものもあります。動的言語では、変数や式が型情報を持たない代わりに**データ自身**が型情報を持つため、**実行時**に型がチェックされます。そのため、型の不整合があった場合は、実行時にエラーが発生します。

さらに型の整合性をチェックする仕組みには、静的であるか動的であるかのほかに「**強さ**」という尺度があります。この強さとは、どの程度まで厳密に型の整合性をチェックするかという厳密性を表します。
たとえばJavaの場合は変数や式が型情報を持ち、コンパイラによって厳密に不整合のチェックが行われるため、「**強い静的型付け言語**」と呼ばれます。一方、C++は、柔軟な構文を採用しているために、厳密なチェックができないことから「弱い静的型付け言語」と呼ばれます。

動的言語にも「強さ」はあります。たとえば、「強い動的型付け言語」の代表例はLispで、「弱い動的型付け言語」の代表例はアセンブラです。

前述のとおり、Javaは強い静的型付け言語です。そのため、Javaは変数や式が型を持ち、コンパイル時に厳密な型チェックが行われます。したがって、選択肢**A**と**C**が正解です。

2. A、E

➡ P12

Javaの特徴に関する問題です。
処理を複数同時に実行することで、ソフトウェア全体の処理性能を向上させる技術のことを「**並行処理**」と呼びます。Javaは、この並行処理を容易に実現できるよう専用の構文や標準クラスライブラリを提供しており、ほかのプログラミング言語では実装が困難だった並行処理が比較的容易に実現できるようになっています。

Javaに限らず、一般的な並行処理の実現方法には次の2つがあります。

・同じアプリケーションを複数実行する
・1つのアプリケーションで複数の処理を交互に切り替えて実行する

これらの実現方法のうち、前者を「**マルチプロセス**」、後者を「**マルチスレッド**」と呼びます。

プロセスとは、アプリケーションが起動するときに、OSから割り当てられたメモリ空間のことを指します。このメモリ空間には、プログラムを実行するのに必要なコードや変数、その他すべてが展開されていることから、プロセスは「アプリケーションそのもの」と呼ぶこともできます。マルチプロセスは、アプリケーション（プロセス）を複数同時に起動し、OSが交互にアプリケーションを切り替えながら実行することで並行処理を実現します。

もう一方のマルチスレッドは、プロセス内を複数に分割することで並行処理を実現します。**スレッド**とはプロセス内で実行される一連の処理の流れのことで、マルチスレッドとは、1つのプロセス内で複数の処理の流れ（スレッド）が並行に実行される並行処理の形態です。

マルチプロセスの問題点は、マルチスレッドに比べて起動に時間がかかることです。これは、プロセス起動時にOSからメモリ空間を割り当ててもらう必要があるためです。それに対してマルチスレッドは、割り当て済みのメモリ空間を分割して使うため、マルチプロセスに比べてパフォーマンスに優れるというメリットがあります。その他の違いは次の表のとおりです。

【マルチプロセスとマルチスレッドの違い】

	マルチプロセス	マルチスレッド
メリット	・プログラム異常時に、ほかの処理に影響が出にくい ・プログラムがシンプルでわかりやすい ・各処理の終了時にリソースが確実に解放される ・利用可能なメモリ量やCPU時間などのリソース制限を受けない	・リソースを節約できる ・並行する処理数をすばやく増やせる ・データ連携が簡単である ・さまざまなプラットフォームで実現しやすい
デメリット	・リソースを多く必要とする ・プロセス生成時の処理に時間がかかる ・プロセス間のデータ連携が面倒 ・UNIX系OS以外では利用が難しい	・プログラムが複雑化しやすい ・プログラム異常時に、ほかの処理に影響が出る ・利用可能なメモリ量やCPU時間などのリソースの制限を受ける

Javaはシングルプロセス、マルチスレッドで並行処理を実現しますが、どのスレッドを実行するかはJVMが判断するため、プログラムから並行処理を完全に制御できるわけではありません。
Javaでは、並行して「やりたいこと」をプログラミングできますが、その処理順を自由に制御できるわけではないことに注意しましょう。

