Oracle Direct Seminar

ORACLE

超入門！はじめてみようJavaプログラミング

日本オラクル株式会社

Oracle Direct

Oracle Direct Seminar

以下の事項は、弊社の一般的な製品の方向性に関する概要を説明するものです。また、情報提供を唯一の目的とするものであり、いかなる契約にも組み込むことはできません。以下の事項は、マテリアルやコード、機能を提供することをコミットメント（確約）するものではないため、購買決定を行う際の判断材料になさらないで下さい。オラクル製品に関して記載されている機能の開発、リリースおよび時期については、弊社の裁量により決定されます。

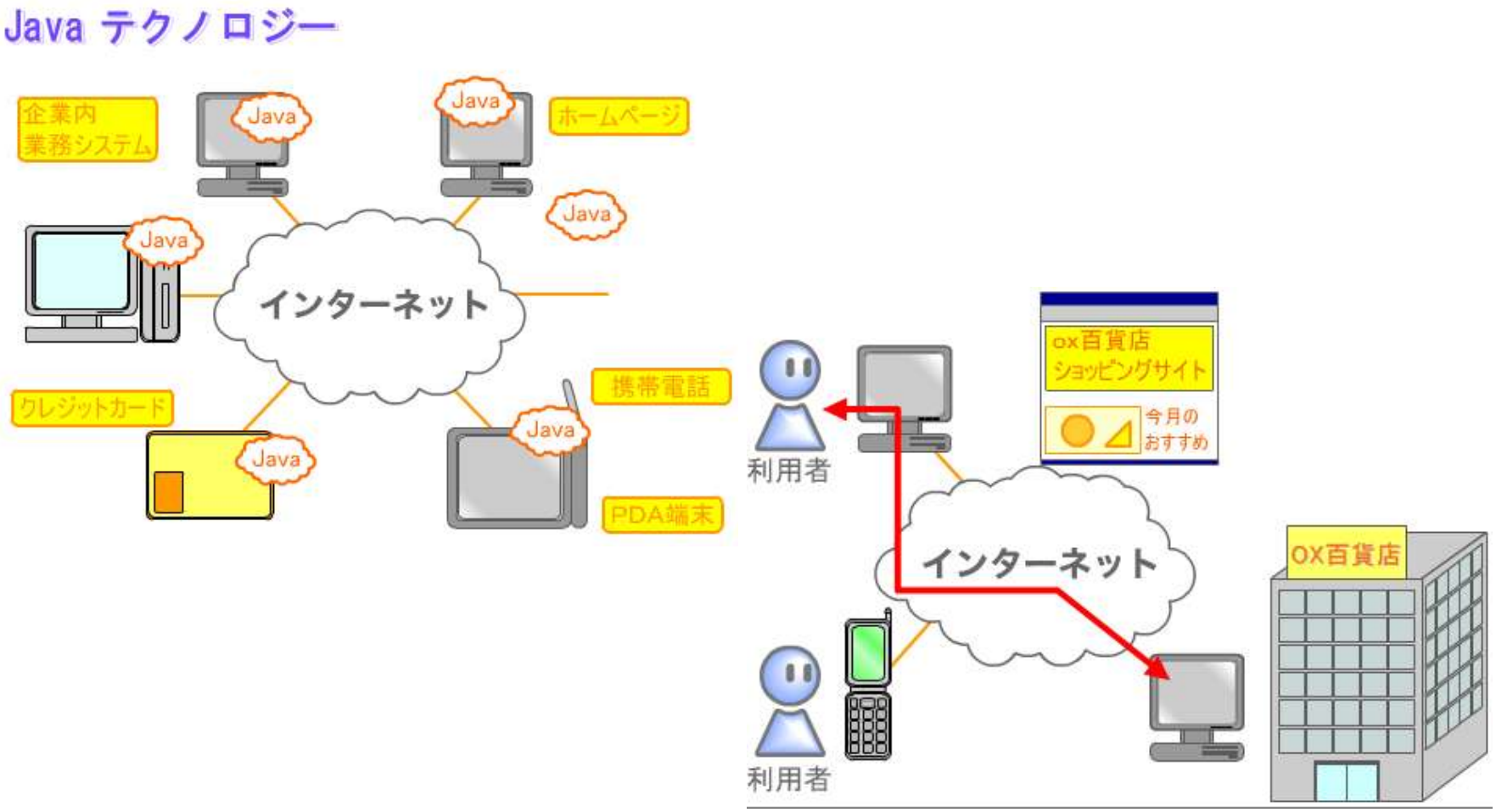
Agenda

- Java って何？
- 基本的なJavaプログラムの作り方
- お勧め研修コース

Agenda

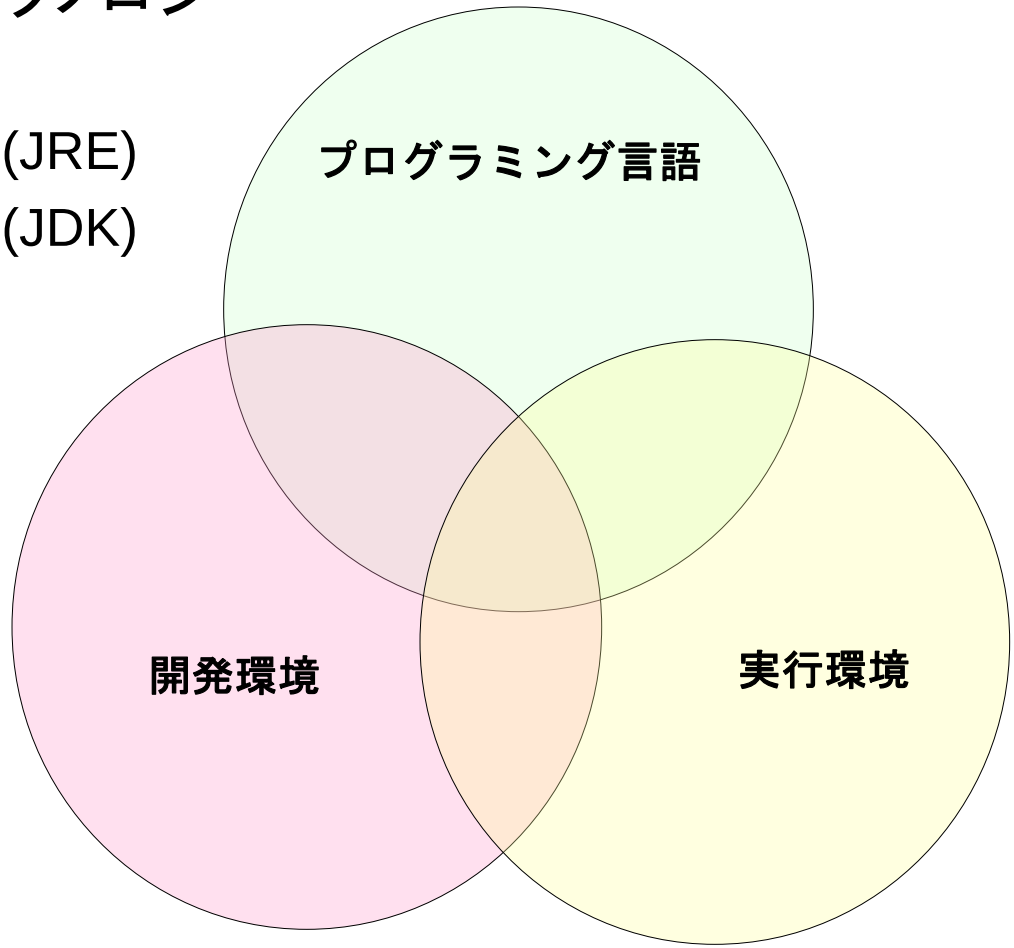
- Java って何？
  - Javaって何？
  - Javaテクノロジー
  - プログラミング言語としてのJava
  - 実行環境としてのJava
  - 開発環境としてのJava
  - Javaの構成
  - Javaのエディション
  - プログラムの作成手順
- 基本的なJavaプログラムの作り方
- お勧め研修コース

