Java概説

応用プログラミングI

作田 誠

2022.9.22

内容

- Javaの歴史
- Javaの特徴
- C, C++, C# との比較
- Javaプログラムの利用状況

Javaの歴史 (1)

- 1991年 ジェームズ・ゴスリングらにより、家電等への組み込みを想定した言語として開発が開始される。当初はOak と呼ばれていたが、後に名前変更
- 1995年 Java言語正式発表。
  Netscape Navigator への搭載が発表され注目される
- 1996年 JDK (Java Development Kit) 1.0 の正式リリース
- 1997年 JDK 1.1 リリース
- 1998年 JDK 1.2 リリース。以降は **Java2** SDK (Software Development Kit) と呼ばれる
- 2000年 Java2 SDK 1.3 リリース。
  J2EE, J2SE, J2ME の3エディション

Javaの歴史 (2)

- 2001年 Java2 SDK 1.4 リリース
- 2004年 J2SE 5.0 (1.5) リリース。言語仕様が大きく拡張。ジェネリクス、オートボクシング、enum、拡張for、static import、アノテーション、...
- 2005年 Java EE, Java SE, Java ME に名称変更
- 2006年 Java SE 6 リリース。Scripting対応、高機能化、高速化
- 2007年 オープンソース版 OpenJDK リリース
- 2007年 モバイル向けプラットフォーム Android 発表
- 2008年 Android搭載マシン発売
- 2010年 開発元だったSun MicrosystemsがOracleに買収される

Javaの歴史 (3)

- 2011年 Java SE 7 リリース。switch文でStringで分岐、例外のマルチキャッチ、try-with-resources文、ダイアモンド演算子、数値リテラル _ で区切り、2進リテラルなど
- 2013年 Oracleがアプレット/JWSの証明書を必須に
  - 証明書は年間数万円程度なので、アマチュア的にはアプレット/JWSは死滅
- 2014年 Java SE 8 リリース
  - ラムダ式。メソッド参照。Stream APIにより関数型プログラミングにも対応
  - インターフェースのstaticメソッド・デフォルトメソッド
  - 新しい日付時刻API
- 2015年9月 Google Chromeでアプレット動作不可に
- 2017年9月 Java SE 9 リリース。モジュール化に対応

Javaの歴史 (4)

- 2018年3月 Java SE 10 リリース
  - ローカル変数の型推論、など
- 2018年9月 Java SE 11 リリース
  - ラムダ式での型推論、など
  - Javaアプレット・Java Web Start廃止、JavaFX非推奨
  - 11以降は Oracle JDK は有償サポートで商用利用可
無償利用は非商用用途のみ
  - フリー版として OpenJDK が使える
- 2019年 Oracle Java SE 12, 13 リリース。OpenJDKも12, 13 リリース
- 2020年 Java SE 14, 15 リリース。OpenJDKも
- 2021年 Java SE 16, 17 リリース。OpenJDKも
- 2022年 Java SE 18, 19 リリース。OpenJDKも

Javaの特徴

- C / C++ とほぼ同じ構文
- 仮想マシン (VM) コードによる実行
- 自動化されたガベージコレクション
- オブジェクト指向
- 豊富な標準 API
- ネットワーク対応のプログラム開発が容易

C / C++ とほぼ同じ構文

- if ～ (else ～), switch ～ case (default), while, do ～ while, for, continue, break など
  - C / C++ とほぼ同じ (一部拡張)
- gotoはない
- Cと似たデータ型。ただし厳密に規格化
  - int は32ビット符号付き整数 (2の補数表現)
  - long は64ビット符号付き、short は16ビット符号付き、byteは8ビット符号付き
  - float, double は　IEEE 754規格の浮動小数点
- ポインタがない

仮想マシン (VM) による実行

- コンパイル
  - ソース (*.java) → VMコード (*.class)
    ※ バイトコードとも言われる
- VMコードをJava仮想マシンが実行
  - プラットフォーム非依存を実現
  - 実行はインタプリタ的
    - VMコード読み込み・解釈・実行
  - 一部 JIT (Just In Time) コンパイラ技術によってネイティブコードに変換されてCPUが直接実行
    - 実行頻度の高い箇所 (hotspot)

自動化されたガベージコレクション

- ガベージコレクション (garbage collection, GC)
  - 使わなくなったメモリ領域を回収して、再利用できるようにすること
- C / C++ では自動でない
  - プログラムで明示的に解放する必要
  - メモリ解放されないポインタが残る問題
- 【長所】プログラムで気を使わなくてよい
- 【短所】実行時にプログラマが予測できない時点で GC が動き、メインの動作を遅くする危険性

オブジェクト指向

- C++ ではオブジェクト指向開発「も」できるが、従来通りの (Cと同じ) 開発も可能
  - 両者が混在して混乱を招きやすい
- Javaではコードはクラス(またはインタフェース)の中
  - すっきりしている
- オブジェクト指向開発が基本
- クラスの多重継承はできない
- インタフェースは多重継承・複数実装できる

豊富な標準 API

- **API (Application Programming Interface)**
  - プログラム中で使用するライブラリ
- C/C++の標準ライブラリはGUI開発をサポートしない
- Javaでは、GUIも含めほとんどの機能が標準APIに
  - グラフィクス、サウンド、タイマなども
  - ただしPC用JavaとAndroidでは、コア部分のみ互換で、GUI部分等は非互換