以上のことから、選択肢**A**と**E**が正解です。

3. D、E

➡ P12

Javaの特徴に関する問題です。
Javaの特徴を表す言葉に「Write Once, Run Anywhere（一度書いたら、どこでも動く）」があります。この特徴を実現しているのが**JVM**（Java Virtual Machine）という仮想的なコンピュータです。

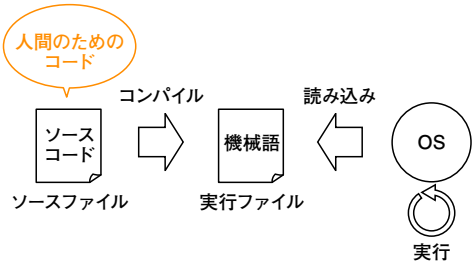
私たちが普段会話に使う言語のことを「自然言語」と呼びます。自然言語に近い文法やボキャブラリを使って記述する**ソースコード**は、人間が読み書きするためのものです。一方、機械語しか理解できないコンピュータはソースコードを理解できません。そのためソースコードは、コンピュータが理解できる「機械語」にプログラムの実行前にいったん変換しておく必要があります。このソースコードから機械語への変換作業のことを「**コンパイル**」、この作業をする変換ソフトウェアのことを「**コンパイラ**」と呼びます。

※次ページに続く

コンパイラは、プログラムを実行するコンピュータのOSが理解できるようにするために、ソースコードをコンパイル（変換）します。そのため、プログラムを実行したいOSが異なれば、コンパイル結果も異なります。たとえば、Windows用にコンパイルしたプログラムはWindows専用、Mac OS用にコンパイルしたプログラムはMac OS専用となり、これらの間に互換性はありません。

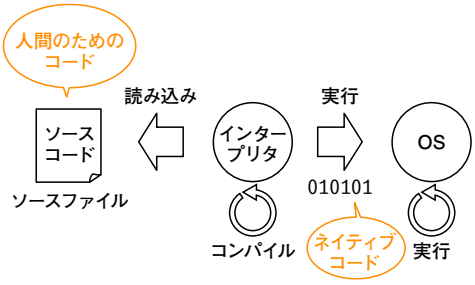
このように、ソースコードをあらかじめコンパイルしてから実行する方式のことを「**事前コンパイル方式**」と呼びます。この方式のメリットは、対象OS専用のコードに変換されているため高速に実行できることです。その反面、専用コードであるために対象OS以外では実行できないことがデメリットです。事前コンパイル方式を採用しているプログラミング言語には、CやC++などがあります。

【事前コンパイル方式】



プログラムの実行方式には、この事前コンパイル方式以外に「**インタープリタ方式**」と呼ばれるものもあります。この方式は、「**インタープリタ**」と呼ばれる仲介アプリケーションを使って、ソースコードを**実行時にコンパイル**することが特徴です。インタープリタ方式では、事前にコンパイルし、専用コードに変換しておく必要がないため、理論上は対象OSごとのインタープリンタを用意すれば、プログラムをどのようなOSでも実行できます。このインタープリタ方式を採用しているプログラミング言語には、PHPやJavaScriptなどがあります。

【インタープリタ方式】



インタープリタ方式はソースコードを1行ずつコンパイルしながら実行するため効率が悪く、事前コンパイル方式に比べて実行速度が落ちてしまうというデメリットがあります（選択肢B）。

Javaは「Write Once, Run Anywhere」を実現するために、このインタープリタ方式を採用しており、JVMがプログラムの実行時にOS専用のコードにコンパイルしながらプログラムを実行します（選択肢C）。そのため、OSごとにJVMを用意すれば、Javaのプログラムはどこでも実行できるのです。なお、Javaも事前にコンパイルをしますが、この作業の必要性については解答4で解説します。