Javaって何？



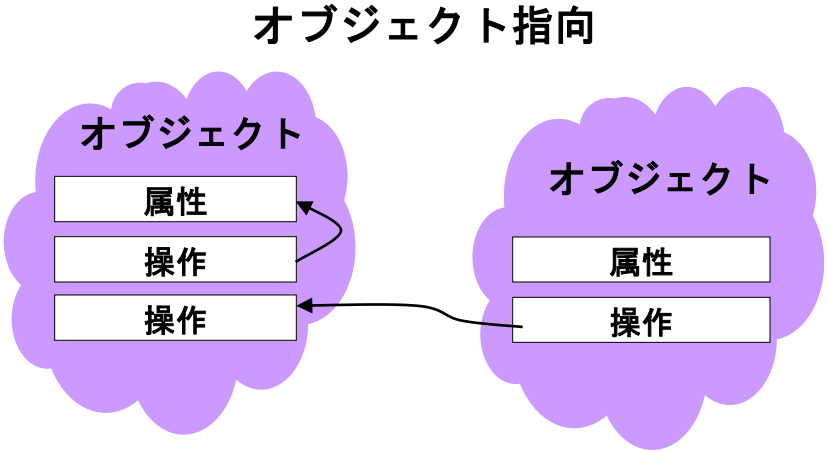
Javaテクノロジー

- 3つの側面を持つテクノロジー
  - プログラミング言語
  - プログラム実行環境(JRE)
  - プログラム開発環境(JDK)

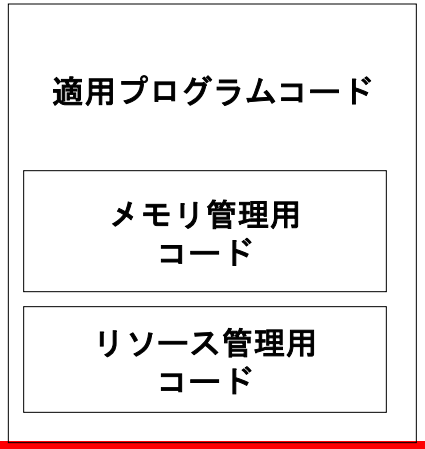


プログラミング言語としてのJava

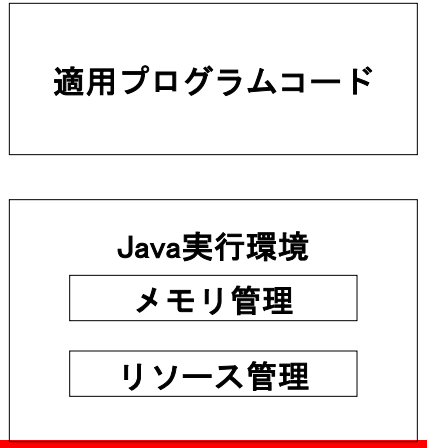
- シンプル
- オブジェクト指向
- 豊富なAPI
- セキュリティ



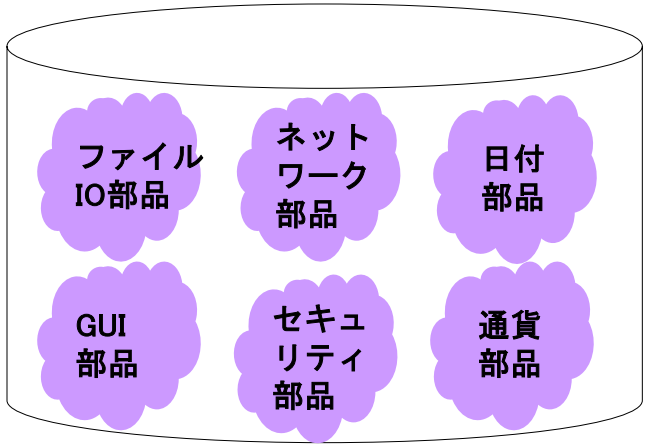
従来のプログラミング言語によるプログラム



Java言語によるプログラム

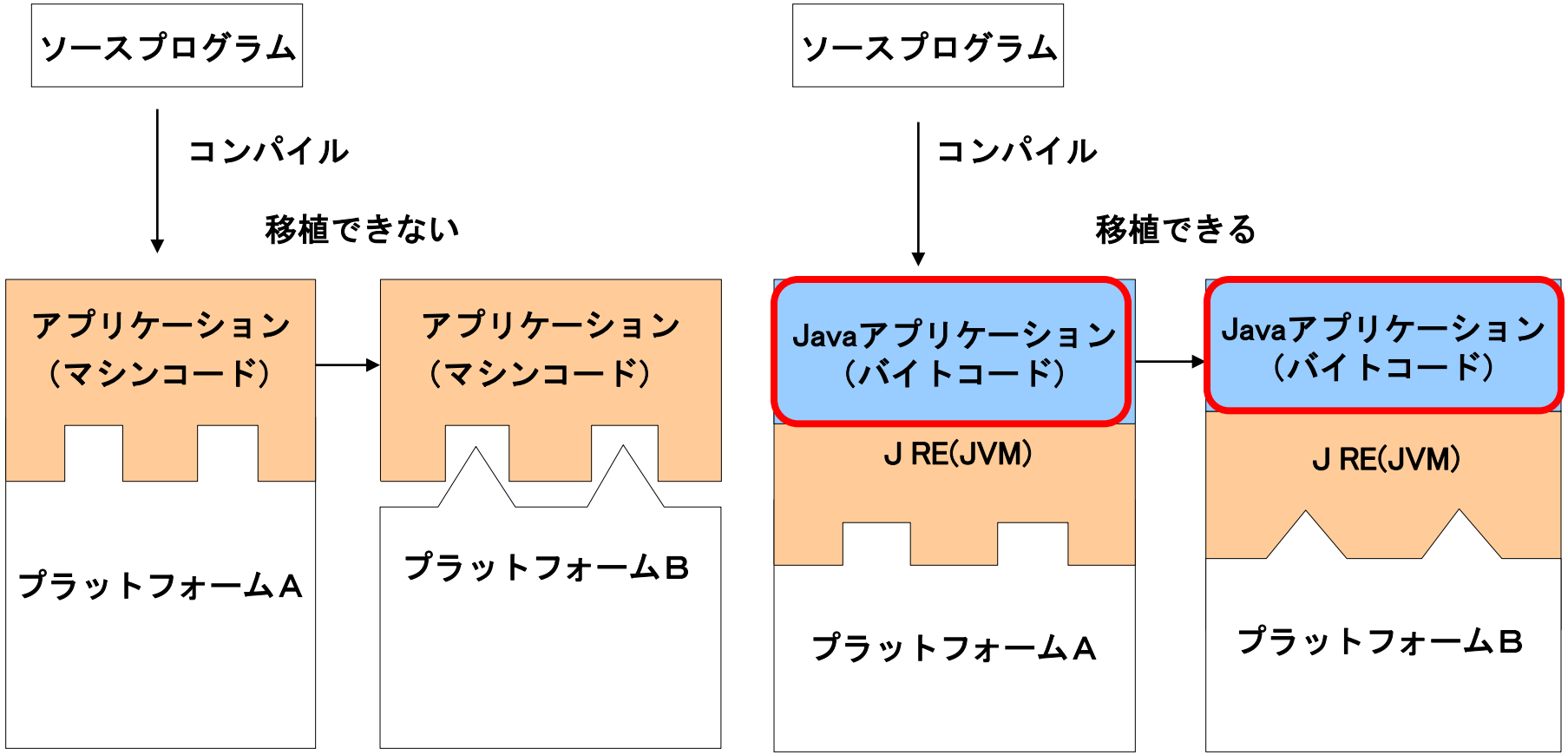


豊富なAPI、ライブラリ群



実行環境としてのJava

- プラットフォーム非依存(Write Once, Run Anywhere)



開発環境としてのJava

Javaソースコード

```
import java.util.*;
public class Vehicle {
    String  id ;
    int price ;
    int sales [ ] ;

    totalSales ( ) {
        ・・・

}
```

