13. Java に関する知識 I

1. 科目の概要

Web アプリケーション構築で利用されることの多い Java 言語について、基本的な仕組み、プログラミング文法、オブジェクト指向によるプログラム設計の概念、各種のアプリケーション開発を解説する。さらに Web アプリケーション開発やデータベースの利用についても述べる。

2. 習得ポイント

本科目の学習により習得することが期待されるポイントは以下の通り。

習得ポイント	説　明	シラバスの対応コマ
I-13-1. Java言語の歴史、特徴、思想と背景	Javaの基本概念と特有な機能、発展の歴史、Javaが登場した背景と必然性、Java利用のメリットなど、Javaプログラミングを理解する前におさえておくべき項目として、Java言語にまつわる様々なトピックを紹介する。	1
I-13-2. Javaを用いたプログラミング方法	Javaプログラムの例を示し、Javaプログラムの記述からコンパイル、実行までの手順を紹介する。さらに様々なJavaのエディションや実行環境の整備など、Javaを用いたプログラミングを実施する際に留意すべき点について述べる。	1
I-13-3. Java言語の基本的な構造、型、演算子、制御構文	Javaの基本的な仕組み、特徴、構造を説明する。基本的なプログラム要素として、識別子、変数、型、クラス、演算子、制御構文について解説する。またアプレットやサーブレットといった形態、クラスライブラリの利用についても言及する。	2
I-13-4. Javaによるオブジェクト指向プログラミング	Javaの大きな特徴であるオブジェクト指向プログラミングについて、その基本的な特徴とメリット、C言語との共通点や違いを説明する。クラス、オブジェクト、メソッド、コンストラクタなどオブジェクト指向の様々な概念を紹介し、オブジェクト指向による代表的なプログラミング手法を示す。	3
I-13-5. Javaによるアプリケーション開発、各種リソースの利用	Javaによるアプリケーション開発の基本的な手順を解説する。既存のドキュメントへアクセスする方法、必要なライブラリの探し方、オブジェクト指向を活用したJavaプログラミング開発技法などを紹介する。	4
I-13-6. Eclipseを用いたJavaプログラミング	統合開発環境Eclipseを用いたJavaプログラミングを概説する。Eclipseの利用方法と活用のメリット、Eclipseのプラグインによる拡張など、Eclipseを活用したJavaアプリケーション開発に役立つ様々な話題についても触れる。	4
I-13-7. Javaによるネットワークプログラミング	Javaによるネットワークプログラミングの概要、特徴を説明する。ネットワークプログラミングの基礎であるソケットを用いたセッション管理について、Javaを例題として説明する。またJavaのRMI (Remote Method Invocation) を活用した分散プログラミングも解説する。	5
I-13-8. ServletとJSPによるWebアプリケーション開発	Webアプリケーションの概要や特徴について説明し、さらにServletとJSPによるWebアプリケーション開発の手順と内容について解説する。また開発プラットフォームとしてのTomcatやJBossといった代表的なOSSの実装を紹介する。	6
I-13-9. JDBCによるデータベースアクセス方法	Webアプリケーションではほぼ必須となるデータベースとの連携を実現する方式として、JDBCを用いたJavaアプリケーションとデータベースの接続方法を説明する。その基本的な概念、プログラミング方法、トランザクション処理などについて解説する。	7
I-13-10. MVCアーキテクチャの基本と特徴	Javaアプリケーションを例題として、Webアプリケーションの基本アーキテクチャとして想定されるMVC (Model-View-Controller)モデルの内容とメリットについて解説する。それぞれを構成する要素技術を示し、モデルとの対応関係について説明する。	8

【学習ガイダンスの使い方】

1. 「習得ポイント」により、当該科目で習得することが期待される概念・知識の全体像を把握する。
2. 「シラバス」、「IT 知識体系との対応関係」、「OSS モデルカリキュラム固有知識」をもとに、必要に応じて、従来の IT 教育プログラム等との相違を把握した上で、具体的な講義計画を考案する。
3. 習得ポイント毎の「学習の要点」と「解説」を参考にして、講義で使用する教材等を準備する。

3. IT 知識体系との対応関係

「13. Java に関する知識Ⅰ」と IT 知識体系との対応関係は以下の通り。

科目名	基本レベル(Ⅰ)								応用レベル(Ⅱ)						
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
13. Javaに関する知識	<Javaの基本>	<Java言語の基本構造>	<オブジェクト指向プログラミングのメリット>	<Javaによるアプリケーション開発手順>	<Javaによるネットワークプログラミング>	<Servlet/JSP/JDBCによるWebアプリケーション開発の概要>	<JDBCによるデータベースアクセス>	<MVCモデル>	<EJBによるアプリケーション開発>	<Javaによる Server処理実装の特徴と設計方法>	<Javaによる Webアプリケーションの設計／実装>	<Javaによる Server Side Java/Webアプリケーション実装>	<オブジェクト指向システム分析／設計／実装の実践技術>	<デザインパターンによる開発手順>	<Javaのパフォーマンスチューニング>

[シラバス：http://www.ipa.go.jp/software/open/ossc/download/Model_Curriculum_05_13.pdf]