Javaがインタープリタ方式を採用したのは、以下のような理由からです。

1. 特定のプラットフォームやOSに依存しない
2. ガベージコレクションによってメモリ管理が自動化できる（選択肢**D**）
3. セキュリティが向上する（選択肢**E**）

理由1は、インタープリタ方式の特徴そのものです。この方式であれば実行時にコンパイルするため、実行環境に合わせたJVMを用意すれば、どのようなコンピュータやOSであっても実行可能です（選択肢A）。

インタープリタ方式の場合、OSが直接プログラムを実行するのではなく、JavaであればJVM、JavaScriptであればブラウザという具合に、コンパイルしながら実行する仲介アプリケーションが存在します。この仲介アプリケーションが、実行時にどのようにメモリを使用するかを決めたり、不要になったメモリはないかを確認したり、問題のあるコードはないかをチェックしたりしながらプログラムを実行するため、上記の理由2と3のようなメリットが生まれるのです。ガベージコレクションについては、解答6を参照してください。
以上のことから、選択肢**D**と**E**が正解です。

4. B、D

➡ P12

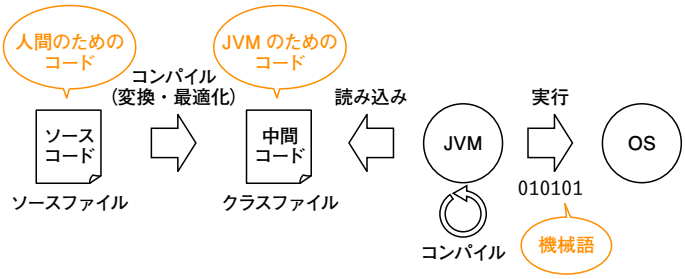
Javaプログラムの作成から実行までの流れに関する問題です。
「Write Once, Run Anywhere」を実現するために、JavaはJVMによるインタープリタ方式の実行形態を採用しています。これによりさまざまなメリットが得られることは解答3で解説したとおりですが、いくつか問題点もあります。もっとも顕著な問題が、パフォーマンスです。

ソースコードは人間が理解しやすいように記述したものであって、プログラムが実行しやすいかどうか、効率よく実行できるかどうかという観点で記述

したものではありません。そこで、ソースコードから不要なコードを排除し、パフォーマンスが向上するようにコードを変換しておく方法が「**実行時コンパイル方式**」です。

実行時コンパイル方式では、コンパイラによって実行に最適化されたコードに変換されているため、インタープリタでもパフォーマンスが低下しにくいという特徴を持ちます。Javaは、この実行時コンパイル方式を採用した「**Hotspot VM**」という技術をJVMに導入しています。その結果、Javaは事前にコンパイルしておく言語と比べても遜色ないほどのパフォーマンスを持つことに成功したのです。

【実行時コンパイル方式】



なお、Javaの実行時コンパイル方式では、コンパイラによって変換されたコードを「**中間コード**」と呼びます。中間コードは、より効率がよいコードへの最適化だけでなく、変換効率を上げるために2進数で表現されるバイトコードで記述されています。**クラスファイル**は、この中間コードが記述されたファイルのことを指します。以上のことから、選択肢Aは誤りで、選択肢**B**と選択肢**D**が正解です。なお、実行時コンパイル方式はインタープリタ方式の一種といえますが、高速化のために事前にバイトコードに変換されている点が異なります。

Javaは、実行時にネイティブコードにコンパイルしてもファイルには書き出さず、CやC++のように実行可能ファイルを作ることはありません。JVMが実行時にコンパイルしたネイティブコードは、そのまま実行されるか、頻繁に実行するコードであればメモリ上にキャッシュされます。よって、選択肢Cも誤りです。

5. C ➡ P13

クラスファイルに関する問題です。
解答4で解説したとおり、Javaは実行時コンパイル方式を採用しています。コンパイル後に生成されるクラスファイルには、「中間コード」と呼ばれる実行に最適化されたコードが記述されています。JVMはこの中間コードを読み込み、機械語にコンパイルして実行します（選択肢A、**C**、D）。
なお、機械語が特定のプロセッサ群の「固有語」であることから、「ネイティブコード」と呼ぶこともあります。そのため、選択肢Bのようにプラットフォームに依存しないネイティブコードというものは存在しません。
以上のことから、選択肢**C**が正解です。