コンパイル

Java
アプリケーション

ドキュメントジェネレータ
（ javadoc.exe ）

コンパイラ
（ javac.exe ）

インタプリタ
（ java.exe ）

デバッガ
（ jdb.exe ）

Java開発環境

実行

デバッグ実行

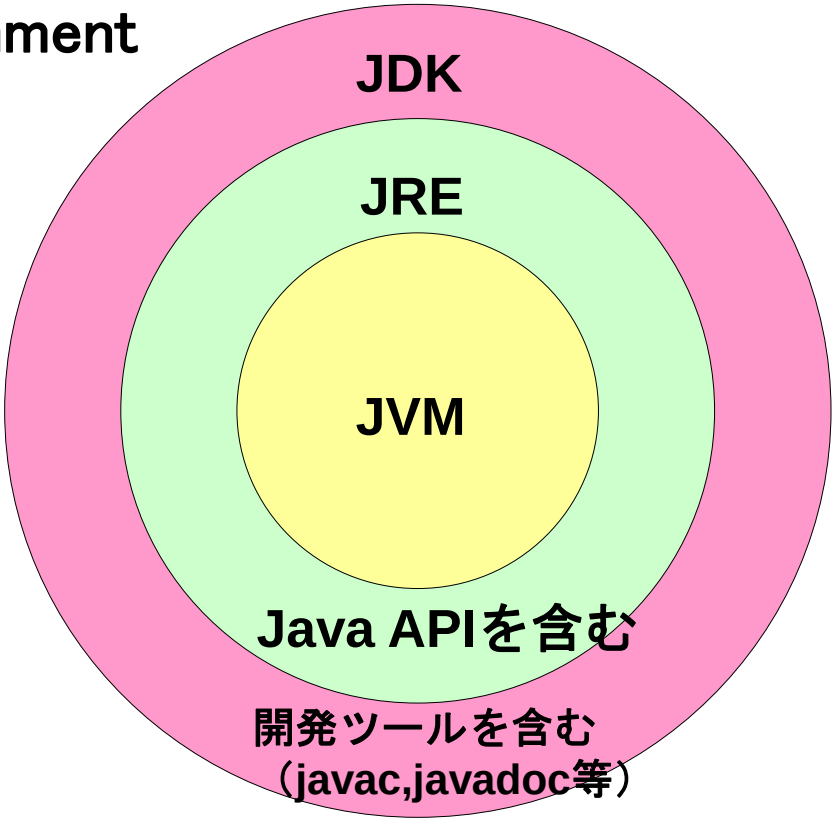
API ドキュメント
（ HTML 形式）

JRE(JVM)

プラットフォーム

Javaの構成

- JDK : Java SE Development Kit
- JRE : Java Runtime Environment
- JVM : Java Virtual Machine



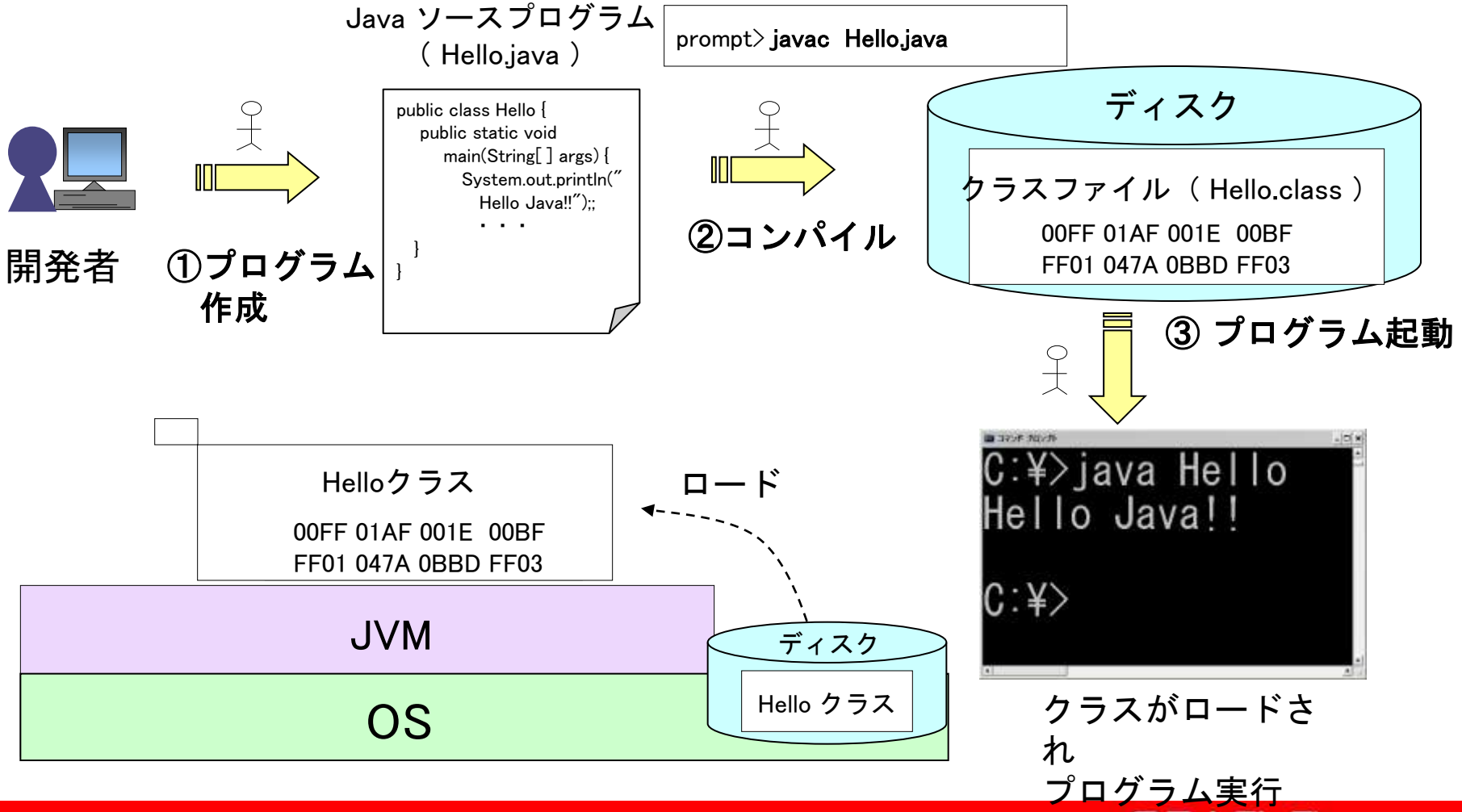
Javaのエディション

Java SE
(Java Platform, Standard Edition)

Java EE
(Java Platform, Enterprise Edition)

Java ME
(Java Platform, Micro Edition)

プログラムの作成手順



簡単なプログラムの例

Hello.java

```
1. class Hello {
2.   public static void main(String[] args) {
3.     System.out.println("Hello Java!!");
4.   }
5. }
```

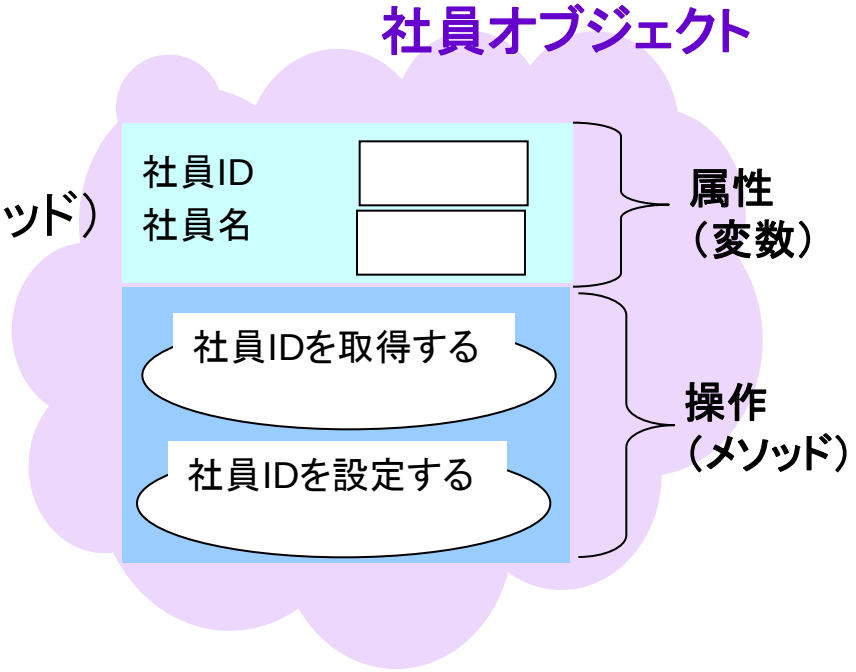
```
> javac Hello.java

> dir
2011/02/24  11:02     416  Hello.class
2011/02/24  11:00     107  Hello.java

> java Hello
Hello Java!!
```