<IT 知識体系上の関連部分>

分野		科目名	1	2	3	4	5	6	7	8	9	10	11	12	13
組織関連事項と情報システム	1	IT-IAS 情報保証と情報セキュリティ	IT-IAS1. 基礎的な問題	IT-IAS2. 情報セキュリティの仕組み（対策）	IT-IAS3. 運用上の問題	IT-IAS4. ポリシー	IT-IAS5. 攻撃	IT-IAS6. 情報セキュリティ分野	IT-IAS7. フォレンジック（情報証拠）	IT-IAS8. 情報の状態	IT-IAS9. 情報セキュリティサービス	IT-IAS10. 脅威分析モデル	IT-IAS11. 脆弱性		
	2	IT-SP 社会的な観点とプロフェッショナルとしての課題	IT-SP1. プロフェッショナルとしてのコミュニケーション	IT-SP2. コンピュータの歴史	IT-SP3. コンピュータを取り巻く社会環境	IT-SP4. チームワーク	IT-SP5. 知的財産権	IT-SP6. コンピュータの法的問題	IT-SP7. 組織の中のIT	IT-SP8. プロフェッショナルとしての倫理的な問題と責任	IT-SP9. プライバシーと個人の自由				
応用技術	3	IT-IM 情報管理	IT-IM1. 情報管理の概念と基礎	IT-IM2. データベース問合わせ言語	IT-IM3. データアーキテクチャ	IT-IM4. データモデリングとデータベース設計	IT-IM5. データと情報の管理 [13-Ⅰ-6, 7]	IT-IM6. データベースの応用分野							
	4	IT-WS Webシステムとその技術	IT-WS1. Web技術 [13-Ⅰ-1,6]	IT-WS2. 情報アーキテクチャ	IT-WS3. デジタルメディア	IT-WS4. Web開発	IT-WS5. 脆弱性	IT-WS6. ソーシャルソフトウェア							
ソフトウェアの方法と技術	5	IT-PF プログラミング基礎	IT-PF1. 基本データ構造	IT-PF2. プログラミングの基本的構成要素	IT-PF3. オブジェクト指向プログラミング [13-Ⅰ-3,8]	IT-PF4. アルゴリズムと問題解決	IT-PF5. イベント駆動プログラミング	IT-PF6. 再帰							
	6	IT-IPT 技術を統合するためのプログラミング	IT-IPT1. システム間通信	IT-IPT2. データ割り当てと交換	IT-IPT3. 統合的コーディング [13-Ⅰ-3]	IT-IPT4. スクリプティング手法	IT-IPT5. ソフトウェアセキュリティの実現	IT-IPT6. 種々の問題	IT-IPT7. プログラミング言語の概要						
	7	CE-SWE ソフトウェア工学	CE-SWE0. 歴史と概要	CE-SWE1. ソフトウェアプロセス	CE-SWE2. ソフトウェアの要求と仕様	CE-SWE3. ソフトウェアの設計	CE-SWE4. ソフトウェアのテストと検証	CE-SWE5. ソフトウェアの保守	CE-SWE6. ソフトウェア開発・保守ツールと環境	CE-SWE7. ソフトウェアプロジェクト管理	CE-SWE8. 言語翻訳	CE-SWE9. ソフトウェアのフォールトトレランス	CE-SWE10. ソフトウェアの構成管理	CE-SWE11. ソフトェアの標準化	
	8	IT-SIA システムインテグレーションとアーキテクチャ	IT-SIA1. 要求仕様	IT-SIA2. 調達/手配	IT-SIA3. インテグレーション	IT-SIA4. プロジェクト管理	IT-SIA5. テストと品質保証	IT-SIA6. 組織の特性	IT-SIA7. アーキテクチャ						
システム基盤	9	IT-NET ネットワーク	IT-NET1. ネットワークの基礎	IT-NET2. ルーティングとスイッチング	IT-NET3. 物理層	IT-NET4. セキュリティ	IT-NET5. アプリケーション分野	IT-NET6. ネットワーク管理							
	10	CE-NWK テレコミュニケーション	CE-NWK0. 歴史と概要	CE-NWK1. 通信ネットワークのアーキテクチャ	CE-NWK2. 通信ネットワークのプロトコル	CE-NWK3. LANとWAN	CE-NWK4. クライアントサーバコンピューティング	CE-NWK5. データのセキュリティと整合性	CE-NWK6. ワイヤレスコンピューティングとモバイルコンピューティング	CE-NWK7. データ通信	CE-NWK8. 組込み機器向けネットワーク	CE-NWK9. 通信技術とネットワーク概要	CE-NWK10. 性能評価	CE-NWK11. ネットワーク管理	CE-NWK12. 圧縮と伸張
			CE-NWK13. クラスタシステム	CE-NWK14. インターネットアプリケーション	CE-NWK15. 次世代インターネット	CE-NWK16. 放送									
	11	IT-PT プラットフォーム技術	IT-PT1. オペレーティングシステム	IT-PT2. アーキテクチャと機構	IT-PT3. コンピュータインフラストラクチャ	IT-PT4. デプロイメントソフトウェア	IT-PT5. ファームウェア	IT-PT6. ハードウェア							
	12	CE-OPS オペレーティングシステム	CE-OPS0. 歴史と概要	CE-OPS1. 並行性	CE-OPS2. スケジューリングとディスパッチ	CE-OPS3. メモリ管理	CE-OPS4. セキュリティと保護	CE-OPS5. ファイル管理	CE-OPS6. リアルタイムOS	CE-OPS7. OSの概要	CE-OPS8. 設計の原則	CE-OPS9. デバイス管理	CE-OPS10. システム性能評価		
コンピュータアーキテクチャ	13	CE-CAO コンピュータのアーキテクチャと構成	CE-CAO0. 歴史と概要	CE-CAO1. コンピュータアーキテクチャの基礎	CE-CAO2. メモリシステムの構成とアーキテクチャ	CE-CAO3. インタフェースと通信	CE-CAO4. デバイスサブシステム	CE-CAO5. CPUアーキテクチャ	CE-CAO6. 性能・コスト評価	CE-CAO7. 分散・並列処理	CE-CAO8. コンピュータによる計算	CE-CAO9. 性能向上			
複数領域にまたがるもの	14	IT-ITF IT基礎	IT-ITF1. ITの一般的なテーマ	IT-ITF2. 組織の問題	IT-ITF3. ITの歴史	IT-ITF4. IT分野（学科）とそれに関連のある分野（学科）	IT-ITF5. 応用領域	IT-ITF6. IT分野における数学と統計学の活用							
	15	CE-ESY 組込みシステム	CE-ESY0. 歴史と概要	CE-ESY1. 低電力コンピューティング	CE-ESY2. 高信頼性システムの設計	CE-ESY3. 組込み用アーキテクチャ	CE-ESY4. 開発環境	CE-ESY5. ライフサイクル	CE-ESY6. 要件分析	CE-ESY7. 仕様定義	CE-ESY8. 構造設計	CE-ESY9. テスト	CE-ESY10. プロジェクト管理	CE-ESY11. 並行設計(ハードウェア、ソフトウェア	CE-ESY12. 実装
			CE-ESY13. リアルタイムシステム設計	CE-ESY14. 組込みマイクロコントローラ	CE-ESY15. 組込みプログラム	CE-ESY16. 設計手法	CE-ESY17. ツールによるサポート	CE-ESY18. ネットワーク型組込みシステム	CE-ESY19. インタフェースシステムと混合信号システム	CE-ESY20. センサ技術	CE-ESY21. デバイスドライバ	CE-ESY22. メンテナンス	CE-ESY23. 専門システム	CE-ESY24. 信頼性とフォールトトレランス	

4. OSS モデルカリキュラム固有の知識

OSS モデルカリキュラム固有の知識として、Java 言語の特徴、記述、開発手法、および Web やデータベースを扱う手法がある。Web に関してはサーバ側のサービスを実現する Servlet や JSP を、データベースに関してはドライバである JDBC を内容として含む。

科目名	第1回	第2回	第3回	第4回	第5回	第6回	第7回	第8回
13.Java に関する知識Ⅰ	(1)Java プログラムの例	(1)Java 言語の書き方の特徴	(1)オブジェクト指向プログラミング機能の実装仕様	(1)Java の開発手順	(1)ネットワークプログラミングの概要	(1)Web アプリケーション	(1)JDBC の概念と接続形態	(1)MVC アーキテクチャとは
	(2)Java のオープンソースとしての特徴	(2)基本的なプログラム記述の例	(2)オブジェクト指向を活用したプログラミング手順	(2)統合開発環境 Eclipse		(2)Servlet	(2)JDBC プログラミング	(2)MVC アーキテクチャのメリット
		(3)アプレット				(3)JSP	(3) PreparedStatement の利用方法とそのメリット	(3)Servlet／JSP／JavaBeans の役割
		(4)入出力				(4)開発プラットフォーム	(4)トランザクション(Transaction)処理	(4)システム機能／非機能要件への対応
		(5)例題					(5)データソース、JDBC	

（網掛け部分は IT 知識体系で学習できる知識を示し、それ以外は OSS モデルカリキュラム固有の知識を示している）

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	13　Java に関する知識 I	基本
習得ポイント	I-13-1. Java 言語の歴史、特徴、思想と背景	
対応する コースウェア	第 1 回（Java の基本）	

I-13-1. Java 言語の歴史、特徴、思想と背景

Java の基本概念と特有な機能、発展の歴史、Java が登場した背景と必然性、Java 利用のメリットなど、Java プログラミングを理解する前におさえておくべき項目として、Java 言語にまつわる様々なトピックを紹介する。

【学習の要点】

* Java はサン・マイクロシステムズ社が 1996 年に発表した言語であり、現在ではオープンソースとして開発が続けられている。
* Java には、オブジェクト指向、ポインタの廃止、ガベージコレクションを装備等の特徴がある。
* Java は仮想マシンと中間言語を用い、Write Once、 Run Anywhere を実現している。
* 現在 Java は非常に広い分野で扱われており、技術者としては必須ともいえる言語である。

