Java Annotations Processing

durga.p

Agenda

Motivations for Annotations

Custom Annotations

Intro to Annotation Processing

- Libraries

$v = \frac{dx}{dt}$ (velocity is the slope of an x vs t graph)

$a = \frac{dv}{dt}$ (accel is the slope of a v vs t graph)

$\Delta x = \int v\,dt$ (displacement is the area under a v vs t graph)

$\Delta v = \int a\,dt$ (change in velocity is the area under an a vs t graph)

For constant acceleration ONLY:

$\Delta x = v_0 t + \frac{1}{2} a t^2$

$\Delta x = \frac{1}{2}(v_0 + v)t$

$v = v_0 + at$

$v^2 = v_0^2 + 2a\Delta x$

Label or metadata of something

Annotations in code?

I never annotated any textbooks or formulas why code?

My code is **so** expressive that I don’t even put comments or javadocs why annotations?

My favourite language has no annotation support, which means they are not really needed.

Convention over Configuration

Configuration:

Using a config (file) based approach to interact with a framework

Convention:

Assuming and expecting a convention to interact with a framework

Convention over Configuration

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
 "-//Hibernate/Hibernate Mapping DTD//EN"
 "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
   <class name="Employee" table="EMPLOYEE">
      <meta attribute="class-description">
         This class contains the employee detail.
      </meta>
      <id name="id" type="int" column="id">
         <generator class="native"/>
      </id>
      <property name="firstName" column="first_name" type="string"/>
      <property name="lastName" column="last_name" type="string"/>
      <property name="salary" column="salary" type="int"/>
   </class>
</hibernate-mapping>
```

```java
@Entity
@Table(name = "EMPLOYEE")
public class Employee {
   @Id @GeneratedValue
   @Column(name = "id")
   private int id;

   @Column(name = "first_name")
   private String firstName;

   @Column(name = "last_name")
   private String lastName;

   @Column(name = "salary")
   private int salary;
```

Convention over Configuration

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-app_2_4.xsd">

    <servlet>
        <servlet-name>Servlet Name For Demo1</servlet-name>
        <servlet-class>com.mkyong.ServletDemo1</servlet-class>
    </servlet>

    <servlet-mapping>
        <servlet-name>Servlet Name For Demo1</servlet-name>
        <url-pattern>/Demo1</url-pattern>
    </servlet-mapping>
</web-app>
```

```java
@Path("/hello-world")
@Produces(MediaType.APPLICATION_JSON)
public class HelloWorldResource {
    private final String template;
    private final String defaultName;
    private final AtomicLong counter;

    public HelloWorldResource(String template, String defaultName) {
        this.template = template;
        this.defaultName = defaultName;
        this.counter = new AtomicLong();
    }

    @GET
    @Timed
    public Saying sayHello(@QueryParam("name") Optional<String> name) {
        final String value = String.format(template, name.or(defaultName));
        return new Saying(counter.incrementAndGet(), value);
    }
}
```

Aspect Oriented Programming

Program as set of concerns and solve for crosscutting concerns. By adding functionality to existing code without modifying the code itself.

Examples:

- log arguments of all functions with prefix “set”
- Monitor execution time for all functions with prefix “fetch”

Aspect Oriented Programming

```java
public boolean register(String sellerId) {
  long start = System.currentTimeMillis();
  if(sellerId == null){
    long stop = System.currentTimeMillis();
    measure(start-stop);
    return false;
  }
  else if (sellerId.length() > 0) {
    long stop = System.currentTimeMillis();
    measure(start-stop);
    return false;
  }
  else {
    //Register
    long stop = System.currentTimeMillis();
    measure(start-stop);
    return true;
  }
}
```

```java
@Timed
public boolean register(String sellerId) {
  if(sellerId == null){
    return false;
  }
  else if (sellerId.length() > 0) {
    return false;
  }
  else {
    //Register
    return true;
  }
}
```

Why Java Annotations

- Enable Java to do modern approaches
  - convention over configuration
  - aspect oriented programming
- Communicate with Java pipeline
  - Compiler
  - Build Tool

What are Java Annotations

Labelling of Java code elements

Interpreted to have desired effect.