Agenda

- Java って何？
- 基本的なJavaプログラムの作り方
  - オブジェクト
  - クラス
  - Javaプログラムの構成
  - クラス定義
  - オブジェクト生成とアクセス
  - メソッドのオーバーロード
  - オブジェクトの初期化
  - コンストラクタ
  - カプセル化とデータ隠蔽
  - アクセス修飾子
- お勧め研修コース

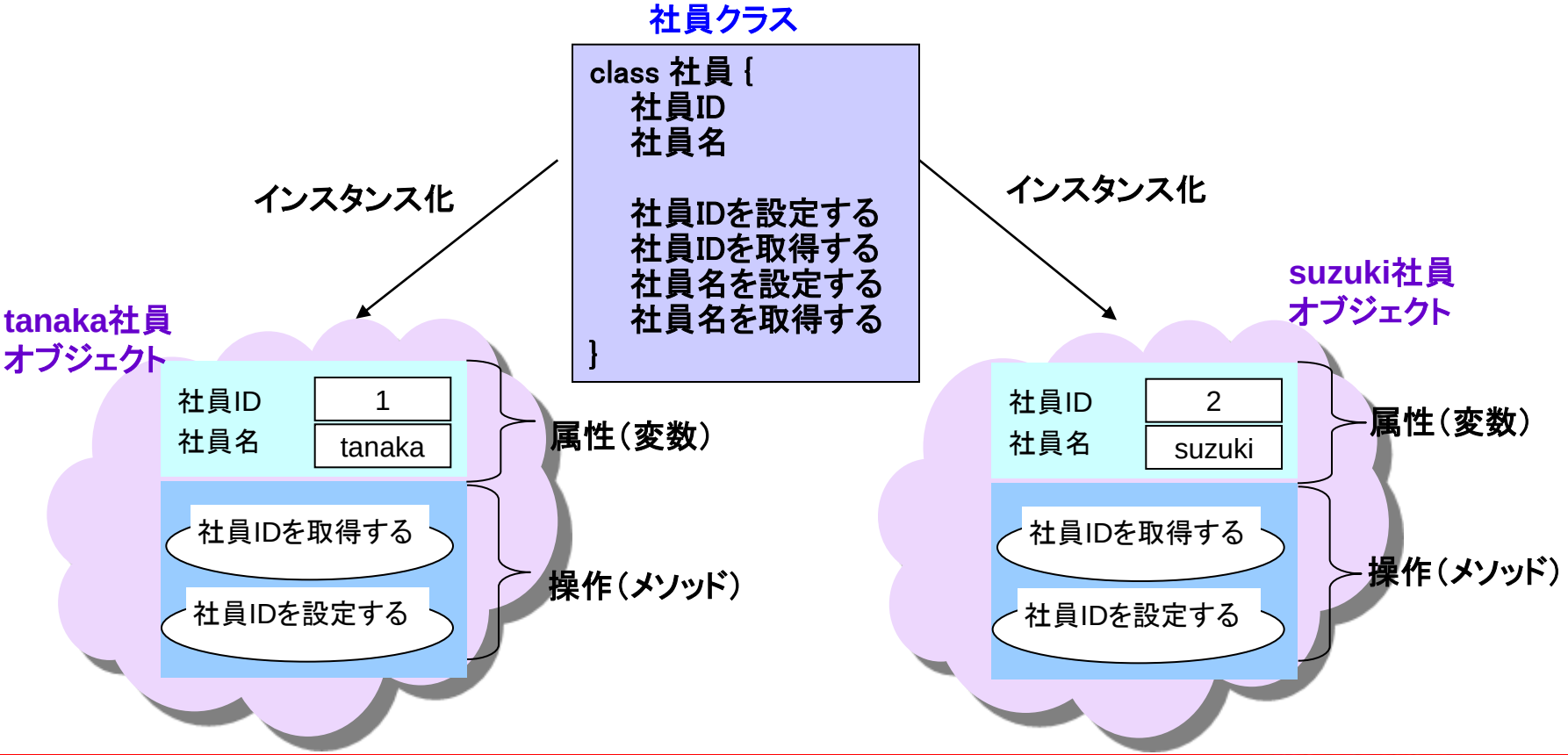
オブジェクト

- オブジェクト＝Object＝「もの」
- オブジェクトの種類
  - 実在するもの　：　車、自転車、ノート、パソコン etc.
  - 概念的なもの　：　社員、銀行口座 etc.
- オブジェクトが持つもの
  - 属性：特性、状態を表す（変数）
  - 操作：動作、ふるまいを表す（メソッド）