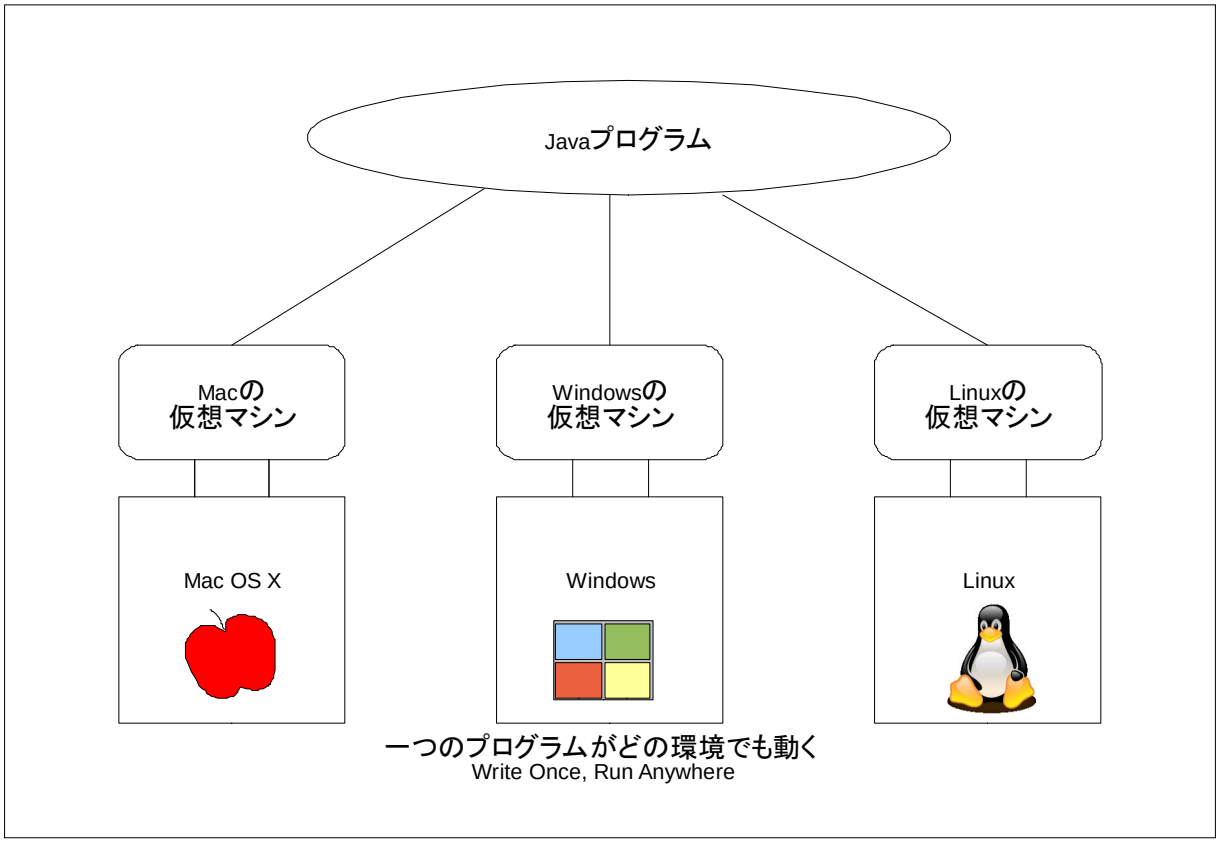

図 I-13-1.　Java プログラムの実行環境

【解説】

1）Java の歴史

Java はサン・マイクロシステムズ社によって開発され、1996 年に正式発表された。
初期は Web ブラウザ上で動く、アプレットという動的な Web ページを実現するアプリケーションとして使われていたが、このアプレットとしての役割は Flash などに取って代わられた。しかしその後、Java SE といった基本的な開発環境に加え、Java EE といったサーバサイドのアプリケーション開発に特化した開発環境や、Java ME といった組み込み機器用の開発環境がリリースされた。そして今では、主にサーバサイドの Web アプリケーションや、携帯電話等に搭載されている組み込み機器技術など、非常に幅広い分野で利用されている。

2）Java の特徴

＊ 仮想マシンにより環境を問わず実行できるため、環境に依存しないアプリケーションが作れる。
＊ オブジェクト指向言語であり、コードの生産性が高い。
＊ ガベージコレクションを備えており、メモリ管理の手間が非常に少ない。
＊ ポインタを廃止したことにより、不正なメモリ参照が起きなくなった。
＊ 多重継承を禁止しているため、メソッド呼び出しの際のあいまいさが排除された。
＊ 静的な型づけを採用しているため、変数に型自体からして違う値などが入って起こるバグを最小限に抑えられる。
＊ コンパイルしたファイルは中間言語に変換され、Java 仮想マシン上で実行される。

3）Java の思想と背景

＊ Java の思想を表す言葉で、『Write Once, Run Anywhere』という言葉がある。これは、プログラムを一度作ってしまえばそのプログラムがどこの環境でも動くということを意味する。
＊ 大人数でプログラムを作る際、オブジェクト指向であることにより各自が作ったソースを統合するのが簡単になる。
＊ メモリ管理は非常に複雑でバグの原因やパフォーマンスの低下につながっていたが、ガベージコレクションによってそれらの問題が軽減される。
＊ ポインタを廃止したことで、ポインタ起因のバグが出なくなった。
＊ オブジェクトの扱いのあいまいさや複雑さを防ぐため、多重継承を禁止した。
＊ バグの原因の多くは変数に意図しない値が入ることだが、それを防ぐために静的な型づけが採用された。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	13　Java に関する知識 I	基本
習得ポイント	I-13-2. Java を用いたプログラミング方法	
対応する コースウェア	第 1 回 (Java の基本)	

I-13-2. Java を用いたプログラミング方法

Javaプログラムの例を示し、Javaプログラムの記述からコンパイル、実行までの手順を紹介する。さらに様々な Java のエディションや実行環境の整備など、Java を用いたプログラミングを実施する際に留意すべき点について述べる。

【学習の要点】

* Java の開発環境を整えるには、通常 Sun のサイトから Java SE Development Kit (JDK)をダウンロードする。これは無償で利用できる。
* Java のソースコードをコンパイルすると、中間言語で記述されたクラスファイルが作成され、それを仮想マシン上で実行することができる。

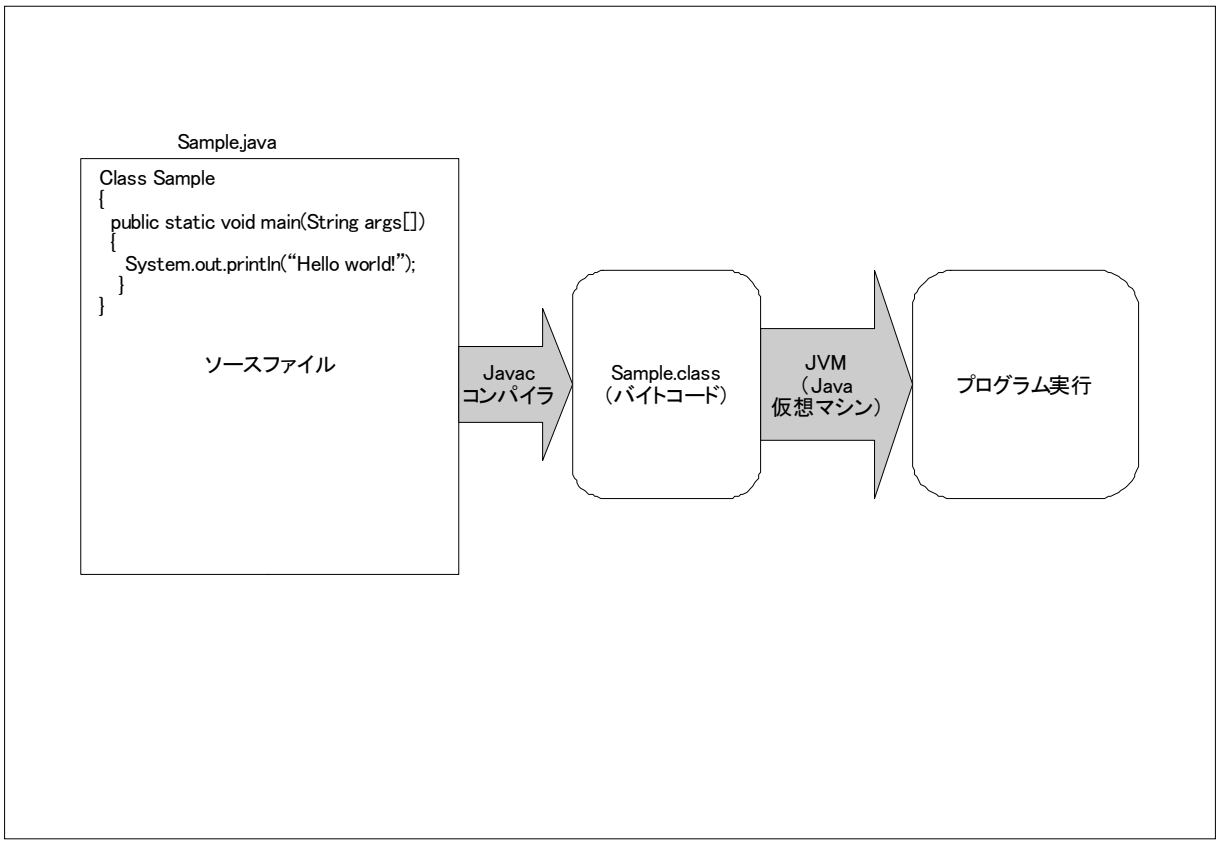

図 I-13-2.　Java プログラミングの流れ

【解説】

1) Java プログラミング環境の整え方

Sun の Java のページ、http://java.sun.com/にアクセスして、JDK (Java SE Development Kit) をダウンロードする。特に理由がなければ、最新版をダウンロードするとよい。通常の Java アプリケーションを開発したい場合は Java SE SDK を、Web 用の Java アプリケーションを開発したい場合は Java EE SDK を、モバイル用の Java アプリケーションを開発したい場合は Sun Java Wireless Toolkit を、ダウンロードする。通常は Java SE SDK をダウンロードすればよい。

2) Java プログラムの記述の仕方

* プログラムは、テキストエディタで記述する。
* プログラム中でコメントとして扱われる部分は次の 2 通りがある。
  - 「//」から行末までの間
  - 「/*」から「*/」までの間
* プログラム中の各命令は、一般に「;」で終わる。
* ファイルを保存するときは、拡張子を「.java」とする。

3) Java プログラムのコンパイルの仕方

Linux の場合、以下のような手順でコンパイルをする。

* ターミナルを起動しソースファイルが保存されているディレクトリに移動する。
* javac <ソースファイル名>と入力してコンパイルを実行する。ソースファイル名には、.java まで含める。
* ソースファイルが保存されているディレクトリに、 .class という拡張子が付いたファイルが作成されていることを確認する。

4) Java プログラムの実行の仕方

Linux の場合、以下の手順でコンパイルしたファイルを実行する。

* ターミナルを起動し、クラスファイルが保存されているディレクトリに移動する。しかし、コンパイルをした直後で既にラスファイルが保存されているディレクトリにいるなら移動する必要はない。
* java <クラス名>と入力して JVM を起動する。例えば、クラスファイルが Sample.class という名前だったら、java Sample と入力する。最後の .class は含めないことに注意する。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	13　Java に関する知識 I	基本
習得ポイント	I-13-3. Java 言語の基本的な構造、型、演算子、制御構文	
対応する コースウェア	第 2 回（Java 言語の基本構造）	