Have no direct effect on the code

JDK Examples

@Override

@Deprecated

@SuppressWarnings

@Deprecated

```
public class User {

  private String userName;
  private String phoneNumber;

  @Deprecated
  public String getPhoneNumber() {
    return phoneNumber;
  }

  public Collection<String> getPhoneNumbers() {
    return Arrays.asList(phoneNumber);
  }

  public void setUserName(String userName) {
    this.userName = userName;
  }
```

@SupressWarning @Override

```
public class Seller extends User {

  private String sellerName;

  @Override
  public void setUserName(String userName) {
    super.setUserName(userName);
    this.sellerName = userName + "_seller";
  }

  @SuppressWarnings({"deprecation"})
  public String getIdentifier() {
    return super.getPhoneNumber() + this.sellerName;
  }
}
```

Android Support Annotation

DimenRes	Denotes that an integer parameter, field or method return value is expected to be a dimension resource reference (e.g.
DrawableRes	Denotes that an integer parameter, field or method return value is expected to be a drawable resource reference (e.g.

FloatRange	Denotes that the annotated element should be a float or double in the given range
	Example:
	`@FloatRange(from=0.0,to=1.0)`
	`public float getAlpha() {`
	`...`
	`}`

FractionRes	Denotes that an integer parameter, field or method return value is expected to be a fraction resource reference.
IdRes	Denotes that an integer parameter, field or method return value is expected to be an id resource reference (e.g.
IntDef	Denotes that the annotated element of integer type, represents a logical type and that its value should be one of the explicitly named constants.
IntegerRes	Denotes that an integer parameter, field or method return value is expected to be an integer resource reference (e.g.
InterpolatorRes	Denotes that an integer parameter, field or method return value is expected to be an interpolator resource reference (e.g.
IntRange	Denotes that the annotated element should be an int or long in the given range Example:

Custom Annotations

```
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
public @interface Builder {
}
```

Label *something* **(**what - **Target).**

Interpreted **(**when -**Retention)** for desired effect.

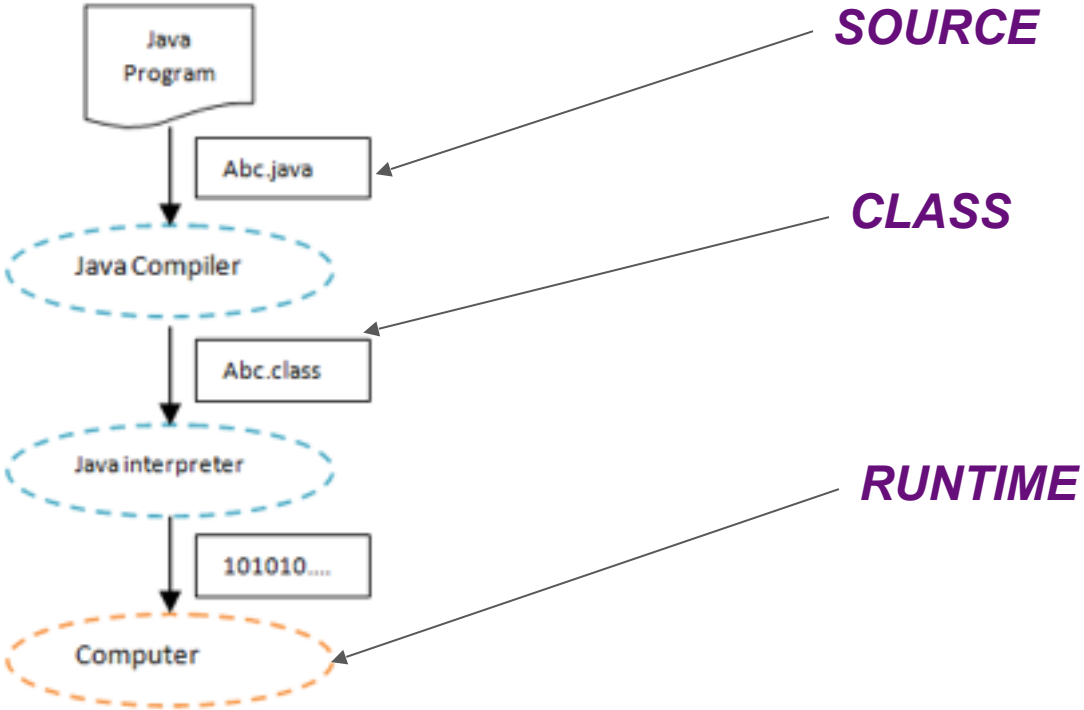
*value()