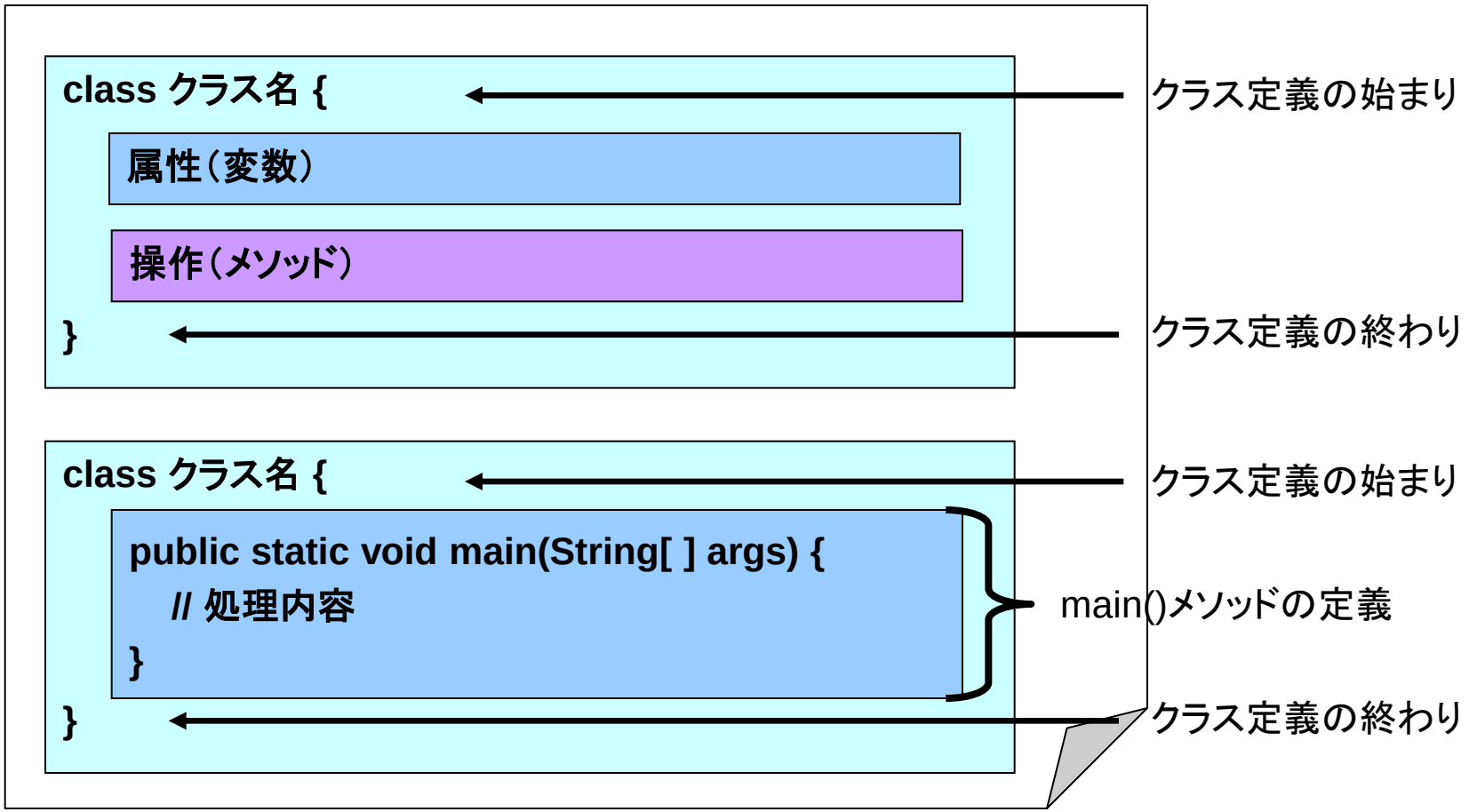
クラス

- さまざまなデータを1つにまとめて扱うための型
- オブジェクトが持つ属性や操作を定義



Javaプログラムの構成

- Javaプログラムはクラスの集合で構成



クラス定義

- クラスという単位でプログラムを作成
- クラスとは独自のデータ型のようなもの

```
class クラス名 {

    // インスタンス変数の定義
    [修飾子]  データ型  インスタンス変数名;

    // メソッドの定義
    [修飾子]  戻り値の型  メソッド名(引数リスト) {
            // 処理内容
    }

}
```

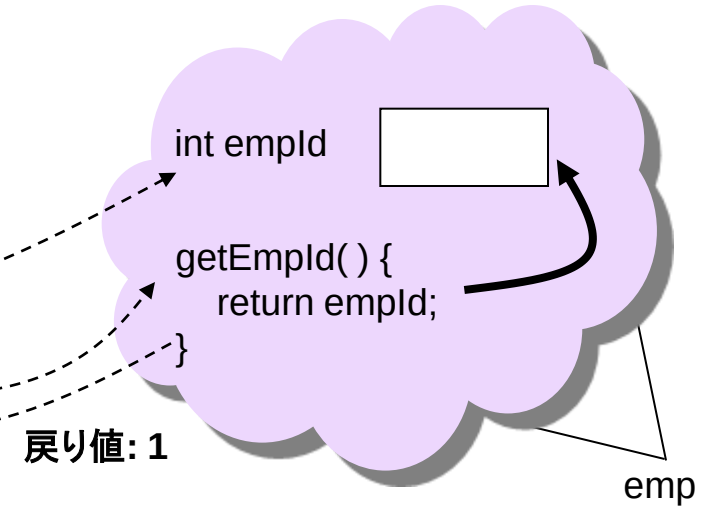
例:

```
class Employee {
  // インスタンス変数の定義
  int  empId;

  // メソッドの定義
  int getEmpId() {
    return empId;
  }

  void setEmpId(int id) {
    empId = id;
  }
}
```

オブジェクト生成とアクセス

1. オブジェクトを生成し、参照するための変数に代入
   クラス名　参照変数名　=　new　クラス名( );
2. .（ドット）演算子を使用してアクセス
   参照変数名.インスタンス変数名
   参照変数名.メソッド名( )

```
class Employee {
   // インスタンス変数の定義
   int empId;
   //メソッド定義
   int getEmpId() { return empId; }
}
class UseEmployee {
  public static void main(String[ ] args) {
   // オブジェクトの生成
   Employee emp = new Employee();
   //変数へアクセス(参照変数名.変数名)
   emp.empId = 1;
   //メソッドへアクセス(参照変数名.メソッド名)
   int id = emp.getEmpId();
  }
}
```



サンプル

```
 1. class Employee {
 2.   // インスタンス変数の定義
 3.   int empId;        // 社員ID
 4.   String empName;  // 社員名
 5.
 6.   // メソッドの定義
 7.   void setData(int id, String name) { //インスタンス変数に値を設定するメソッド
 8.     empId = id;
 9.     empName = name;
10.   }
11.   int getEmpId() {       // 社員IDを取得するメソッド
12.     return empId;
13.   }
14.   String getEmpName(){  // 社員名を取得するメソッド
15.     return empName;
16.   }
17. }
```

サンプル

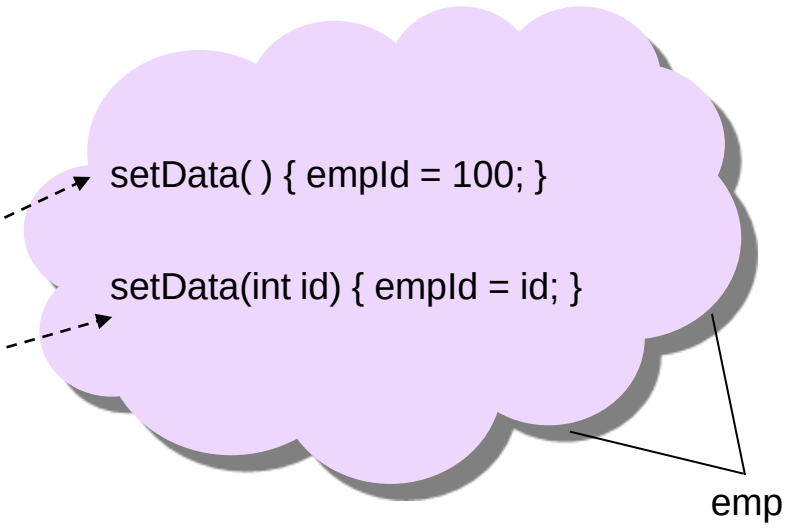
```
18. class CreateSample {
19.   public static void main(String[] args) {
20.     Employee emp1 = new Employee();  // Employeeオブジェクトの生成
21.     emp1.setData(1, "Tanaka");       // setData()メソッドの呼び出し
22.     System.out.println("Employee ID : " + emp1.getEmpId());
23.     System.out.println("Employee Name : " + emp1.getEmpName());
24.
25.     Employee emp2 = new Employee();  // Employeeオブジェクトの生成
26.     emp2.setData(2, "Suzuki");       // setData()メソッドの呼び出し
27.     System.out.println("Employee ID : " + emp2.getEmpId());
28.     System.out.println("Employee Name : " + emp2.getEmpName());
29.   }
30. }
```

```
> java CreateSample
Employee ID : 1
Employee Name : Tanaka
Employee ID : 2
Employee Name : Suzuki
```

メソッドのオーバーロード

- 同一クラス内に同名のメソッドを定義すること
- オーバーロードのルール
  - 引数の数、型が異なっていること

```
class Employee {
  int empId;
  String empName;
   //引数を取らないsetData()メソッド
     void setData() { empId = 100; }
    // 引数を1つ取るsetData()メソッド
    void setData(int id) { empId = id; }
              :
}

class UseEmployee {
              :
  // オブジェクトの生成
  Employee emp = new Employee();
  // メソッド呼び出し
  emp.setData( );
  emp.setData(1);
}
```



サンプル

```
 1. class Employee {
 2.   // インスタンス変数の宣言
 3.   int empId;
 4.   String empName;
 5.   // 引数を1つ取るsetData()メソッド
 6.   void setData(int id) {
 7.     empId = id;
 8.     empName = "unknown";
 9.   }
10.   // 引数を2つ取るsetData()メソッド
11.   void setData(int id, String name) {
12.     empId = id;
13.     empName = name;
14.   }
15.   void display() {
16.     System.out.println("Employee ID : " + empId);
17.     System.out.println("Employee Name : " + empName);
18.   }
19. }
```