I-13-3. Java 言語の基本的な構造、型、演算子、制御構文

Java の基本的なしくみ、特徴、構造を説明する。基本的なプログラム要素として、識別子、変数、型、クラス、演算子、制御構文について解説する。またアプレットやサーブレットといった形態、クラスライブラリの利用についても言及する。

【学習の要点】

* Java には、識別子、変数、型、クラス、演算子、制御構文が基本的な仕組みとして存在する。
* Java プログラムには Web ブラウザ上で動作するアプレットや Web サーバー上で動作するサーブレットといった形態がある。
* Java にはクラスライブラリが用意されており、それを利用することでプログラムの生産性の向上が見込める。

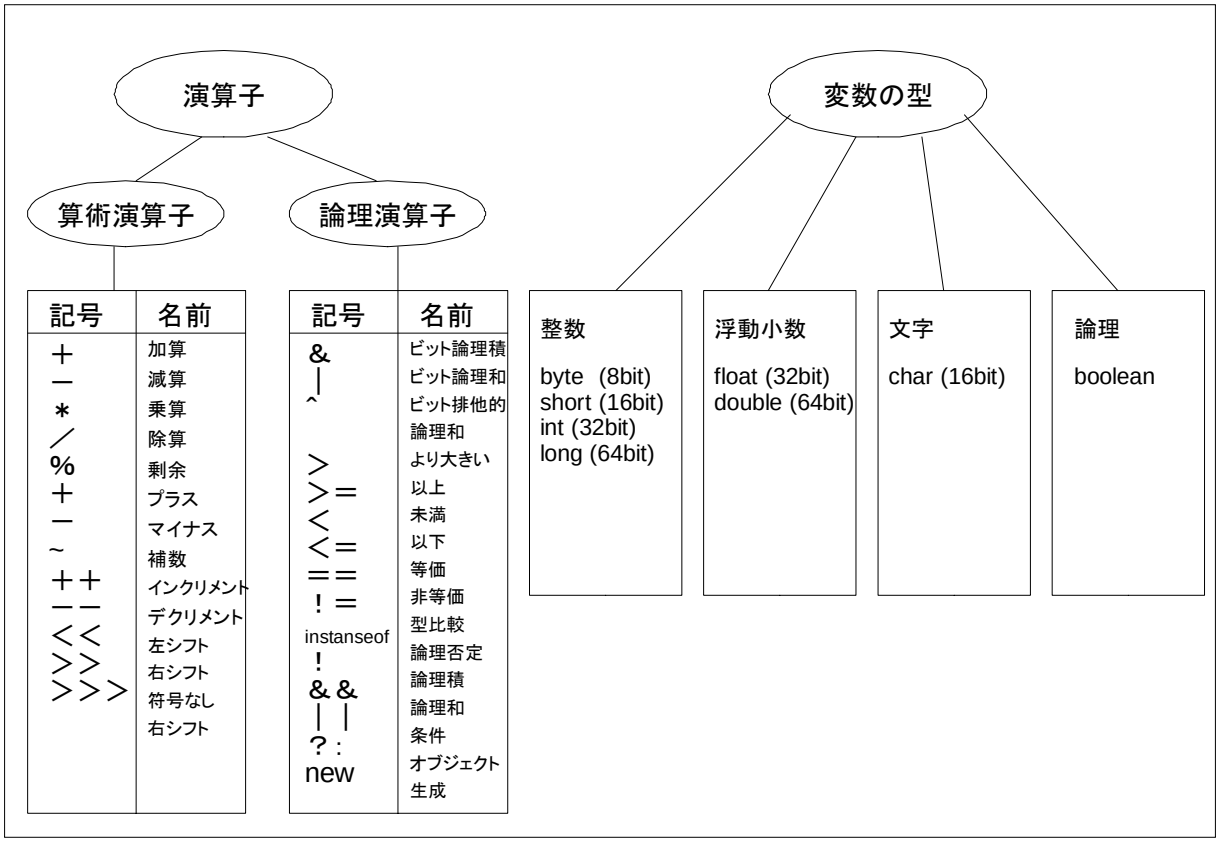

図 I-13-3.　演算子と変数の種類

【解説】

1）識別子とは

Java でいう識別子とは、変数などを人間が読みやすい英数字などの名前で識別するものである。void、class など Java の予約語は識別子として用いることはできない。

2）Java の型

Java は静的な型づけを採用しているため、変数を用いる際には必ず型の宣言が必要になる。

3）演算子について

プログラムは主に値を計算することによって様々な機能を果たす。値をどのように計算するのかを命令するためのものが演算子である。

4）制御構文

Java には制御構文が用意されている。これらの制御構文にデータを渡すと、そのデータの値によってプログラムの挙動が変わるようにすることができる。主な制御構文とその働きは以下のとおりである。

* if ～ else if ～ else：ある条件に当てはまった場合と、当てはまらなかった場合の挙動を制御する。
* switch：複数の条件とそれに対する挙動を制御できる。
* for：指示した回数だけ挙動を繰り返す。
* while：指示した条件に当てはまっている間だけ挙動を繰り返す。
* do ～ while：まず一度は挙動が実行され、その後指示した条件に当てはまっている間だけ挙動を繰り返す。

5）Java プログラムの種類

Java プログラムの実行形態にはコマンドラインで実行されるコンソールプログラム、Java アプレット、Java サーブレットといった種類がある。
Java アプレットはネットワーク経由でダウンロードされ、クライアント（Web ブラウザ）で実行される Java プログラムのことである。現在では Adobe Flash などの普及によりあまり利用されなくなった。
Java サーブレットは Web サーバ上で実行される Java プログラムである。Perl や Ruby などスクリプトで処理される CGI と比べると、様々なオーバーヘッドが無いために動作が速いという特徴がある。

6）クラスライブラリ

Java のプログラムをを書くときに、すべてのクラスをいちから定義しなくとも、よく使われるクラスに関しては、クラスライブラリによってすでに用意されているものを利用すればよい。クラスライブラリを利用することによりコードの生産性を高めることができるうえ、十分にテストされたライブラリを利用すればバグを減らすことも可能となる。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	13　Java に関する知識 I	基本
習得ポイント	I-13-4. Java によるオブジェクト指向プログラミング	
対応する コースウェア	第 3 回（オブジェクト指向プログラミングのメリット）	

I-13-4. Java によるオブジェクト指向プログラミング

Java の大きな特徴であるオブジェクト指向プログラミングについて、その基本的な特徴とメリットを説明する。クラス、オブジェクト、メソッド、コンストラクタなどオブジェクト指向の様々な概念を紹介し、オブジェクト指向による代表的なプログラミング手法を示す。

【学習の要点】

* オブジェクト指向プログラミングとは、データと関数を同時に扱うことによってコードの独立性を強くし、疎結合なプログラムを設計するプログラミング技法である。
* クラスはフィールドとメソッドによって成り立っている、ひな形の概念である。
* オブジェクトとはモノの概念であるクラスに実際の値を与えて作られる、クラスを実体化したものである。
* コンストラクタはオブジェクトを初期化する際に利用する。