Target (What)

	Answer
FIELD,	**Two**,
METHOD,	**One**,
PARAMETER,	**Five**,
CONSTRUCTOR,	**Four**,
TYPE,	**Three**,

```
@Three
public class MyClass {

  @Two
  public MyClass(String myField) {
    this.myField = myField;
  }

  @Four
  private String myField;

  @One
  public boolean myMethod(@Five String argument) {
    return this.myField.equals(argument);
  }

}
```

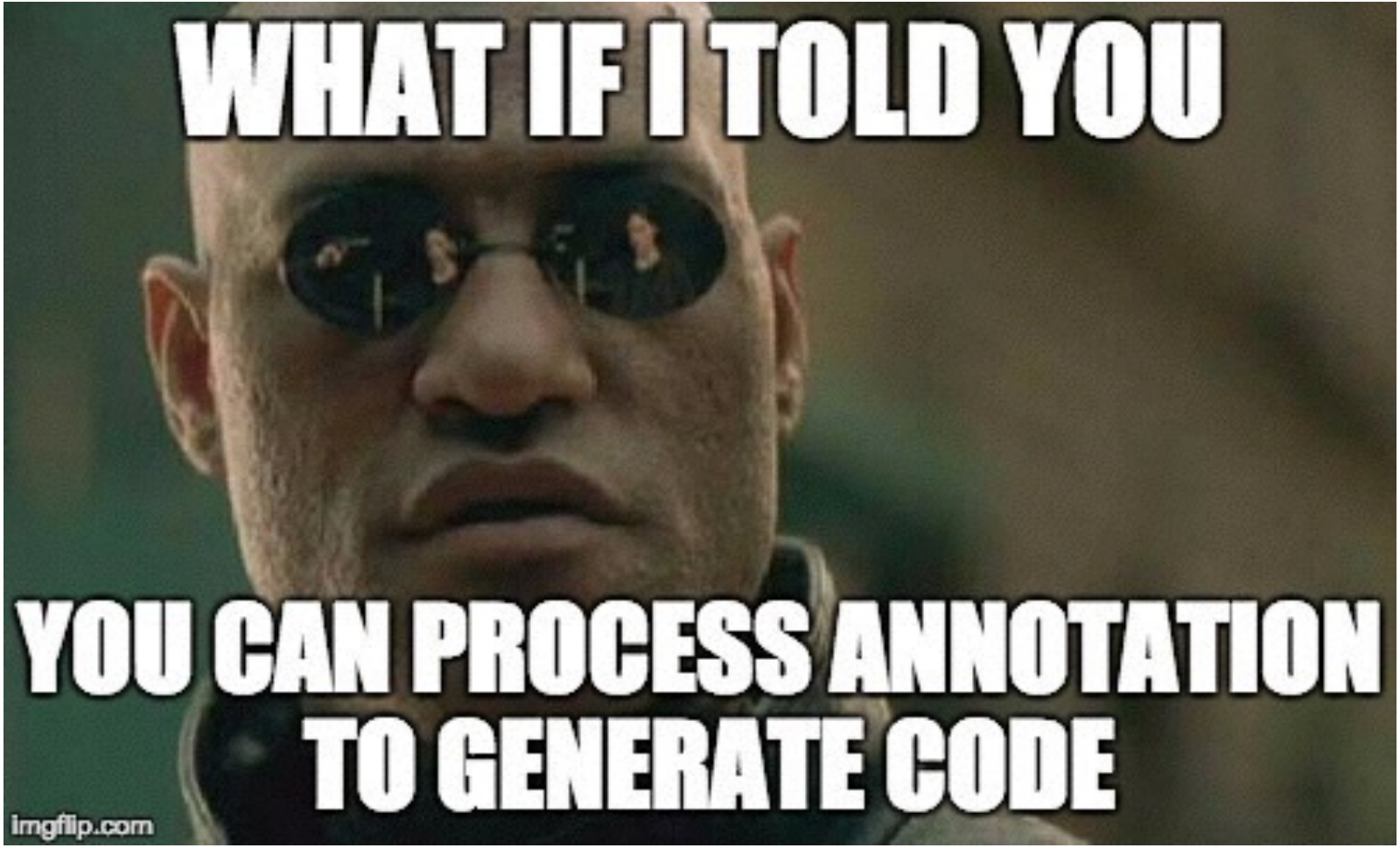
Retention (~~When~~ how long)