サンプル

```
20. class OverloadSample {
21.   public static void main(String[] args) {
22.     Employee emp1 = new Employee();
23.     emp1.setData(1);            // 引数を1つ取るsetData()メソッドの呼び出し
24.     emp1.display();
25.
26.     Employee emp2 = new Employee();
27.     emp2.setData(2, "Suzuki"); // 引数を2つ取るsetData()メソッドの呼び出し
28.     emp2.display();
29.   }
30. }
```

```
> java OverloadSample
Employee ID : 1
Employee Name : unknown
Employee ID : 2
Employee Name : Suzuki
```

オブジェクトの初期化

- オブジェクト生成時に自動的に初期化
- デフォルトの初期値

データ型	初期値
byte	0
short	0
int	0
long	0
float	0.0f
double	0.0d
char	'¥u0000'
boolean	false
参照型	null

コンストラクタ

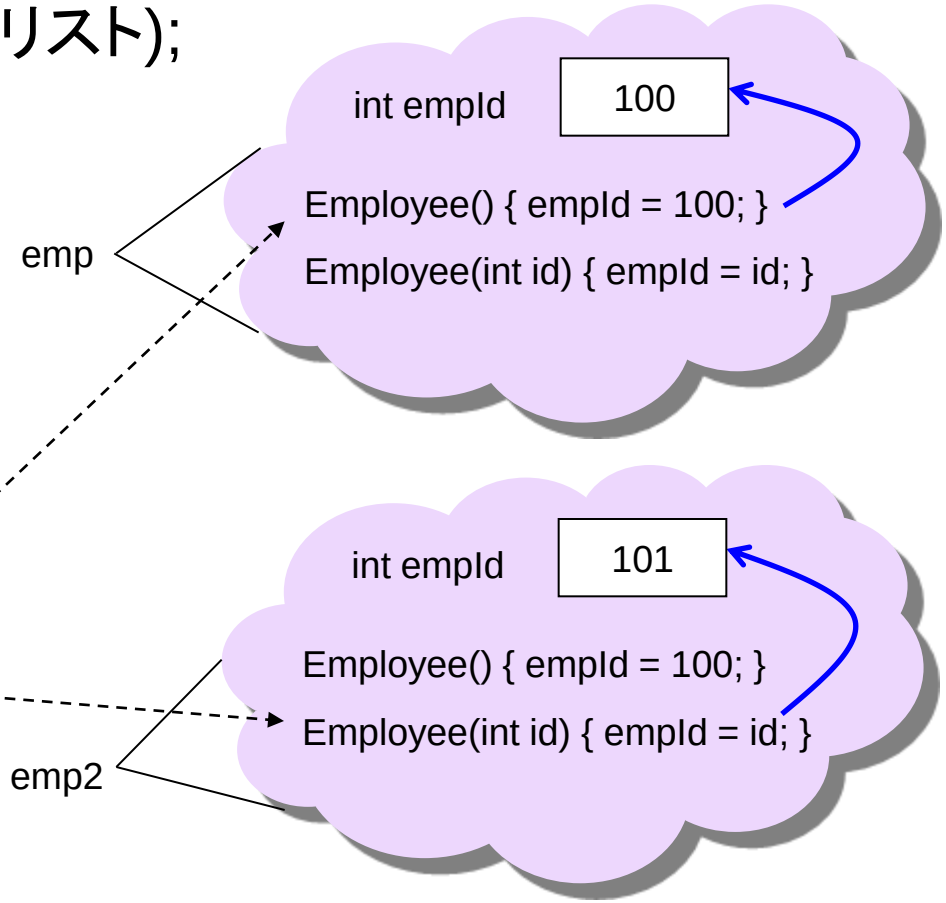
- オブジェクトを初期化するための処理ブロック
- オブジェクト生成時に一度だけ呼び出される
- 定義ルール
  - クラス名と同じ名前
  - 戻り値を持たない（戻り値の型宣言もなし）
  - 引数を持つことが可能
  - オーバーロード可能

```
Employee() { empId = 100; }    // 引数を取らないコンストラクタ
Employee(int id) { empId = id; }  //引数を1つ取るコンストラクタ
```

コンストラクタの呼び出し

- オブジェクト生成時に呼び出される
  new クラス名(引数リスト);

```
class Employee {
  int empId;
   //コンストラクタ
     Employee() { empId = 100; }
    Employee(int id) { empId = id; }

              ;
    }
}

class UseEmployee {
              :
  // コンストラクタ呼び出し
  Employee emp = new Employee();
  Employee emp2 = new Employee(101);
              :
}
```



デフォルトコンストラクタ

- コンパイラによって自動生成されるコンストラクタ
  - 引数なし
  - 本体は空

サンプル

```
 1. class Employee {
 2.   // インスタンス変数の宣言
 3.   int empId;
 4.   String empName;
 5.   // 引数を1つ取るコンストラクタ
 6.   Employee(int id) {
 7.     empId = id;
 8.     empName = "unknown";
 9.   }
10.   // 引数を2つ取るコンストラクタ
11.   Employee(int id, String name) {
12.     empId = id;
13.     empName = name;
14.   }
15.   void display() {
16.     System.out.println("Employee ID : " + empId);
17.     System.out.println("Employee Name : " + empName);
18.   }
19. }
```

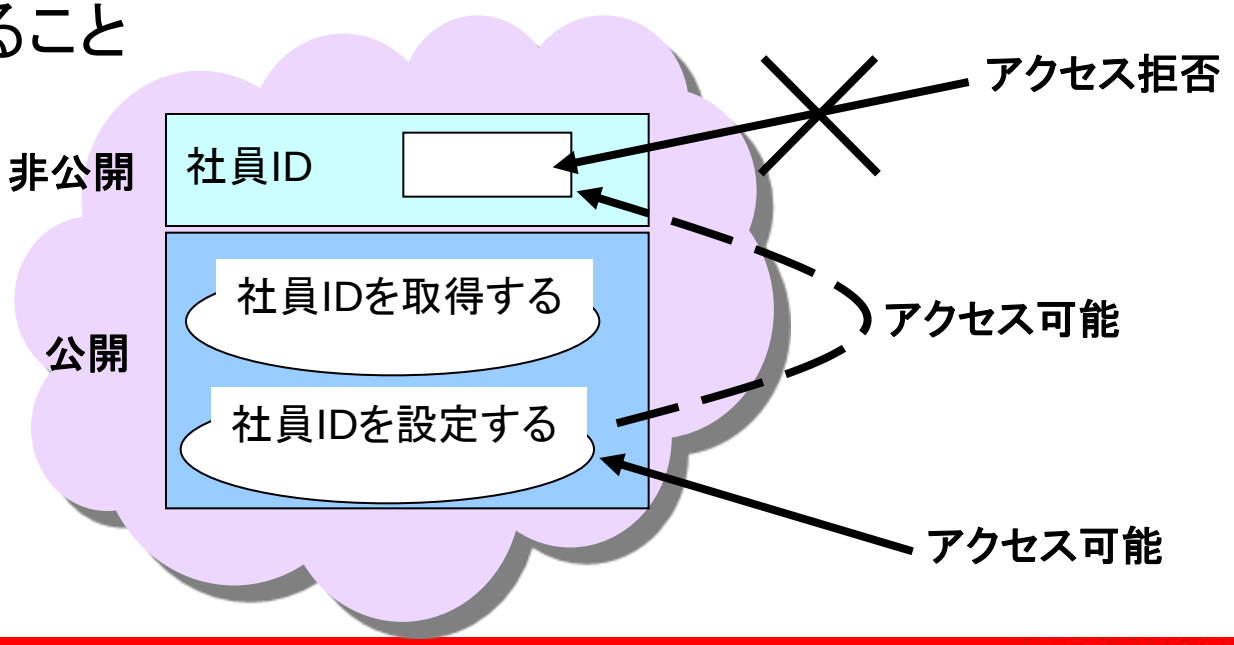
サンプル

```
20. class ConstSample {
21.   public static void main(String[] args) {
22.     // 引数を1つ取るコンストラクタの呼び出し
23.     Employee emp1 = new Employee(1);
24.     emp1.display();
25.
26.     // 引数を2つ取るコンストラクタの呼び出し
27.     Employee emp2 = new Employee(2, "Suzuki");
28.     emp2.display();
29.   }
30. }
```

```
> java ConstSample
Employee ID : 1
Employee Name : unknown
Employee ID : 2
Employee Name : Suzuki
```