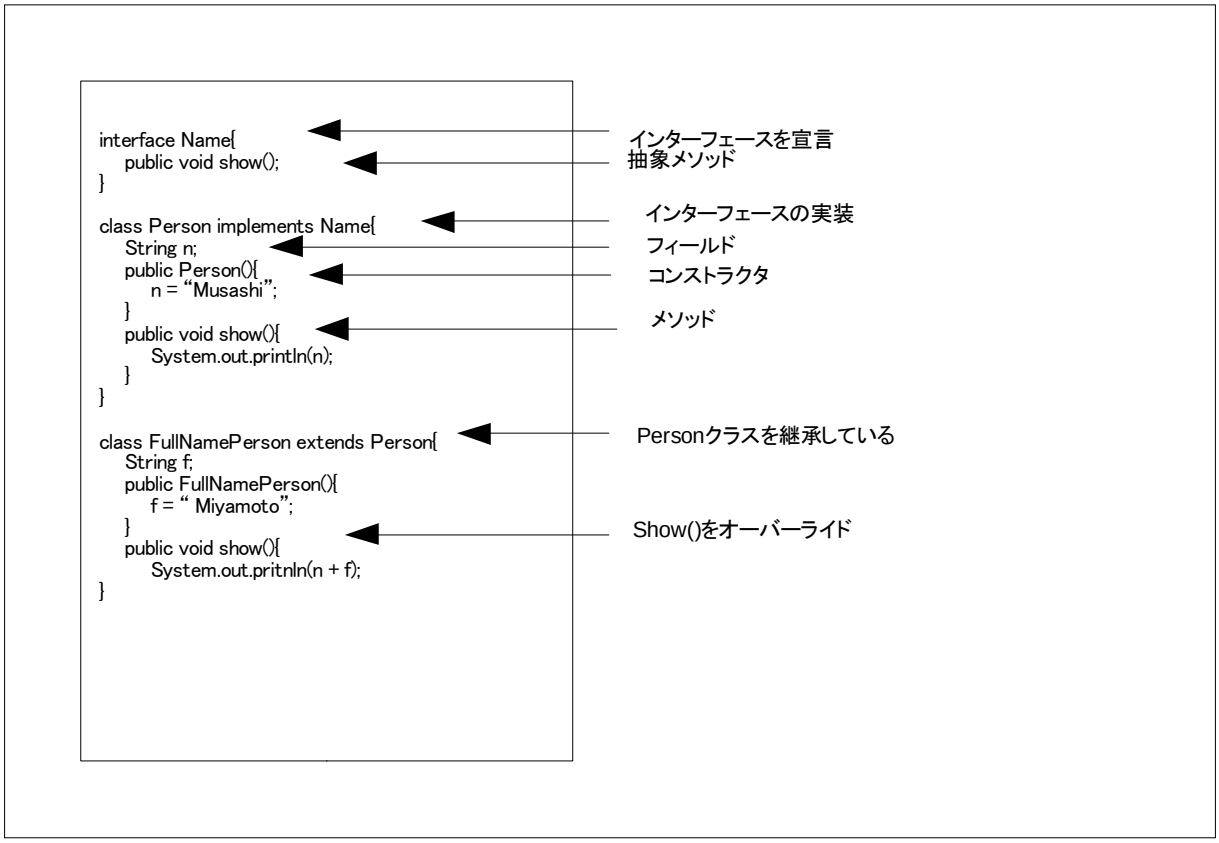

図 I-13-4.　オブジェクトを使用したプログラミング例

【解説】

1) オブジェクト指向プログラミングとは

オブジェクト指向プログラミングとは、データと関数を同時に扱うことによってコードの独立性を強くし、疎結合なプログラムを設計するためのプログラミング技法である。オブジェクト指向により、コードの変更がしやすい柔軟なプログラムを作ることができる。

2) クラスとは

クラスとは、対象とするものに付随させる変数や関数(処理のまとまり)を定義したものである。クラスの変数をフィールド、クラスの関数をメソッドと呼ぶ。また、フィールドやメソッドを、そのクラスのメンバと呼ぶ。例えば Person というクラスを定義したい場合は以下のように記述する。

例)

```
class Person {
    フィールドの宣言
    :
    メソッドの宣言と処理の記述
    :
}
```

3) オブジェクト(インスタンス)とは

クラスの性質や機能を持った実体のことをオブジェクトという。

オブジェクトを利用するためには、次の処理を行う。

* 定義したクラスを型とする変数を宣言する。
* 定義したクラスのオブジェクトを生成し、宣言した変数にそのオブジェクトを代入する。

例)

```
Person person1;
person1 = new Person();
```

オブジェクトは、一つのクラスから複数作ることができる。クラスという型からオブジェクトという実体が複数生成されると考えれば分かりやすい。

4) コンストラクタとは

コンストラクタとは、オブジェクトの生成時に自動実行される処理を指す。Java では、クラス定義の中で、クラス名と同じ名前で処理を定義することで、その処理がコンストラクタとして扱われる。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	13　Java に関する知識 I	基本
習得ポイント	I-13-5. Java によるアプリケーション開発、各種リソースの利用	
対応するコースウェア	第 4 回（Java によるアプリケーション開発手順）	

I-13-5. Java によるアプリケーション開発、各種リソースの利用

Java によるアプリケーション開発の基本的な手順を解説する。既存のドキュメントへアクセスする方法、必要なライブラリの探し方、オブジェクト指向を活用した Java プログラミング開発技法などを紹介する。

【学習の要点】

* 標準 API のリファレンスは Sun の公式ページを参照する。
* リファレンスはクラスライブラリを配布しているサイトから入手、もしくはそのサイト上の Web ページを検索することで見つける。
* 開発技法として、デザインパターンを調べる方法をマスターする。
* Java の標準 API、クラスライブラリは JDK がインストールされている環境ならどこでも利用することができる。

図 I-13-5.　Java SE の API ドキュメント（http://java.sun.com/javase/ja/6/docs/ja/api/index.html）

【解説】

1) Java のリソース、リファレンス

Java のリソースおよびリファレンスが載っているサイトには、以下のようなものがある。

* Java Developer Connection
  http://sdc.sun.co.jp/java/
  Java テクノロジーについての技術的な情報を掲載したサイト。
* Java Platform, Standard Edition (Java SE) 6 RC
  http://java.sun.com/javase/ja/6/
  Java SE 6 についてのドキュメントが掲載されているサイト。
* The Jakarta Project
  http://jakarta.apache.org/ (英語)
  主に Web 関連の Java のオープンソースプロジェクトがまとめられているサイト。
* SourceForge.net
  http://sourceforge.net/
  様々なオープンソースプロジェクトがまとめられているサイト。
* The Codehaus
  http://codehaus.org/
  Java のオープンソースプロジェクトがまとめられているサイト。

2) Javadoc

Java のライブラリには、必ずリファレンスが用意されている。これを調べることで、そのライブラリの利用方法などが分かるようになっている。新しいライブラリを手に入れたら、まずそのリファレンスを調べてみるとよい。これらのリファレンスは、javadoc と呼ばれるコマンドで作成されている。

3) デザインパターン

Java や C++ などのオブジェクト指向型プログラミングが利用されるにつれ、そのプログラムの設計方法のノウハウが蓄積されてきた。それらのノウハウをまとめたものを総称してデザインパターンという。

重要なノウハウとして、再利用しやすいクラスを設計するための工夫がある。クラスの再利用性が高ければ、再利用性の高さが生産性の向上に直結するからである。それらのデザインパターンを正しく活用することによって、プログラミングの初学者でも再利用性の高いコードを作ることができる。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	13　Java に関する知識 I	基本
習得ポイント	I-13-6. Eclipse を用いた Java プログラミング	
対応する コースウェア	第 4 回（Java によるアプリケーション開発手順）	

I-13-6. Eclipse を用いた Java プログラミング

統合開発環境 Eclipse を用いた Java プログラミングを概説する。Eclipse の利用方法と活用のメリット、Eclipse のプラグインによる拡張など、Eclipse を活用した Java アプリケーション開発に役立つ様々な話題についても触れる。

【学習の要点】

* Eclipse は統合開発環境であり、ソースの記述、ソースをコンパイル、プログラムのテスト実行、プログラムのデバッグが一つのアプリケーション上で行える。非常に効率よくプログラム開発ができる。
* Eclipse にはコード補完や自動コンパイルなど、開発効率を上げる機能が付いている。
* Eclipse のプラグインによって、さらに便利に Eclipse が使えるようカスタマイズすることが可能である。

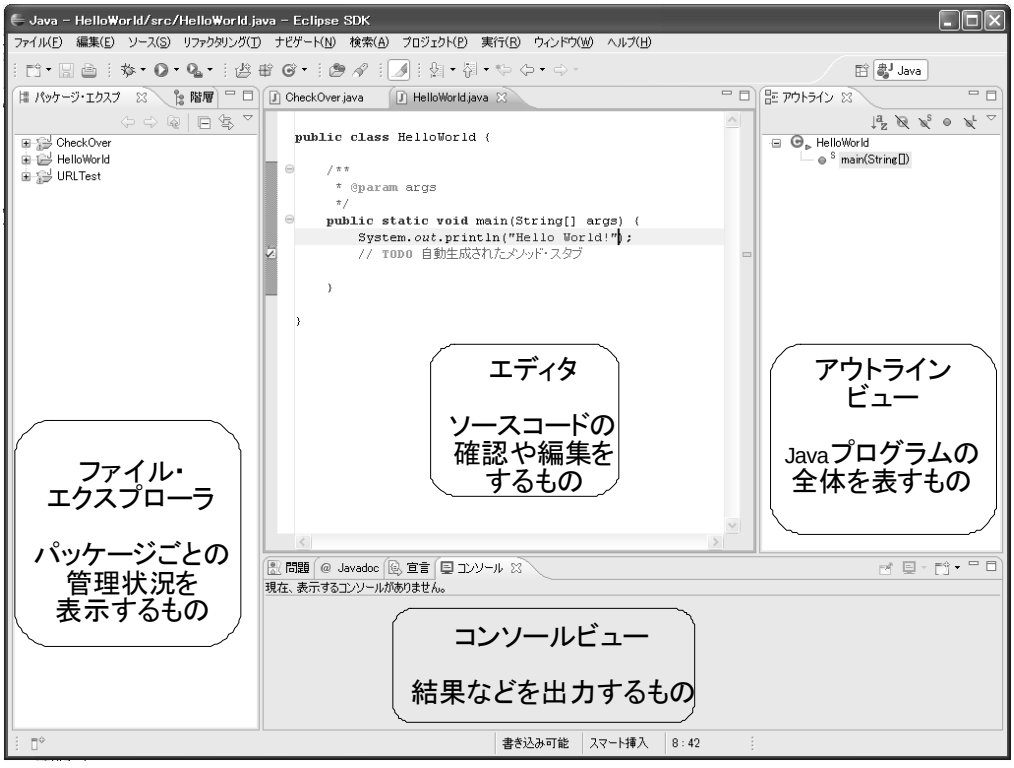

図 I-13-6.　Eclipse のスクリーンショット

【解説】

1) Eclipse の特徴

Eclipse は Java で作られた統合開発環境(IDE)である。その主な特徴は以下のとおりである。

* オープンソース
* プラグインにより Java に限らずさまざまな言語の開発が可能
* テストツールなど、豊富な各種プラグインが無料で利用できる
* グラフィカルなデバッグツールが利用でき、初心者でもデバッグがしやすい
* コード編集支援機能を搭載した、プログラム開発に特化したエディタが使用できる