6. A、D ➡ P13

Javaのメモリ管理に関する問題です。
メモリは有限なリソースであり、無尽蔵に使えるわけではありません。次々とメモリを使用し続けた結果、もしメモリが不足してしまえば、メモリリークが発生し、処理速度が極端に遅くなったり、エラーが発生したり、システムが突然終了したりする事態が発生します。最悪の場合、OSをも巻き込んだトラブルへと発展します。ソフトウェアを安定稼働させるためには、使わなくなったメモリ領域を解放し、プログラムの実行に必要な空きメモリを常に確保するメモリ管理が欠かせません。

Java以前の言語では、プログラマーがメモリを解放するコードを明示的に記述していました。当時は「今、メモリはどのような状況なのか？」「どのメモリをいつ解放すればよいか？」「このタイミングで解放しても大丈夫なのか？」という具合に、常にメモリの状況を考えながらプログラミングする必要がありました。このようなプログラミングは考慮すべきことが多いために生産性が落ちてしまい、開発コストを増加させる原因になっていたのです。

この問題を解決するために、Javaはガベージコレクションという自動メモリ管理機能を備えています。ガベージコレクションは、確保しておく必要がなくなったメモリ領域を自動的に検出し、解放する機能です（選択肢**A**）。この機能のおかげで、プログラマーはメモリ管理を行う必要がなくなり、より生産的な作業に時間を費やすことができるようになりました。

ガベージコレクションは、「ガベージコレクタ」と呼ばれるJVMの機能が実行します。ガベージコレクションは、ガベージコレクタのアルゴリズムに従って実行されるため、プログラマーがメモリ解放のタイミングを制御することはできません（選択肢B）。Systemクラスにはgcメソッドというガベージコレクションに関するメソッドがありますが、このメソッドはJVMにガベージコ

レクションの実行を促すだけであって、必ずしもガベージコレクションが発生するわけではありません。

また、ガベージコレクタには、ガベージコレクションによってメモリ領域に空き領域ができたとき、細切れになったメモリ領域を整理し、空き領域を確保する「コンパクション」と呼ばれる機能もあります。Javaは、この機能のおかげで効率的にメモリ領域を使うことができるのです（選択肢**D**）。

CやC++といった言語では、メモリアドレスを指し示す「ポインタ」を用いてメモリを自由に操作することができます。このポインタはプログラミングの自由度を上げる反面、解決が困難なバグの原因にもなっていました。そのため、Javaでは、プログラムから直接メモリ操作はできないようになっています（選択肢C）。

以上のことから、選択肢**A**と**D**が正解です。

7. B、D ➡ P13

ソースファイルに関する問題です。
ソースファイルには、publicで修飾されたクラスやインタフェース、列挙型は1つしか記述できません（選択肢A、E）。

ソースファイルの名称はpublicなクラスの名前、インタフェース名、列挙型名のいずれかと一致させなければいけません（選択肢**B**）。もし、名称が一致しなければコンパイルエラーが発生します。たとえば、次のSampleクラスを「sample.java（最初のSが小文字）」として保存し、コンパイルすると次のようなコンパイルエラーが発生します。

例 Sampleクラス

```
public class Sample {
    // any code
}
```

例 実行結果

```
> javac sample.java
sample.java:1: クラス Sample は public であり、ファイル Sample.java で宣言しなければなりません。
public class Sample {
       ^
エラー 1 個
```

publicなクラスやインタフェース、列挙型はソースファイルに1つしか記述できませんが、それ以外のアクセス修飾子を持つものであれば複数記述できます（選択肢C、**D**）。たとえば、次のコードのようにSampleクラスとTestクラスを1つのソースファイルに記述することが可能です。