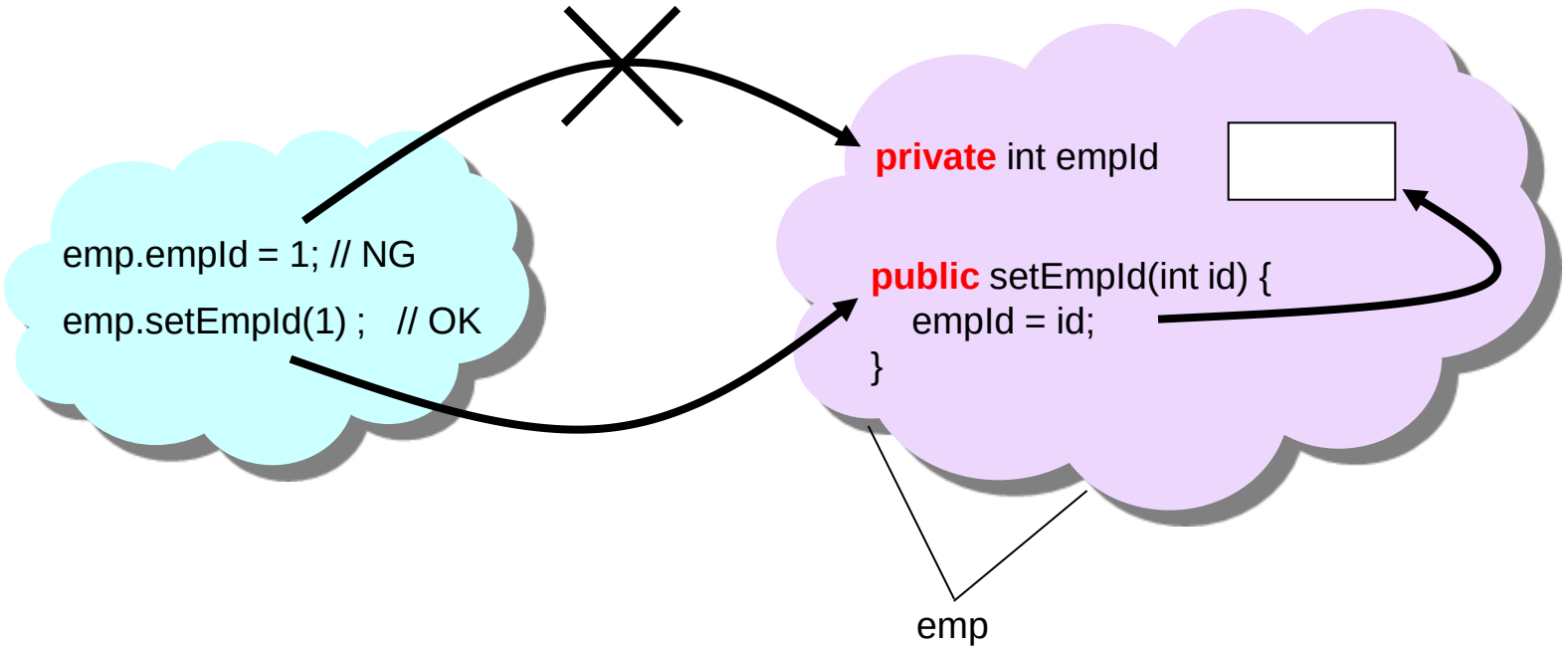
カプセル化とデータ隠蔽

- カプセル化とは
  オブジェクト内に属性（変数）とそれに対する操作（メソッド）をひとつにまとめて持たせること
- データ隠蔽
  あるオブジェクト内の変数やメソッドに対するアクセスを制限すること



アクセス修飾子

- データ隠蔽を実現するために使用する修飾子
  - public : どこからでもアクセス可
  - private :変数およびメソッドが定義されたクラス内のメソッドからのみアクセス可能



サンプル

```
 1. class Employee {
 2.   public int empId;         // public 変数
 3.   private String empName;  // private 変数
 4.
 5.   public Employee(int id, String name) {
 6.     empId = id;
 7.     empName = name;
 8.   }
 9.
10.   public int getEmpId() {
11.     return empId;
12.   }
13.
14.   public String getEmpName() {
15.     return empName;
16.   }
17.
18. }
```

サンプル

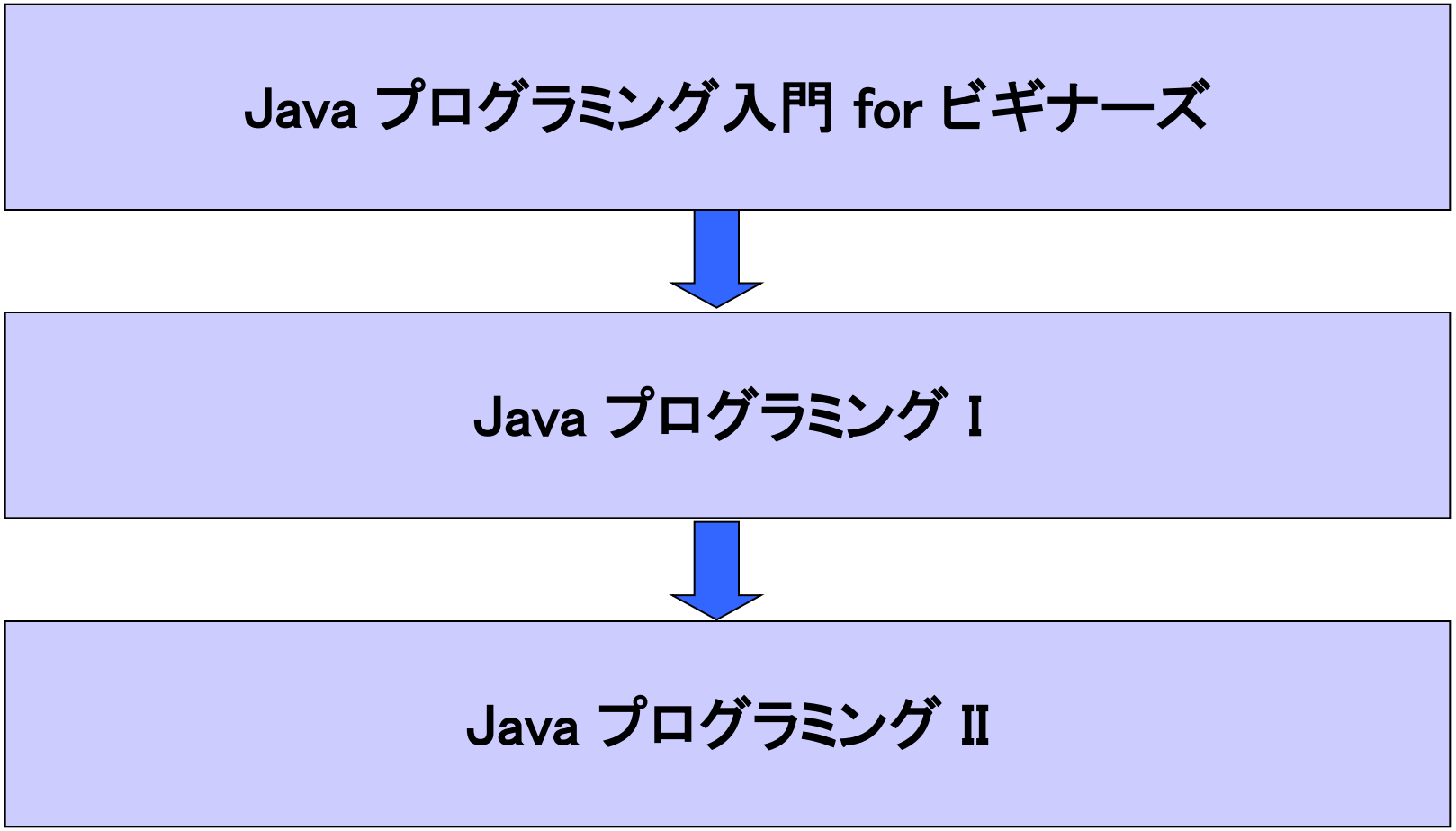
```
19. class AccessSample {
20.   public static void main(String[] args) {
21.     Employee emp1 = new Employee(1, "Tanaka");
22.
23.     // インスタンス変数の適切な取得方法
24.     System.out.println("Employee ID : " + emp1.getEmpId());
25.     System.out.println("Employee Name : " + emp1.getEmpName());
26.
27.     // エラーにはならないが、不適切な取得方法
28.     System.out.println("Employee ID : " + emp1.empId);
29.
30.     // private変数にアクセスしているため、コンパイルエラー
31.     // System.out.println("Employee Name : " + emp1.empName);
32.   }
33. }
```

```
> java AccessSample
Employee ID : 1
Employee Name : Tanaka
Employee ID : 1
```