Retention	Answers
@Override	Source
@Deprecated	Runtime
@SuppressWarnings	Source

Runtime Annotations

- Interpreted at Runtime
  - By normal code
  - *via Guice, Jersey etc
- For a backend service like seller-mapi
  - SecurityLevel
  - For InternalUseOnly
  - RateLimiting


WHAT IF I TOLD YOU
YOU CAN PROCESS ANNOTATION TO GENERATE CODE
imgflip.com

Annotation Processing

Usecases

lombok

- Getter Setter
- NoArgsConstructor, AllArgsConstructor
- EqualsAndHashCode
- Slf4j

lombok

```java
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@EqualsAndHashCode
public class Employee {

  private String userName;
  private String emailId;

}
```

```java
//Setter example
//Noargs constructor
Employee employee = new Employee();
employee.setEmailId("test@flipkart.com");
employee.setUserName("test");

//Getter example
String emailId = employee.getEmailId();
String userName = employee.getUserName();

//Required args constructor
Employee
    reqEmployee = new Employee("test","test@flipkart.com");

//Returns True
reqEmployee.equals(employee);
```

Slf4j (lombok)

```java
@Slf4j
public class Company {

  public static void main(String[] args) {
    log.info("testing @Sl4fj");
  }
}
```

Android Use Cases

Disclaimer

Android Use Cases

Save/Restore Instance States: IcePick (https://github.com/frankiesardo/icepick)

```
class ExampleActivity extends Activity {
  @State String username; // This will be automatically saved and restored

  @Override public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    Icepick.restoreInstanceState(this, savedInstanceState);
  }

  @Override public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    Icepick.saveInstanceState(this, outState);
  }
```