例 publicなSampleクラス、およびTestクラスが記述されたソースファイル

```
public class Sample {
    // any code
}

class Test{
    // any code
}
```

ソースファイル内に複数のクラスを定義した場合でも、コンパイルするとクラスファイルはそれぞれクラスごとに出力されます。そのため、前述のソースファイル（Sample.java）をコンパイルすると、Sample.classとTest.classの2つのクラスファイルが出力されます。
ただし、このように1つのファイルに複数のクラスを記述することは推奨されません。ソフトウェアの規模が大きくなり、クラス数が増えてくると、どのファイルにどのクラスを定義したのかがわかりづらくなるためです。

8. A、B ➡ P13

Java SEの特徴に関する問題です。
Javaには用途に応じて、Java SE（Java Platform, Standard Edition）、Java EE（Java Platform, Enterprise Edition）、Java ME（Java Platform, Micro Edition）の3つのエディションが用意されています。設問は、この3つのエディションのうち**Java SE**の特徴について問うものです。

Java SEの主な特徴は次のとおりです。

- JVMの提供
- 標準クラスライブラリの提供
- 各種の開発ツールの提供

Java SEは、**JRE**と**JDK**という2つのパッケージで提供されています。JREはJava Runtime Environmentの略で、Javaプログラムの実行に必要なライブラリ、JVM、その他必要なコンポーネントをまとめて提供しています（選択肢**A**）。もう1つのJDKはJava Development Kitの略で、JREに加えて開発に必要なコン

パイラやデバッガ、各種開発ツールが含まれています。

標準クラスライブラリは、大きく分けて3つの機能を提供しています。1つ目が基本的な機能を提供する「基本ライブラリ」です。このライブラリには、java.langやjava.utilをはじめとする基本的なパッケージが含まれています。ほかにも、I/Oやシリアライズ、ネットワーク機能、セキュリティや国際化対応、JVMを監視するためのJMX（Java Management Extensions）、XMLを扱うためのJAXP（Java API for XML Processing）、Javaとネイティブアプリケーションの連携を実現するJNI（Java Native Interface）などもこのライブラリに含まれます。

2つ目は、応用的な機能を提供する「統合ライブラリ」です。これにはデータベース連携を実現するJDBC（Java Database Connectivity）や分散アプリケーションを開発するためのRMI（Remote Method Invocation）、CORBA（Common Object Request Broker Architecture）、RMI-IIOP（RMI over IIOP）、ディレクトリサービス連携を実現するJNDI（Java Naming and Directory Interface）といった機能が含まれます。

最後が「ユーザーインタフェースライブラリ」です。これには、GUIを実現するAWTやSwing、画像処理をするためのJava 2D、印刷サービスやテキスト変換などが含まれています（選択肢**B**）。

大規模システム開発では、大きなシステムが複数のサブシステムに分割され、それらが互いに連携し合いながら全体の処理を進めていきます。このような連携機能や付随する機能を提供するのはJava EEの役割です（選択肢C）。

また、Java SEがライブラリとして実装を提供するのに対し、Java EEは実装を提供しません。Java EEは仕様の集合体として提供され、その実装は各種ベンダーやオープンソースコミュニティが提供しています（選択肢D）。

以上のことから、選択肢**A**と**B**が正解です。

9. A、C

➡ P14

Java EEの特徴に関する問題です。
仕事の多くが自動化され、データ化された昨今、1つの企業で使われるソフトウェアの数は増えるばかりです。1,000を超えるソフトウェアを連携させている企業も珍しくありません。このようにたくさんのソフトウェアが連携し、企業の仕事全体を効率化するソフトウェア群のことを「エンタープライズシステム」と呼びます。