Agenda

- Java って何？
- 基本的なJavaプログラムの作り方
- お勧め研修コース

お勧め研修コース



お勧め研修コース

Java プログラミング入門 for ビギナーズ
（2日間）

- プログラミング言語未経験者向けコース
- プログラミング言語の基礎を学ぶ

このコースでは、プログラミング経験のない方のために、プログラムの作成から実行の流れ、コンパイルなど、Java プログラミングの基礎を学習します。また、Java プログラム内で扱う変数やリテラルなど、データの種類と使用方法、およびデータの演算を行うための各種演算子について学習します。さらに、プログラムの実行順序を制御する制御文や、一連の処理をひとまとめにするメソッドの機能と使用方法など、プログラミングの基礎スキルを実習を通して学習します。

お勧め研修コース

Java プログラミング I
（3日間）

- 他言語経験者向けコース
- Javaプログラミングの基礎を学ぶ

このコースは、Java の概要 および Java 言語の基本文法について習得します。クラスとオブジェクト、カプセル化、継承、ポリモフィズムなど、オブジェクト指向プログラミングの基本知識およびテクニックについて実習を通して学習します。また、基本的なライブラリの使用方法や例外処理についても学習します。

お勧め研修コース

Java プログラミング II
（2日間）

- Java SE の基本テクノロジーを学ぶ

このコースでは、java.util パッケージに含まれるコレクション・フレームワークおよびジェネリックスの使用方法について習得します。また、Java で並列処理を実現するスレッドの利用方法、Java プログラムにおけるファイル入出力について学習します。さらにソケットによるネットワークプログラミングについて学習します。

お勧め研修コース

- Java研修コース詳細情報
  - 集合研修
    http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getpage?page_id=402&p_nl=SUNL
  - オンライントレーニング
    http://education.oracle.com/pls/web_prod-plq-dad/db_pages.getCourseDesc?dc=D67614JP10&p_org_id=70&lang=JA

OTN×ダイセミ　でスキルアップ!!

・一般的な技術問題解決方法などを知りたい！
・セミナ資料など技術コンテンツがほしい！

Oracle Technology Network(OTN)を御活用下さい。

http://forums.oracle.com/forums/main.jspa?categoryID=484

一般的技術問題解決にはOTN掲示版の
「Java」をご活用ください

※OTN掲示版は、基本的にOracleユーザー有志からの回答となるため100%回答があるとは限りません。
ただ、過去の履歴を見ると、質問の大多数に関してなんらかの回答が書き込まれております。

http://www.oracle.com/technetwork/jp/testcontent/index-086873-ja.html

過去のセミナ資料、動画コンテンツはOTNの
「OTNセミナー　オンデマンド コンテンツ」へ

※ダイセミ事務局にダイセミ資料を請求頂いても、お受けできない可能性がございますので予めご了承ください。
ダイセミ資料はOTNコンテンツ　オン　デマンドか、セミナ実施時間内にダウンロード頂くようお願い致します。

ORACLE

OTNセミナー　オンデマンド　コンテンツ

ダイセミで実施された技術コンテンツを動画で配信中!!

ダイセミのライブ感はそのままに、お好きな時間で受講頂けます。

twitter

最新情報つぶやき中

oracletechnetjp

- 人気コンテンツは？
- お勧め情報
- 公開予告　など

OTN　オンデマンド　検索

※掲載のコンテンツ内容は予告なく変更になる可能性があります。
期間限定での配信コンテンツも含まれております。お早めにダウンロード頂くことをお勧めいたします。

Oracle エンジニアのための技術情報サイト

オラクルエンジニア通信

http://blogs.oracle.com/oracle4engineer/

twitter

最新情報つぶやき中

oracletechnetjp

- 技術資料
  - ダイセミの過去資料や製品ホワイトペーパー、スキルアップ資料などを多様な方法で検索できます
  - キーワード検索、レベル別、カテゴリ別、製品・機能別
- コラム
  - オラクル製品に関する技術コラムを毎週お届けします
  - 決してニッチではなく、誰もが明日から使える技術の「あ、そうだったんだ！」をお届けします

こんな資料が人気です

- ✓ 6か月ぶりに資料ダウンロードランキングの首位が交代！ 新王者はOracle Database構築資料でした。
- ✓ データベースの性能管理手法について、Statspack派もEnterprise Manager派も目からウロコの技術特集公開中

オラクルエンジニア通信

ITプロジェクト全般に渡る無償支援サービス

Oracle Direct Conciergeサービス

■パフォーマンス診断サービス
- Webシステム　ボトルネック診断サービス NEW
- データベースパフォーマンス 診断サービス

■移行支援サービス
- SQL Serverからの移行支援サービス
- DB2からの移行支援サービス
- Sybaseからの移行支援サービス
- MySQLからの移行支援サービス
- Postgre SQLからの移行支援サービス
- Accessからの移行支援サービス
- Oracle Application ServerからWeblogicへ移行支援サービス NEW

■システム構成診断サービス
- Oracle Database構成相談サービス
- サーバー統合支援サービス
- 仮想化アセスメントサービス
- メインフレーム資産活用相談サービス
- BI EEアセスメントサービス
- 簡易業務診断サービス

■バージョンアップ支援サービス
- Oracle Databaseバージョンアップ支援サービス
- Weblogic Serverバージョンアップ支援サービス NEW
- Oracle Developer/2000(Froms/Reports) Webアップグレード相談サービス

オラクル社のエンジニアが 直接ご支援します
お気軽にご活用ください!

オラクル　無償支援　検索

1日5組限定！

製品無償評価サービス

提供シナリオー例

・データベースチューニング　・無停止アップグレード

・アプリケーション性能・負荷検証　・Webシステム障害解析

インストールすることなく、すぐに体験いただけます

- サービスご提供までの流れ
  1. お問合せフォームより「製品評価サービス希望」と必要事項を明記し送信下さい
  2. 弊社より接続方法手順書およびハンズオン手順書を送付致します
  3. 当日は、弊社サーバー環境でインターネット越しに製品を体感頂けます

※サービスご提供には事前予約が必要です

Web問い合わせフォーム

「ダイデモ」をキーワードに検索することで申し込みホームページにアクセスできます

http://www.oracle.com/jp/direct/services/didemo-195748-ja.html

ORACLE

あなたにいちばん近いオラクル

Oracle Direct

まずはお問合せください

Oracle Direct　検索

システムの検討・構築から運用まで、ITプロジェクト全般の相談窓口としてご支援いたします。

システム構成やライセンス/購入方法などお気軽にお問い合わせ下さい。

Web問い合わせフォーム

専用お問い合わせフォームにてご相談内容を承ります。

http://www.oracle.com/jp/direct/inquiry-form-182185-ja.html

※こちらから詳細確認のお電話を差し上げる場合がありますので、ご登録されている連絡先が最新のものになっているか、ご確認下さい。

フリーダイヤル

0120－155－096

※月曜~金曜　9:00~12:00、13:00~18:00

（祝日および年末年始除く）

ORACLE