ネットワーク対応

- ネットワークAPIも標準化されている
  - Cに比べ利用も簡単
- ネットワーク対応のプログラム開発が容易
  - ただし、細かい制御はできない
- プログラムのネットワーク上での利用も容易
  - アプレット
  - Java Web Start (JWS)
  - サーバサイドJava

C, C++, C# との比較 (1)

- C
  - 現在開発現場で主流の C++ / Java の元
  - シンプルな仕様
  - OSやデバイスドライバ開発、組み込み系
- C++
  - Cをオブジェクト指向開発「も」できるように拡張
    - 構造体 (struct) を拡張し、クラス (class) を導入
  - Windowsアプリ開発の主流だった
    - Visual C++ と MFC (クラスライブラリ)
  - 巨大な言語仕様であり、さらに多機能化進行中 (多重継承、演算子のオーバーロード、テンプレート、右辺値参照、ラムダ関数、型推論、range-based for …)、ライブラリ (標準ライブラリ拡張中。STL)

C, C++, C# との比較 (2)

- Java
  - 肥大化したC++の反省
  - すっきりした仕様 (例) ポインタの排除、コードはクラス/インターフェースの中、多重継承なし
  - VM、ガベージコレクション、充実した標準API
- C#
  - MicrosoftがC++とJavaを参考に開発
  - .NET Framework 上のアプリケーション開発
  - Javaに似ている点が多い
    - よく似た構文
    - VMに相当するCLR（Common Language Runtime）環境による実行、ガベージコレクション、等
  - Microsoft環境以外に広く普及するかは不明

Javaプログラムの利用状況

- Java SE
  - アプレット (applet)
  - アプリケーション (application)
- Java ME
  - 携帯機器(携帯電話等)用アプリケーション
- Java EE
  - サーバサイド Java
  - サーブレット, JSP, Webアプリケーション
  - EJB (Enterprise JavaBeans)
- Android
  - スマートフォン, タブレット型マシンなど

アプレット (applet)

- Webブラウザ上で動作するプログラム
  - サーバから自動的にダウンロードされ実行
- Javaが最初に成功した分野
- セキュリティ上の制約 (ファイル保存、ネットワークアクセス)
- Microsoft による妨害
  - MS JVM バージョンを 1.1 に凍結
  - Windows XPマシンには VM がインストールされていなかった
- 将来性はない
  - Windows Vista / 7 以降のマシンでは、少なくともOSインストールしただけでは使えない
- 2013年Oracleが証明書を必須に。2015年Chromeでアプレット動作不可に ⇒ アプレットはほぼ死滅
- 2017年のJava 9 ではブラウザプラグインに対応せず。⇒ アプレットは完全に死滅

アプリケーション (application)

- OS上で独立して動作するプログラム
- 動作には実行環境 JRE が必要
- **Java Web Start** (JWS) によるインストール自動化
  - 2013年Oracleが証明書を必須に ⇒ アマチュア的にはJWSによるWeb公開は死滅
- OSの種類に依存せずに、プログラムを開発できる
  - **Write Once, Run Anywhere**
- リッチクライアントは将来性ない
  - Swing または SWT によるGUIプログラム
- ローカルPCでの実行は有効

旧携帯用アプリケーション

- 旧形式の携帯電話上で動作するプログラム
  - プログラムサイズ、使用メモリ量などに制限
- iアプリ (NTTドコモ)
- MIDP (au オープンアプリ、ソフトバンク S!アプリなど)
- 機能を縮小したJava (Java ME) 環境
- ゲーム、占いなど多くのサイト
- 端末台数が圧倒的に多かった
- JavaScript や Flash が動くフルブラウザの普及で利用されなくなった
- スマートフォンの普及で従来型(いわゆるガラパゴス携帯)は縮小し、消え去る運命

Androidアプリケーション

- Android OSの搭載されたモバイル機器で動作
- 2011年にはスマートフォン分野のトップシェア
- Androidプラットフォーム
  - Googleが中心となった Open Handset Alliance (OHA) によって開発されているモバイル機器用の共通ソフトウエア
  - Javaプログラムは、従来のJava VM上ではなく、モバイル向けに設計されたDalvik仮想マシン上で動作
  - Android 5.0からは ART (Android Runtime) 仮想マシン上で動作。アプリケーションの動作効率などが向上
  - コア部分は Java SE との互換性が高い
    - Java 1.6 (JDK 6) 互換
    - APIレベル 19 (Android 4.4.2) で Java 7 サポート
    - APIレベル 24 (Android 7.0) で Java 8 サポート
  - コア以外(GUI等)の部分はパソコン用Javaとは全く別個

サーバサイド Java

- またたく間に主流の一つとなった
- サーブレット (servlet)
  - Webサーバ上で動き、HTTPの応答を受け持つJavaプログラム
- JSP (JavaServer Pages)
  - HTML中に、コード(スクリプトレット)やXML風のタグ構文で処理を記述
  - サーブレットに変換・コンパイルされて実行される
- Webアプリケーション
  - HTMLや上記を組み合せ、Webサイト上で動く
  - オンラインショッピング、掲示板など

今後の見通し

- サーバサイドJavaは今後も利用されていく
- Javaアプレットは死滅
- Javaアプリケーションは、JWSによるWeb公開は死滅。ローカルPCでの実行は有効
- Androidアプリは今後も非常に有望。大きなシェアがある
- いずれにせよJavaが「ちゃんと」書ければキャリアアップ可能