2) Eclipse による Java プログラミングの流れ

* メニューバーのファイルから新規プロジェクトを選択し、プロジェクト名を入力して新しいプロジェクトを作る。
* パッケージエクスプローラーからプロジェクトの下にある src フォルダを右クリックして、新規→クラスを選択、クラス名を入力して新しいクラスを作る。
* テキストエディタにソースを入力するためのエディタが開くので、そこにソースを記述する。
* ソースを書き終わったら保存をし、ウインドウ上部にある緑の丸に白い三角形が描かれた実行ボタンを押すと、自動的にソースをコンパイルし、コンパイルされたプログラムの実行が行われる。

3) Eclipse のプラグイン

Eclipse は世界中の開発者が作った豊富なプラグインによって、機能をカスタマイズすることができる。プラグインを利用することによって、Java に限らず C 言語や C++ 、Ruby、PHP の開発もできるようになる。ここでは、代表的なプラグインを紹介する。

* WTP(Web Tools Platform)
  Web アプリケーション開発に有用なプラグイン。
* Pleiades
  Eclipse プラグインを日本語化するためのプラグイン。
* FindBugs
  バグが存在する可能性のあるコードを指摘するプラグイン。
* Visual Editor
  GUI アプリケーション開発に有用なプラグイン。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	13　Java に関する知識 I	基本
習得ポイント	I-13-7. Java によるネットワークプログラミング	
対応する コースウェア	第 5 回（Java によるネットワークプログラミング）	

I-13-7. Java によるネットワークプログラミング

Java によるネットワークプログラミングの概要、特徴を説明する。ネットワークプログラミングの基礎であるソケットを用いたセッション管理について、Java を例題として説明する。また Java の RMI (Remote Method Invocation)を活用した分散プログラミングも解説する。

【学習の要点】

＊ ネットワークプログラミングとは、複数のサーバーやクライアント間で情報をやり取りする働きを持つアプリケーションを作ることである。

＊ Java によるネットワークプログラミングの特徴は、豊富なライブラリにより比較的短いコードでネットワークアプリケーションを作ることができる。

＊ ソケットを用いたセッション管理とは、クライアントとサーバーの間の接続を開始から終了まで確立するものである。

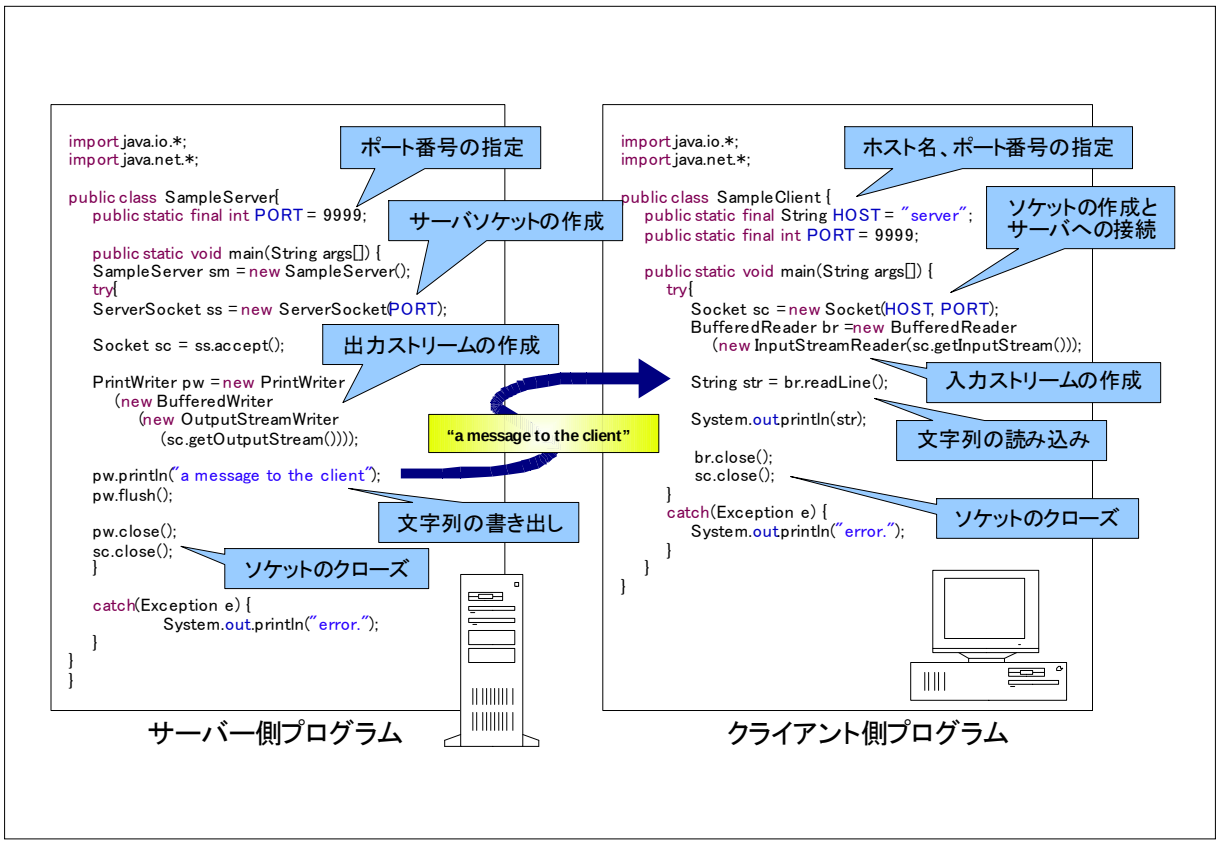

図 I-13-7.　ソケットによる通信

【解説】

1) ソケットを用いたセッション管理

ソケットは、ホスト名または IP アドレスと、TCP ポート番号とを利用して、リモートホストに接続するための仕組みである。
接続が確立されてから切断されるまでの間を、セッションという。TCP/IP の通信においては、サーバとクライアント間の通信が、他のクライアントとの通信と混じらないように、セッション管理の機構が必須である。

2) Java によるネットワークプログラミング

URL、IP アドレスなどのネットワークの基礎知識は、Java を用いてネットワークプログラムを書くときに必要になる知識である。これらを扱うクラスライブラリが、java.net パッケージである。ソケット関連のクラスは以下のようなものがある。

* ServerSocket
  サーバーソケットのクラス。ポート番号を指定して生成すると、そのポートに対してサーバーソケットが作られる。
* Socket
  ソケット接続のクラス。サーバでは ServerSocket クラスの accept()メソッドにより待ち受けとして生成される。クライアントではホストとポート番号とを指定して生成する。
* InputStream
  入力ストリームのクラス。Socket クラスの getInputStream()メソッドにより生成される。
* InputStreamReader
  入力ストリームからの読み取りのクラス。InputStream クラスのオブジェクトを指定して生成する。
* BufferedReader
  読み取りバッファのクラス。InputStreamReader クラスのオブジェクトを指定して生成する。
* OutputStream
  出力ストリームのクラス。Socket クラスの getOutputStream()メソッドにより生成される。
* OutputStreamWriter
  出力ストリームへの書き込みのクラス。OutputStream クラスのオブジェクトを指定して生成する。
* BufferedWriter
  書き込みバッファのクラス。OutputStreamWriter クラスのオブジェクトを指定して生成する。

3) RMI によるネットワークプログラミング

Java を使ってネットワークプログラミングをするには、RMI（Remote Method Invocation)を用いる方法もある。RMI を使うことによって、トランスポート層を意識しなくてもネットワークプログラミングができるようになった。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	13　Java に関する知識 I	基本
習得ポイント	I-13-8. Servlet と JSP による Web アプリケーション開発	
対応する コースウェア	第 6 回（Servlet/JSP/JDBC による Web アプリケーション開発の概要）	