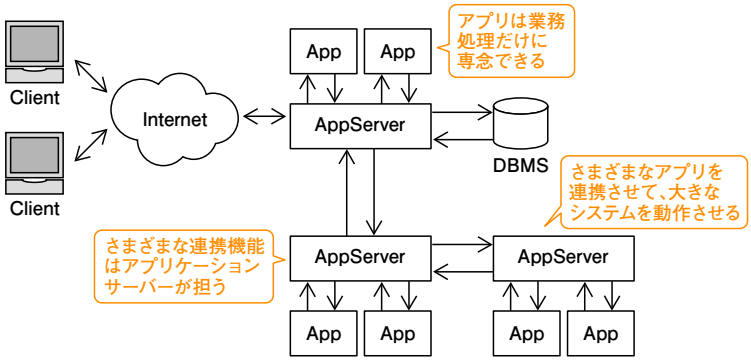
エンタープライズシステムの基盤となるのは、次のようなソフトウェア間の連携機能です。

- 物理的に離れた場所にあるソフトウェアの機能をほかのソフトウェアが利用する機能
- ネットワーク上にあるソフトウェア群から目的のものを探し出す機能
- 連携した情報を保存しておき、あとから再利用する機能
- 一連のデータ処理が確実に行えることを保証する機能
- 連携するソフトウェアが変更されても影響を受けないようにする機能

このような基盤機能は、エンタープライズシステムであればほとんどのソフトウェアに必要なものばかりです。そのような機能を、アプリケーションを開発するたびに作っていては非効率的であるため、これらの機能を実装したソフトウェアを利用します。なお、このような基盤機能を提供してくれるソフトウェアのことを「アプリケーションサーバー」と呼びます。

アプリケーションサーバーは、基盤機能を提供するだけであり、単体で動作することはありません。アプリケーションサーバーとさまざまな業務処理をするアプリケーションとを組み合わせることで、既存のエンタープライズシステムと連携して動作する「エンタープライズアプリケーション」となります。

【エンタープライズアプリケーション】



現在、利用可能なアプリケーションサーバーは、有償・無償合わせて数多く存在します。アプリケーションサーバーごとにAPIが異なると、業務処理をするアプリケーションは、利用しているアプリケーションサーバー以外と連携できなくなります。これではJavaのもっとも重要な特徴である「Write Once, Run Anywhere」が実現できません。そこで、これらのアプリケーションサーバーが実現しなければいけない機能の仕様やそのAPIを定め、これらに準拠しているかどうかのテストをクリアした製品が流通する仕組みが用意されました。この機能の仕様やAPIを定めたものが「**Java EE**」です。

Java EEは、各社が提供するソフトウェアが不足なく機能を提供し、かつ互換性があることを保証するための仕様を定めています。さまざまなベンダーが製品を提供していますが、Java EEが規定する仕様を満たす製品であれば、同じ機能が提供され、かつ互換性を保って使うことができるのです。なお、このJava EEの仕様を満たし、オラクル社から「Java EE Compatibility」認定を受けた製品のことを、「**Java EEアプリケーションサーバー**」と呼びます。以上のことから、選択肢**A**と**C**が正解です。その他の選択肢は、以下の理由により誤りです。

B. Java EEは、Java SEを拡張して作られています。そのため、Java EEアプリケーションサーバーの実行にはJava SEが必須です。
D. 携帯電話などのようなリソースが限られたデバイス向けの機能を提供しているのはJava MEです。

10. A、D ➡ P14

Java MEの特徴に関する問題です。
Java MEは、携帯電話やPDAなどの携帯端末、工業用ロボット、テレビのセットトップボックス、プリンタなど、多種多様なハードウェアを制御するソフトウェアを作るためのエディションです。このようなハードウェアは、限られたメモリサイズ、狭いディスプレイ、容量が少ないバッテリーなど多くの制約があります。コンピュータ向けの応用的な使い方やユーザーインタフェースに関するライブラリは、こうした制約のあるハードウェアを制御するためには必要ありません。このため、Java MEのライブラリはJava SEの標準ライブラリから最低限のAPIだけを抜き出して提供されています（選択肢**A**）。

