

# Adding Metadata with Java Annotations

---

## UNDERSTANDING ANNOTATIONS



**Jim Wilson**

MOBILE SOLUTIONS DEVELOPER & ARCHITECT

@hedgehogjim jwhh.com



# Overview



**The role of annotations**

**Adding metadata with annotations**

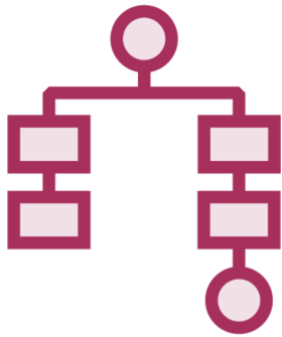
**Commonly used annotations**

**Declaring custom annotations**

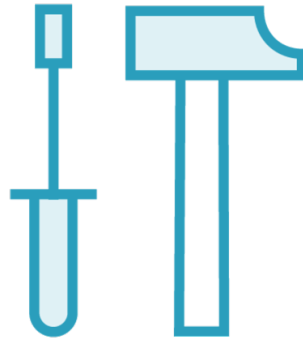


# Programs Fit Into a Larger Picture

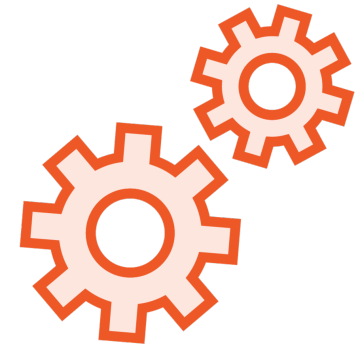
Incorporate developer's assumptions



About the type system

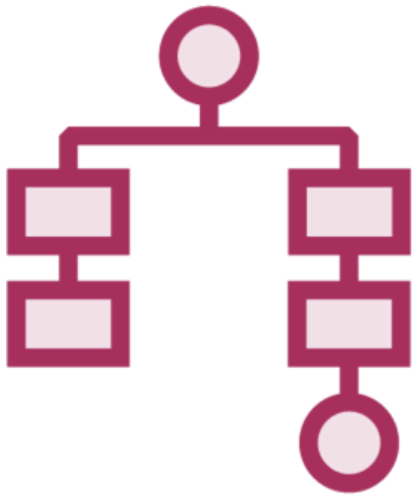


About the toolset



About the execution  
environment

# Programs Incorporate Context and Intent



Type system solves much of the issue



But standard type system isn't enough


```
public class Person {  
    private String name;  
    private int age;  
  
    public Person(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
  
    public int compareTo(Person person) {  
        int compare = name.compareTo(person.name);  
        return compare != 0 ? compare : person.age - age;  
    }  
}
```

```
public class Person implements Comparable<Person> {  
    private String name;  
    private int age;  
  
    public Person(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
  
    public int compareTo(Person person) {  
        int compare = name.compareTo(person.name);  
        return compare != 0 ? compare : person.age - age;  
    }  
}
```

Person.java

Implied extension of  
Object class

```
public class Person implements Comparable<Person> {  
    private String name;  
    private int age;  
  
    public Person(String name, int age) { . . . }  
    public int compareTo(Person person) { . . . }  
  
    public String toString() {  
        return name;  
    }  
}
```



# We Often Manually Supplement Type System



A priori knowledge



Code comments



Documentation



# We Need a Better Way



**Need a structured way to express  
context and intent**



**Allow tools and other code to  
act on context and intent**



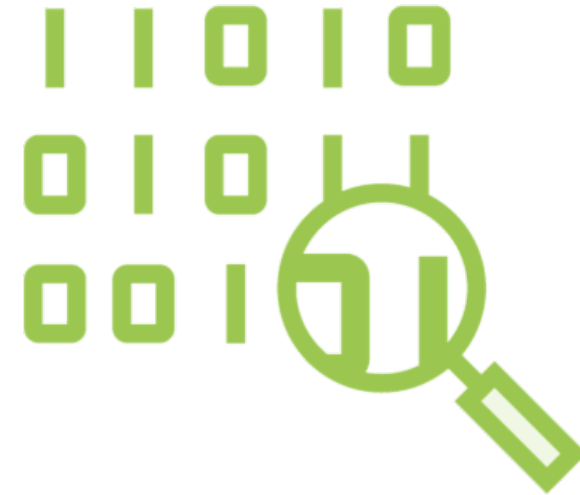
# Annotations



**Special types that act as metadata**

Applied to a specific target

No direct affect on target behavior



**Must be interpreted**

Tools

Execution environments

Any program

# Applying Annotations



Annotation always  
preceded by @



Placed directly before  
the target



Allowable targets vary  
with annotation

```
public class Person implements Comparable<Person> {  
    private String name;  
    private int age;  
  
    public Person(String name, int age) {  
        .  
        .  
        .  
    }  
  
    public int compareTo(Person person) {  
        .  
        .  
        .  
    }  
  
    @Override  
    public String toString() {  
        return name;  
    }  
}
```