I-13-8. Servlet と JSP による Web アプリケーション開発

Servlet と JSP による Web アプリケーション開発の手順と内容について解説する。また開発プラットフォームとしての Tomcat や JBoss といった代表的な OSS の実装を紹介する。

【学習の要点】

＊ Servlet、JSP は Java で作られた Web アプリケーションである。

＊ Tomcat や JBoss などは Web コンテナと呼ばれる、Web アプリケーションの実行環境である。

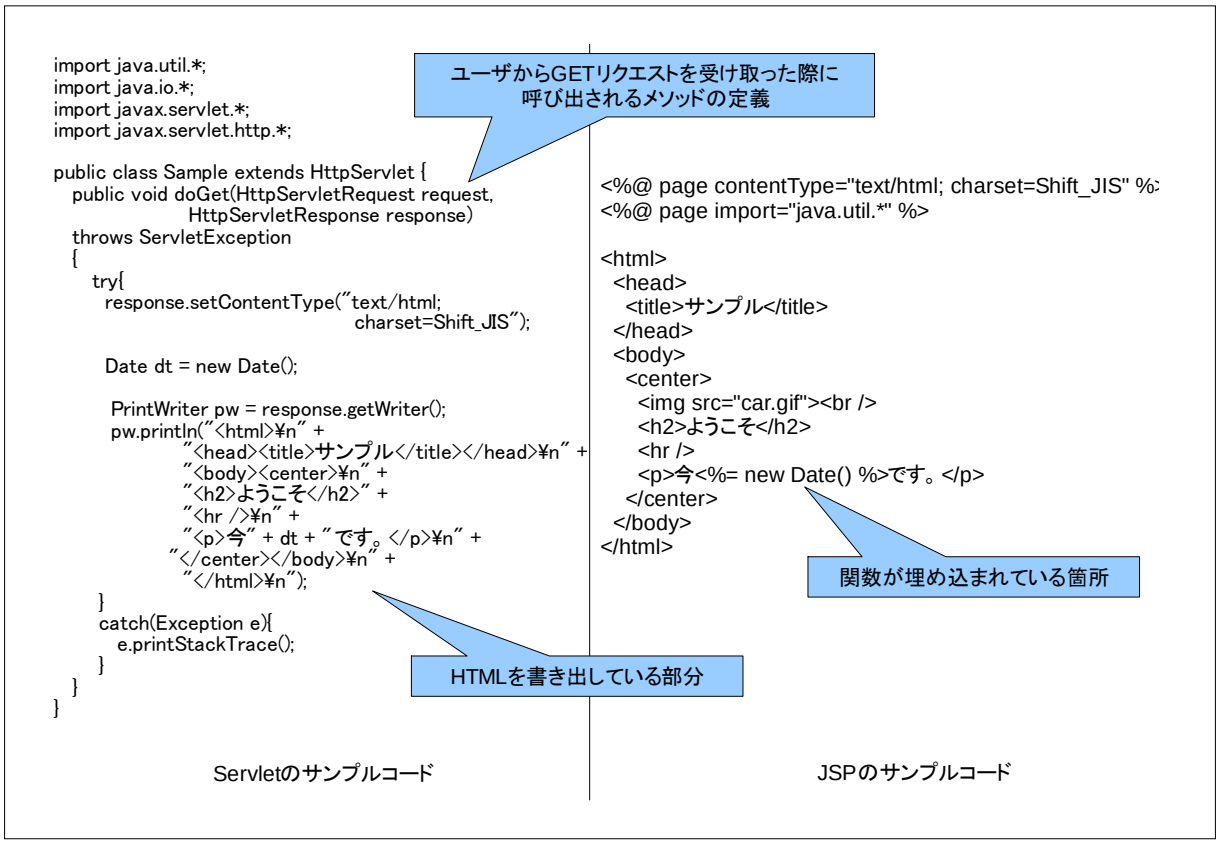

図 I-13-8.　Servlet と JSP のサンプルコード

【解説】

1) Servlet と JSP

Servlet も JSP も、共に Java で作られたユーザーからのテキスト入力やボタン入力に応じて Web ページを生成して返す Web アプリケーションの一種である。その特徴は、次のとおりである。

＊ Servlet
Java により動的なページを返すプログラム。Perl や Python で記述される CGI と比べると、サーブレットはメモリに常駐し待機しているため動作が高速という優位性を持つ。
その動作の概要を以下に示す。

- クライアントからページのリクエストを受ける。
- Web ページがサーブレットによって動的に生成される。
- 生成された Web ページがリクエストに対するレスポンスとして返される。

＊ JSP
JSP は HTML の中にプログラムが書かれている。クライアントが JSP ページをリクエストすると、Web コンテナが JSP から Servlet のコードを呼び出して実行する。明示的なコンパイルは不要。
その動作の概要を以下に示す。

- クライアントから JSP ページのリクエストを受ける。
- JSP ページの中のプログラムから、Servlet を呼び出す。
- JSP ページをコンパイルする。
- ユーザーにコンパイルしたページを返す。

2) 開発プラットフォーム

代表的なオープンソースの開発プラットフォームとして、以下のようなものがある。

＊ Apache Tomcat
http://tomcat.apache.org/
サーブレットの実行環境である Web コンテナ。

＊ JBoss Application Server
http://labs.jboss.com/jbossas/
クライアントとバックエンドの間にあるアプリケーションサーバ。

＊ Jetty
http://jetty.mortbay.org/
Web サーバーでありサーブレットコンテナでもあるソフト。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	13　Java に関する知識 I	基本
習得ポイント	I-13-9. JDBC によるデータベースアクセス方法	
対応する コースウェア	第 7 回（JDBC によるデータベースアクセス）	

I-13-9. JDBC によるデータベースアクセス方法

Web アプリケーションではほぼ必須となるデータベースとの連携を実現する方式として、JDBC を用いた Java アプリケーションとデータベースの接続方法を説明する。その基本的な概念、プログラミング方法、トランザクション処理などについて解説する。

【学習の要点】

* JDBC とは、Java からデータベースを操作する時に使う API である。
* JDBC では、クエリとして SQL 文を渡し、データベースからその結果が返ってくる。
* データベースを利用するには、JDBC ドライバを用意してから接続し、用件が終わったら接続を閉じる。

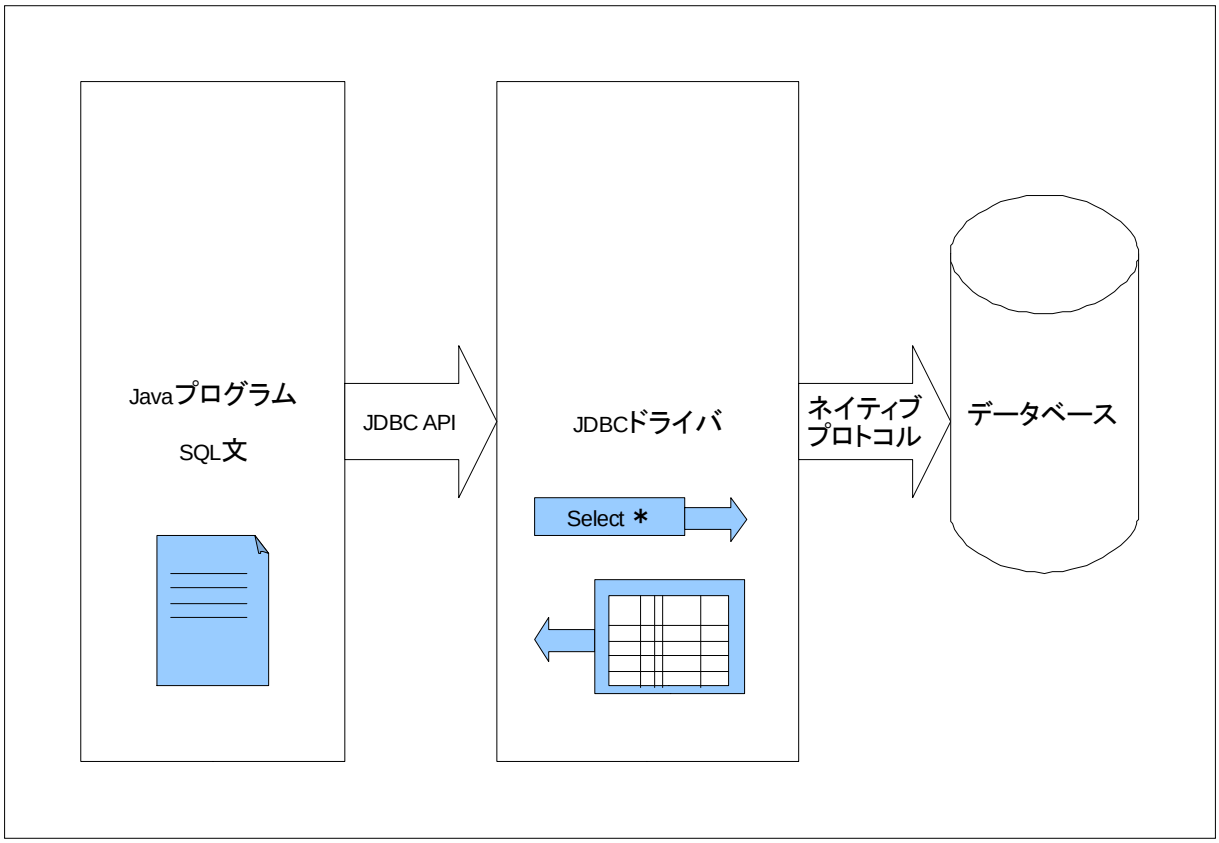

図 I-13-9.　JDBC によるデータベースアクセスの概要

【解説】

1) JDBC とは

JDBC とは、Java プログラムとデータベースを接続するための API である。JDBC を使えば、SQL 文をリレーショナルデータベースに送信することができる。
しかし、現実には各データベースによって必要とされるプロトコルは異なっている。これを解消するために、データベースごとにデータベースと JDBC API 間の差異を吸収する JDBC ドライバが必要になる。

