# Javaプログラミング実習

# 30. アノテーション

株式会社ジードライブ

# 今回学ぶこと

- アノテーションとは
- 代表的なアノテーションとその役割

# アノテーションとは

- コンパイル時や実行時に参照される、プログラムに関する注釈
  - アノテーションをどのように解釈して何を行うかはアノテーションの種類によって異なる

    例：オーバーライドの明示

    コンパイル時の警告の抑制

    エラーチェック内容の指定、等
- @～という形式で記述される

# @Override

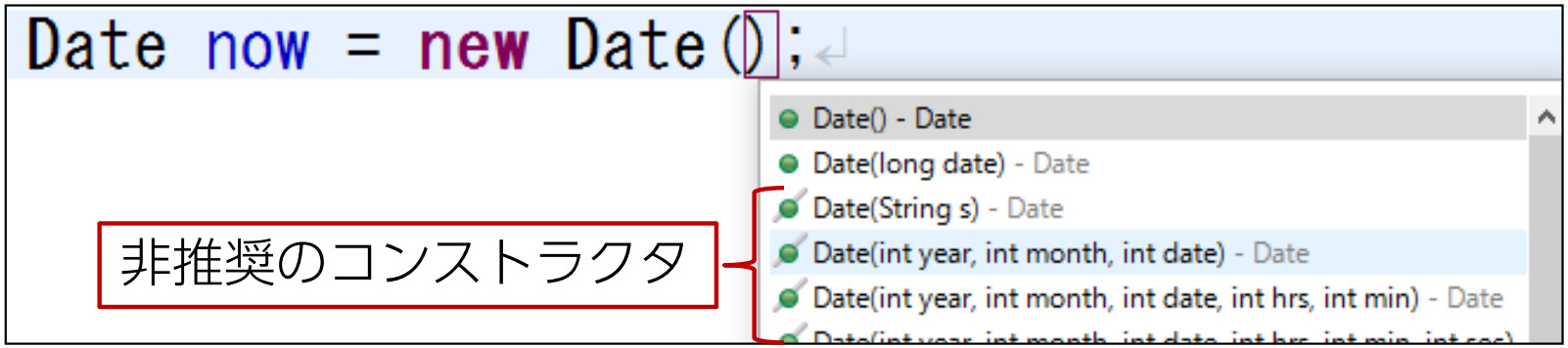
- メソッドをオーバーライドしている事をコンパイラに伝えるためのアノテーション
  - オーバーライドしたメソッドの戻り値の型や引数が違っている場合はコンパイラがエラーを発生させて教えてくれる

使用例

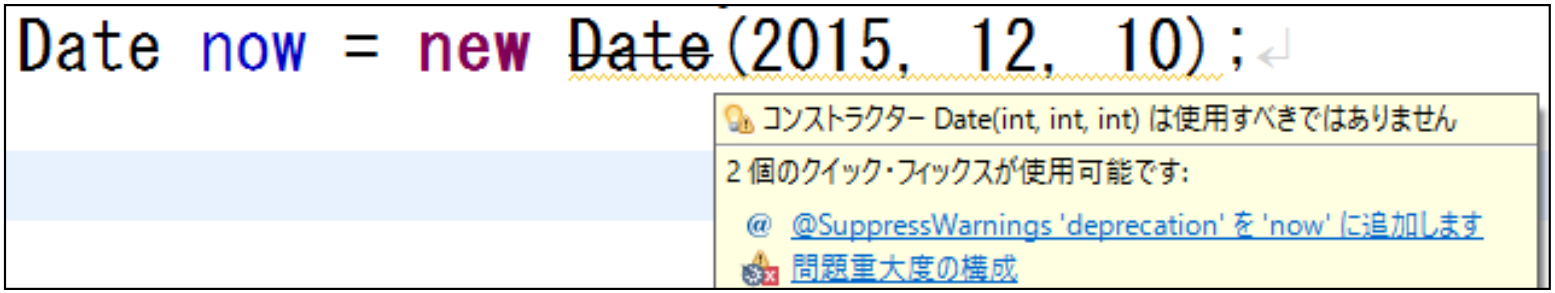
```
public class Car {
    @Override
    public String toString() {
        return "Carクラスです";
    }
}
```

# @Deprecated

- 非推奨のクラスやメンバに付加する(されている)
  - @Deprecatedが付いたクラスやメソッドを利用しようとするとEclipse上で教えてくれる




非推奨のコンストラクタを使用



# @SuppressWarnings

- コンパイラが検出するワーニングの発生を抑制する(警告が表示されなくなる)
  - Suppress: 抑制する
  - ( ) 内に抑制したいワーニングの種類を指定する

使用例

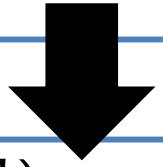
```
public class DateTest {
    @SuppressWarnings("deprecation")
    Date now = new Date(2015, 12, 10);
}
```

# アノテーションの利便性

- Javaのフレームワークでは、様々なアノテーションを使用する
  ⇒煩雑になりがちな記述を省略できる

アノテーション未使用

```
if(age == null) {
    errorMessage = "年齢が未入力です";
} else if(age < 0) {
    errorMessage = "不正な値です";
}
```



アノテーション使用

```
@NotNull("年齢が未入力です")
@Min(value = 0, message = "不正な値です")
private Integer age;
```