Indicates method is intended to  
override an inherited method

Produces an error if there is  
no method with a matching  
signature to override

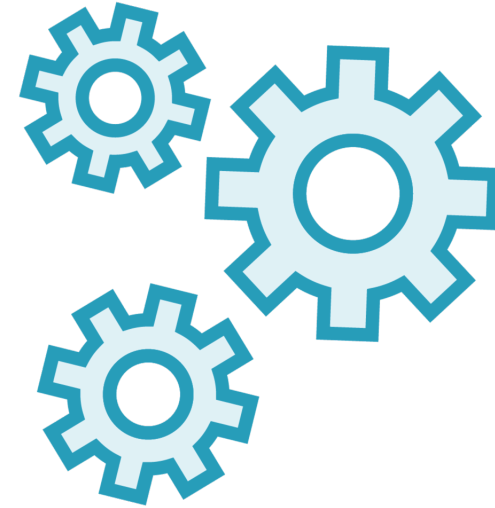
# Annotations in Code



## Core Java Platform

Provides types for creating annotations

Has only a few annotations



## Widely used by other tools/environments

XML and JSON processors

IntelliJ

Java EE

Your own code





## Common Java core platform annotations

- Override
- Deprecated
- SuppressWarnings



```
public class MyWorker {  
  
    void doSomeWork() {  
        Doer d1 = new Doer();  
        d1.doTheThing();  
    }  
  
    void doDoubleWork() {  
        Doer d2 = new Doer();  
        d2.doTheThing();  
        d2.doTheThing();  
    }  
}
```

```
public class Doer {  
    @Deprecated  
    public void doTheThing(){ . . . }  
    public void doTheThingNew(){ . . . }  
}
```



```
@SuppressWarnings("deprecation")
```

```
public class MyWorker {
```

```
    @SuppressWarnings("deprecation")
```

```
    void doSomeWork() {
```

```
        Doer d1 = new Doer();
```

```
        d1.doTheThing();
```

```
    }
```

```
    @SuppressWarnings("deprecation")
```

```
    void doDoubleWork() {
```

```
        Doer d2 = new Doer();
```

```
        d2.doTheThing();
```

```
        d2.doTheThing();
```

```
    }
```

```
}
```

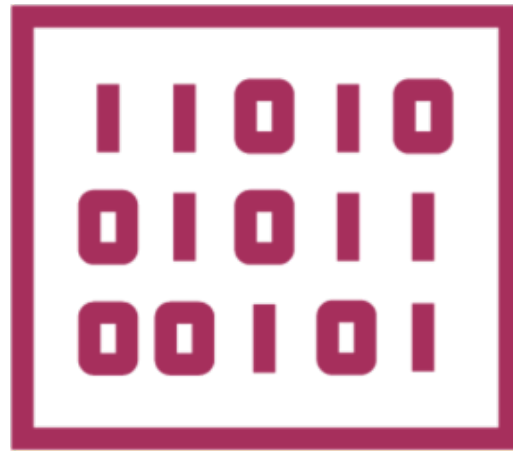
Produces warning  
when compiled

Produces warning  
when compiled

# Declaring Annotations



Can declare custom  
annotations



Acts as custom  
metadata



Same capabilities as  
built-in annotations

# Declaring Annotations

**Annotations are a special kind of interface**

- Types cannot explicitly implement

**Declared with interface keyword**

- Preceded by @ symbol
- Implicitly extend Annotation interface
- Can be declared as a top-level type
- Can be nested within a class or interface

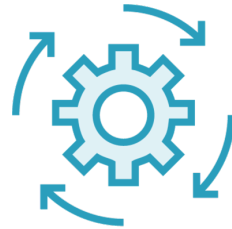


# Declaring Annotations

Annotations can optionally  
have elements



Specify values within  
annotation



Declared as methods



Set similar to fields



# Summary



## Annotations

- Associate metadata with a type
- Extends the ability to express context and intent

## Affect of annotations

- No direct affect on type
- Must be interpreted

# Summary



## Using annotations

- Annotation name preceded by @
- Annotations can be applied to a type or its members



# Summary



## Declared with interface keyword

- Must be preceded by @ symbol

## Annotations can have elements

- Allows specifying values
- Declared similar to methods
- Set similar to fields