Java MEでは、こうしたハードウェアなどの環境に柔軟に対応するために、次の3つの要素から構成されています。

・コンフィギュレーション……もっとも基本的なライブラリと仮想マシン（選択肢B）
・プロファイル……特定のデバイス向けのAPIセット（選択肢C）

・オプショナルパッケージ……特定の技術をサポートするためのAPIセット

本書では詳細を割愛しますが、これらの要素にはいくつもの種類があり、対象となるハードウェアに合わせてさまざまな組み合わせが存在します。さまざまな組み合わせによって、多種多様なハードウェアに対応しています。

前述のとおり、コンフィギュレーションには仮想マシンも含まれます。しかし、すべてのハードウェアがコンピュータ用のJVMを実行できるリソースを持つわけではありません。ハードウェア要件によっては十分なリソースを確保できない場合もあり得ます。そこでJava MEでは、JVMのほかに**KVM**という小型ハードウェア向けの仮想マシンも用意しています（選択肢**D**）。KVMのKは、数十キロバイト単位の小さなサイズを扱うということに由来しています。

以上のことから、選択肢**A**と**D**が正解です。

第2章

データ宣言と使用

- データ型
- プリミティブ型と参照型
- 変数の宣言
- スコープ
- 定数の宣言
- 配列（宣言、生成、初期化）
- javaコマンドのコマンドライン引数
- mainメソッド

1. プリミティブ型として正しいものを選びなさい。（2つ選択）

A. Integer
B. double
C. Number
D. byte
E. Character

➡ P39

2. 参照型として正しいものを選びなさい。（3つ選択）

A. String
B. Date
C. int
D. boolean
E. char[]

➡ P40

3. 整数値を代入できる型として正しいものを選びなさい。（3つ選択）

A. double
B. char
C. short
D. int
E. float

➡ P40

4. 真偽値を保持できる型として正しいものを選びなさい。（1つ選択）

A. char
B. boolean
C. double
D. int
E. String

➡ P41

5. プリミティブ型および参照型変数の説明として正しいものを選びなさい。（2つ選択）

A. プリミティブ型の変数は配列インスタンスへの参照を保持できる
B. プリミティブ型の変数は文字列を保持できる
C. 参照型の変数はインスタンスへの参照を保持できる
D. 参照型の変数は整数値を保持できる
E. プリミティブ型の変数は数値、文字、真偽値を保持できる

➡ P41

6. 変数の宣言方法として正しいものを選びなさい。（2つ選択）

A. `int number;`
B. `length long;`
C. `String name, code;`
D. `boolean long;`
E. `double 3.7;`

➡ P42

7. 数値を扱う変数の宣言と初期化方法として正しいものを選びなさい。（2つ選択）

A. `long number = 2.5;`
B. `float number = 5;`
C. `byte number = 128;`
D. `double number = 4.3;`
E. `int number = 3.0;`

➡ P43

8. 真偽値を扱う変数の宣言と初期化方法として正しいものを選びなさい。（1つ選択）

A. `boolean flag = TRUE;`
B. `boolean flag = "true";`
C. `boolean flag = false;`
D. `boolean flag = 0;`
E. `boolean flag = 'false';`

➡ P45

9. 文字を扱う変数の宣言と初期化方法として正しいものを選びなさい。（3つ選択）

A. `char c = 'AB';`
B. `char c = "T";`
C. `char c = '¥u1F1C';`
D. `char c = 'U';`
E. `char c = 97;`

➡ P45

10. 定数の宣言方法として正しいものを選びなさい。（2つ選択）

A. `int final a = 10;`
B. `int frozen a = 17;`
C. `frozen int a = 13;`
D. `final int a = 8;`
E. `int a final = 3;`
F. `final int a;`

➡ P46

11. 次のコードをコンパイル、実行したときの結果として、正しいものを選びなさい。（1つ選択）

```
1.  public class Main {
2.      public static void main(String[] args) {
3.          final String COLOR = "blue";
4.          // other code
5.          COLOR = "red";
6.          System.out.println(COLOR);
7.      }
8.  }
```

A. 「blue」と表示される
B. 「red」と表示される
C. コンパイルエラーになる
D. 実行時に例外が発生する

➡ P47