2) JDBC ドライバの種類

JDBC ドライバにはその実装方法ごとに、大きく分けて4種類ある。

* JDBC-ODBC ブリッジ・ドライバ
* ネイティブ・ブリッジ・ドライバ
* ネット・プロトコル・ドライバ
* ネイティブ・プロトコル・ドライバ

3) データベースを扱う手順

* DriverManager クラスを利用し、getConnection()メソッドにデータベース接続情報を指定して、データベースに接続する。Connection クラスのオブジェクトが生成される。
* Connection オブジェクトの createStatement()メソッドで、SQL 文を扱う Statement クラスのオブジェクトを生成する。
* Statement オブジェクトの executeQuery()メソッドに SQL 文を指定して、SQL を発行する。結果セットが ResultSet クラスのオブジェクトに返される。
* Statement オブジェクト、Connection オブジェクトを close()メソッドでクローズする。

4) トランザクション処理について

トランザクションを適用することで、トランザクションの開始から終了までのすべてのデータ変更操作を、すべて反映(コミット)するか、すべて取り消し(ロールバック)することができ、トランザクション前後でのデータ整合性を確保できる。
JDBC でトランザクション管理をする場合は以下のような方法を用いる。

* トランザクションの開始
  JDBC ではデフォルトで自動コミットとなっているので、トランザクションを明示的に扱うには、Connection オブジェクトの setAutoCommit()メソッドに「false」を指定して、自動コミットを解除する。トランザクションの開始は自動で行われる。
* コミット
  Connection オブジェクトの commit()メソッドを実行する。
* ロールバック
  Connection オブジェクトの rollback()メソッドを実行する。

スキル区分	OSS モデルカリキュラムの科目	レベル
プログラミング分野	13　Java に関する知識 I	基本
習得ポイント	I-13-10. MVC アーキテクチャの基本と特徴	
対応する コースウェア	第 8 回 (MVC モデル)	

I-13-10. MVC アーキテクチャの基本と特徴

Java アプリケーションを例題として、Web アプリケーションの基本アーキテクチャとして想定される MVC (Model-View-Controller)モデルの内容とメリットについて解説する。それぞれを構成する要素技術を示し、モデルとの対応関係について説明する。

【学習の要点】

* MVC モデルとは、プログラムをモデル(Model)、ビュー(View)、コントローラー(Controller)という3つに分けて設計する手法のことをいう。
* Web アプリケーションを作る際にも、MVC モデルが適用できる。
* MVC モデルを用いることで、作業の分業が明確になり、開発効率の向上が望める。

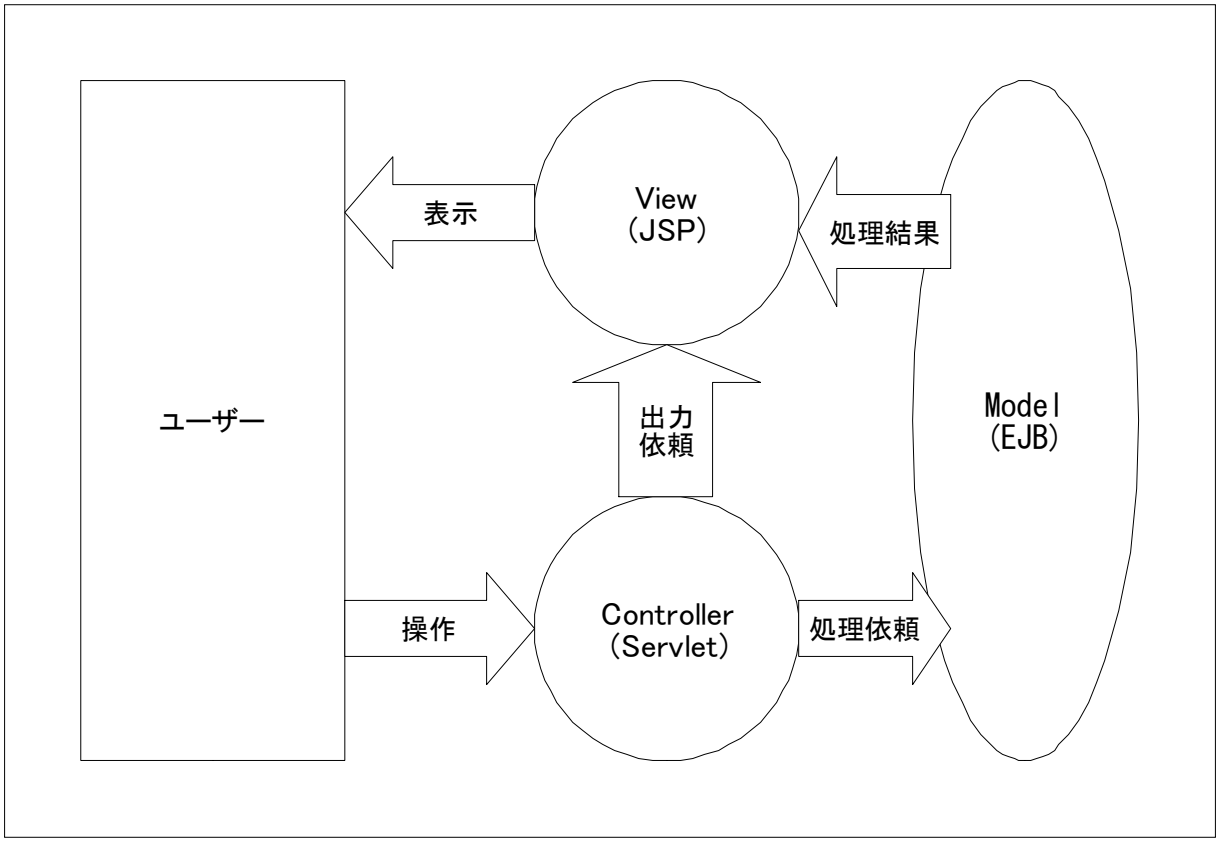

図 I-13-10.　Java における MVC モデルの概要

【解説】

1) MVC モデルとは

MVC モデルとは、プログラムをモデル(Model)、ビュー(View)、コントローラー(Controller)という3つに分けて設計する手法である。

2) Model、View、Controller

* Model (EJB)
  Model はアプリケーションの根幹をなす部分である。アプリケーションの主たる処理内容とデータを表す。特定の環境に依存しない部分のため、EJB の担当箇所とする設計が適切である。
* View (JSP)
  View はアプリケーションのユーザーインタフェースを定義する部分である。Web アプリケーションを例にとるならば、フォームやボタンなどの配置を決めてデザインを作る作業が、この View に当たる。View は HTML を生成しクライアントに送るもののため、JSP 担当の設計が適切である。
* Controller (Servlet)
  Controller は View と Model の間を取り持つ部分である。View のフォームなどから入力された値を受け取り、それを Model に送る。クライアントからの入力は JSP でも担当することができるが、Controller にもロジックが必要なので、サーブレットに担当させる設計が適切である。

3) MVC モデルのメリット

このような手法を採用するメリットは、以下のようなものがある。

* デザイン作り、アプリケーションの根幹、表示とアプリケーションの根幹の間を取り持つ部分といったように開発作業の分業が明確にすることで、開発効率が上がる。
* Model、View、Controller のそれぞれの独立性が高いので、仕様変更に対しても柔軟に対応することができる。
* ユーザーインタフェース部分を変更するのが容易である。

4) MVC モデルで作られたプログラムの動作の流れ

* クライアントからのリクエストを Controller が受け付ける。
* Controller が該当する Model に処理を依頼する。もし必要があれば View にも表示の書き換えを指示する。
* Model が処理結果を View に渡す。
* View は出力(表示)用のデータを整える。
* クライアントにレスポンスを返す。