Runtime permissions: PermissionsDispatcher (https://github.com/hotchemi/PermissionsDispatcher)

```
@RuntimePermissions
public class MainActivity extends AppCompatActivity {

    @NeedsPermission(Manifest.permission.CAMERA)
    void showCamera() {
        getSupportFragmentManager().beginTransaction()
                .replace(R.id.sample_content_fragment, CameraPreviewFragment.newInstance())
                .addToBackStack("camera")
                .commitAllowingStateLoss();
    }

    @OnShowRationale(Manifest.permission.CAMERA)
    void showRationaleForCamera(PermissionRequest request) {
        new AlertDialog.Builder(this)
            .setMessage(R.string.permission_camera_rationale)
            .setPositiveButton(R.string.button_allow, (dialog, button) -> request.proceed())
            .setNegativeButton(R.string.button_deny, (dialog, button) -> request.cancel())
            .show();
    }

    @OnPermissionDenied(Manifest.permission.CAMERA)
    void showDeniedForCamera() {
        Toast.makeText(this, R.string.permission_camera_denied, Toast.LENGTH_SHORT).show();
    }

    @OnNeverAskAgain(Manifest.permission.CAMERA)
    void showNeverAskForCamera() {
        Toast.makeText(this, R.string.permission_camera_neverask, Toast.LENGTH_SHORT).show();
    }
}
```

```
case R.id.button_camera:
    // NOTE: delegate the permission handling to generated method
    MainActivityPermissionsDispatcher.showCameraWithCheck(this);
```

Parceling Pojos: Auto-Parcel (https://github.com/frankiesardo/auto-parcel)

```
@AutoParcel
abstract class SomeModel implements Parcelable {
  abstract String name();
  abstract List<SomeSubModel> subModels();
  abstract Map<String, OtherSubModel> modelsMap();

  static SomeModel create(String name, List<SomeSubModel> subModels, Map<String, OtherSubModel> modelsMa
    return new AutoParcel_SomeModel(name, subModels, modelsMap);
  }
}
```

Fragment Arguments: fragmentargs (https://github.com/sockeqwe/fragmentargs)

```
@FragmentWithArgs
public class MyFragment extends Fragment {

    @Arg
    int id;

    @Arg
    private String title; // private fields requires a setter method

    @Override
    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        FragmentArgs.inject(this); // read @Arg fields
    }
```

```
public class MyActivity extends Activity {

    public void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);

        int id = 123;
        String title = "test";

        // Using the generated Builder
        Fragment fragment =
            new MyFragmentBuilder(id, title)
            .build();
```

AndroidAnnotations (https://github.com/excilys/androidannotations/wiki)

```
@EActivity(R.layout.translate) // Sets content view to R.layout.translate
public class TranslateActivity extends Activity {

    @ViewById // Injects R.id.textInput
    EditText textInput;

    @ViewById(R.id.myTextView) // Injects R.id.myTextView
    TextView result;

    @AnimationRes // Injects android.R.anim.fade_in
    Animation fadeIn;

    @Click // When R.id.doTranslate button is clicked
    void doTranslate() {
         translateInBackground(textInput.getText().toString());
    }

    @Background // Executed in a background thread
    void translateInBackground(String textToTranslate) {
         String translatedText = callGoogleTranslate(textToTranslate);
         showResult(translatedText);
    }

    @UiThread // Executed in the ui thread
    void showResult(String translatedText) {
         result.setText(translatedText);
         result.startAnimation(fadeIn);
    }

    // [...]
}
```

