16. オーバーライド（overriding）の理解

例）

class Animal {

public void move() {

System.out.println("動物が移動します。");

}

}

class Dog extends Animal {

public void move() {

System.out.println("犬が走ります。");

}

}

public class Main {

public static void main(String[] args) {

Animal animal = new Animal(); // Animalのインスタンスを生成

animal.move(); // "動物が移動します。"と出力される

Dog dog = new Dog(); // Dogのインスタンスを生成

dog.move(); // "犬が走ります。"と出力される

}

}

Animalクラスにはmoveメソッドがあります。DogクラスはAnimalクラスを継承しており、moveメソッドをオーバーライドしています。そして、MainクラスでAnimalクラスとDogクラスのインスタンスを生成して、それぞれmoveメソッドを呼び出しています。Animalクラスのインスタンスでは、オーバーライドされたDogクラスのmoveメソッドではなく、Animalクラスのmoveメソッドが呼び出されます。Dogクラスのインスタンスでは、オーバーライドされたDogクラスのmoveメソッドが呼び出されます。

Javaでのメソッドのオーバーライドは、多態性（polymorphism）を実現するための重要な機能の一つです。

・オーバーライドをいつ使うのか？

多態性は、同じクラスを継承している複数のオブジェクトが、同じメソッドを異なる方法で実行することができることを意味します。これは、コードの再利用性を高め、柔軟性を持たせることができます。

オーバーライドは、親クラスのメソッドを子クラスで再定義し、子クラスで定義されたメソッドが呼び出されるようにします。これにより、同じメソッド名で異なる実装を提供することができ、多態性を実現することができます。

例えば、上記の例で、**Animal**クラスと**Dog**クラスは、**move()**メソッドを持っていますが、**Dog**クラスは**Animal**クラスの**move()**メソッドをオーバーライドしています。これにより、**Animal**型のオブジェクトと**Dog**型のオブジェクトで同じ**move()**メソッドを呼び出すことができますが、それぞれのオブジェクトの実際の型に応じて、異なる動作をすることができます。

・Javaでのメソッドのオーバーライドには、以下の条件があります。

1. メソッド名、引数リスト、戻り値型が一致している必要があります。
2. アクセス修飾子の範囲は、オーバーライドするメソッドよりも狭い、または同じでなければなりません。
3. **final**修飾子が付いているメソッドはオーバーライドできません。
4. **static**修飾子が付いているメソッドはオーバーライドできません。
5. 親クラスで定義されたメソッドが**private**である場合、オーバーライドすることはできません。

また、オーバーライドする際には、以下の点に注意する必要があります。

1. オーバーライドしたメソッドの実装は、親クラスで定義されたメソッドの実装を上書きする必要があります。
2. オーバーライドしたメソッドの実装が、親クラスで定義されたメソッドと同じ場合、オーバーライドする必要はありません。
3. オーバーライドしたメソッドの戻り値型は、親クラスで定義されたメソッドの戻り値型と互換性がある必要があります。
4. オーバーライドしたメソッドの引数リストは、親クラスで定義されたメソッドの引数リストと同じである必要があります。
5. オーバーライドしたメソッドで**@Override**アノテーションを使用することが推奨されます。これにより、コンパイラがオーバーライドが正しく行われているかどうかをチェックし、エラーを検出することができます。