ButterKnife

Dagger

Retrofit

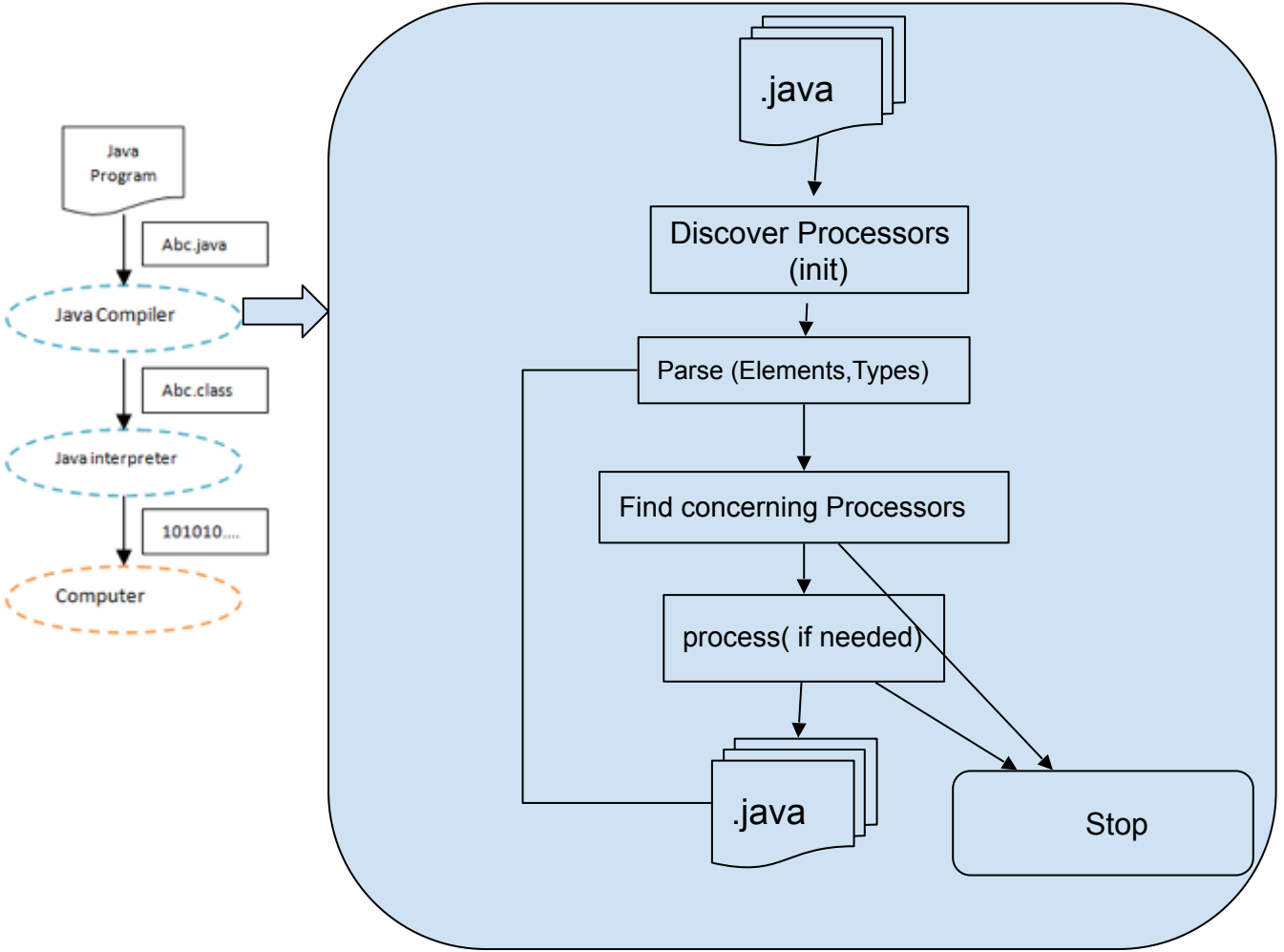
EventBus3

...

Custom Annotation Processor

Processor

```
public interface Processor {
    /**...*/
    Set<String> getSupportedOptions();

    /**...*/
    Set<String> getSupportedAnnotationTypes();

    /**...*/
    SourceVersion getSupportedSourceVersion();

    /**...*/
    void init(ProcessingEnvironment processingEnv);

    /**...*/
    boolean process(Set<? extends TypeElement> annotations,
                    RoundEnvironment roundEnv);
```


Java
Program
Abc.java
Java Compiler
Abc.class
Java interpreter
101010....
Computer
.java
Discover Processors
(init)
Parse (Elements,Types)
Find concerning Processors
process(if needed)
.java
Stop

AnnotationProcessing - Lifecycle

Automatically runs with javac

Happens after parsing and before .class generation

Till no new files are generated

Has its own JVM

- Bring your own libraries
- Compiled sources and its dependencies are not available

Builder Annotation

```java
package com.solvevolve.atp.annotationprocessing;

import java.util.Collection;

@Builder
public class Employee {

  private String userName;
  private String emailId;

  public String getUserName() {
    return userName;
  }

  public void setUserName(String userName) {
    this.userName = userName;
  }

  public String getEmailId() {
    return emailId;
  }

  public void setEmailId(String emailId) {
    this.emailId = emailId;
  }

}
```

```java
package com.solvevolve.atp.annotationprocessing;

import java.lang.String;

public class EmployeeBuilder {

  private Employee holder;

  public Employee build() { return holder; }

  public EmployeeBuilder userName(String userName) {
    this.holder.setUserName(userName);
    return this;
  }

  public EmployeeBuilder emailId(String emailId) {
    this.holder.setEmailId(emailId);
    return this;
  }

}
```

Processor

```
public interface Processor {
    /**...*/
    Set<String> getSupportedOptions();

    /**...*/
    Set<String> getSupportedAnnotationTypes();

    /**...*/
    SourceVersion getSupportedSourceVersion();

    /**...*/
    void init(ProcessingEnvironment processingEnv);

    /**...*/
    boolean process(Set<? extends TypeElement> annotations,
                    RoundEnvironment roundEnv);
```

Custom AnnotationProcessor

```
@AutoService(Processor.class)
public class BuilderAnnotationProcessor extends AbstractProcessor {

  @Override
  public Set<String> getSupportedAnnotationTypes() {
    return Sets.newHashSet(
        "com.solvevolve.atp.annotationprocessing.Builder");
  }

  @Override
  public SourceVersion getSupportedSourceVersion() {
    return SourceVersion.latestSupported();
  }

  @Override
  public synchronized void init(ProcessingEnvironment processingEnv) {

  }

  @Override
  public boolean process(Set<? extends TypeElement> annotations,
                         RoundEnvironment roundEnv) {
    return false;
  }

}
```

Custom AnnotationProcessor

```java
@SupportedAnnotationTypes(
    "com.solvevolve.atp.annotationprocessing.Builder"
)
@SupportedSourceVersion(SourceVersion.RELEASE_7)
@AutoService(Processor.class)
public class BuilderAnnotationProcessor extends AbstractProcessor {

  @Override
  public synchronized void init(ProcessingEnvironment processingEnv) {
  }

  @Override
  public boolean process(Set<? extends TypeElement> annotations,
                         RoundEnvironment roundEnv) {
    return false;
  }

}
```

Registering your Processor

Processor instances are loaded by javac using Java's ServiceLoader.

ServiceLoader expects META-INF/services/javax.annotation.processing.Processor

Google's AutoService

Processing Environment

```
public class BuilderAnnotationProcessor extends AbstractProcessor {

  private Filer filer;
  private Messager messager;
  private Types typeUtils;
  private Elements elementUtils;

  @Override
  public synchronized void init(ProcessingEnvironment processingEnv) {
    this.filer = processingEnv.getFiler();
    this.messager = processingEnv.getMessager();
    this.typeUtils = processingEnv.getTypeUtils();
    this.elementUtils = processingEnv.getElementUtils();
  }
```

Messenger

Channel to communicate messages back to IDE/user

Allows multiple levels like ERROR, WARNING, NOTE

Allows messages to be shown on corresponding element

```
messager.printMessage(
    Diagnostic.Kind.ERROR,
    "Applicable only to top level Classes",
    rootElement);
```

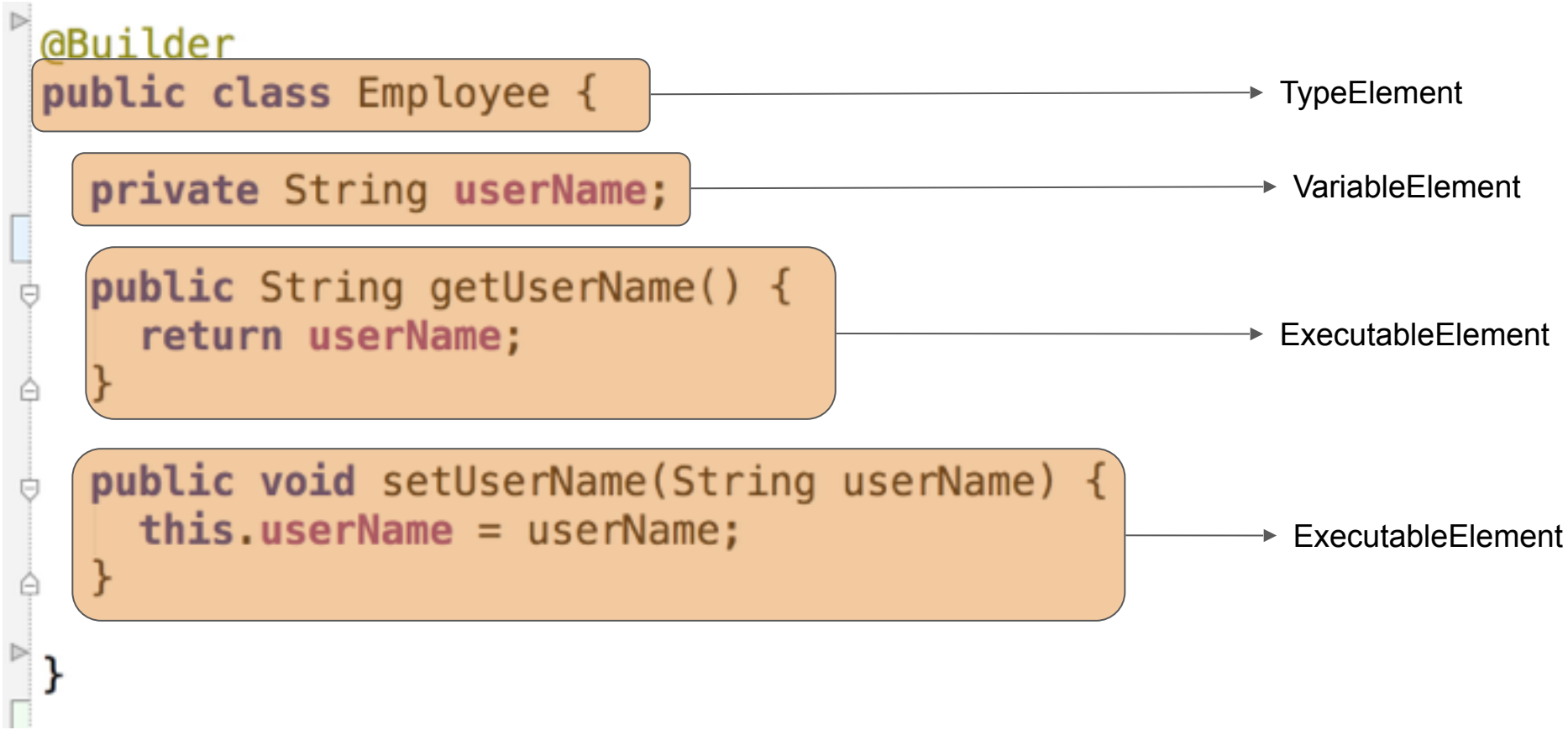
Filer

Used to generate output Files

Helper libraries to output code

Java Poet (Github: https://github.com/square/javapoet)

CodeModel(https://codemodel.java.net/)


@Builder
public class Employee {
private String userName;
public String getUserName() {
return userName;
}
public void setUserName(String userName) {
this.userName = userName;
}
}
TypeElement
VariableElement
ExecutableElement
ExecutableElement

Elements

Code is tokenized into **Element** s (javax.lang.model.element).

Element are

- TypeElement
- VariableElement
- ExecutableElement etc

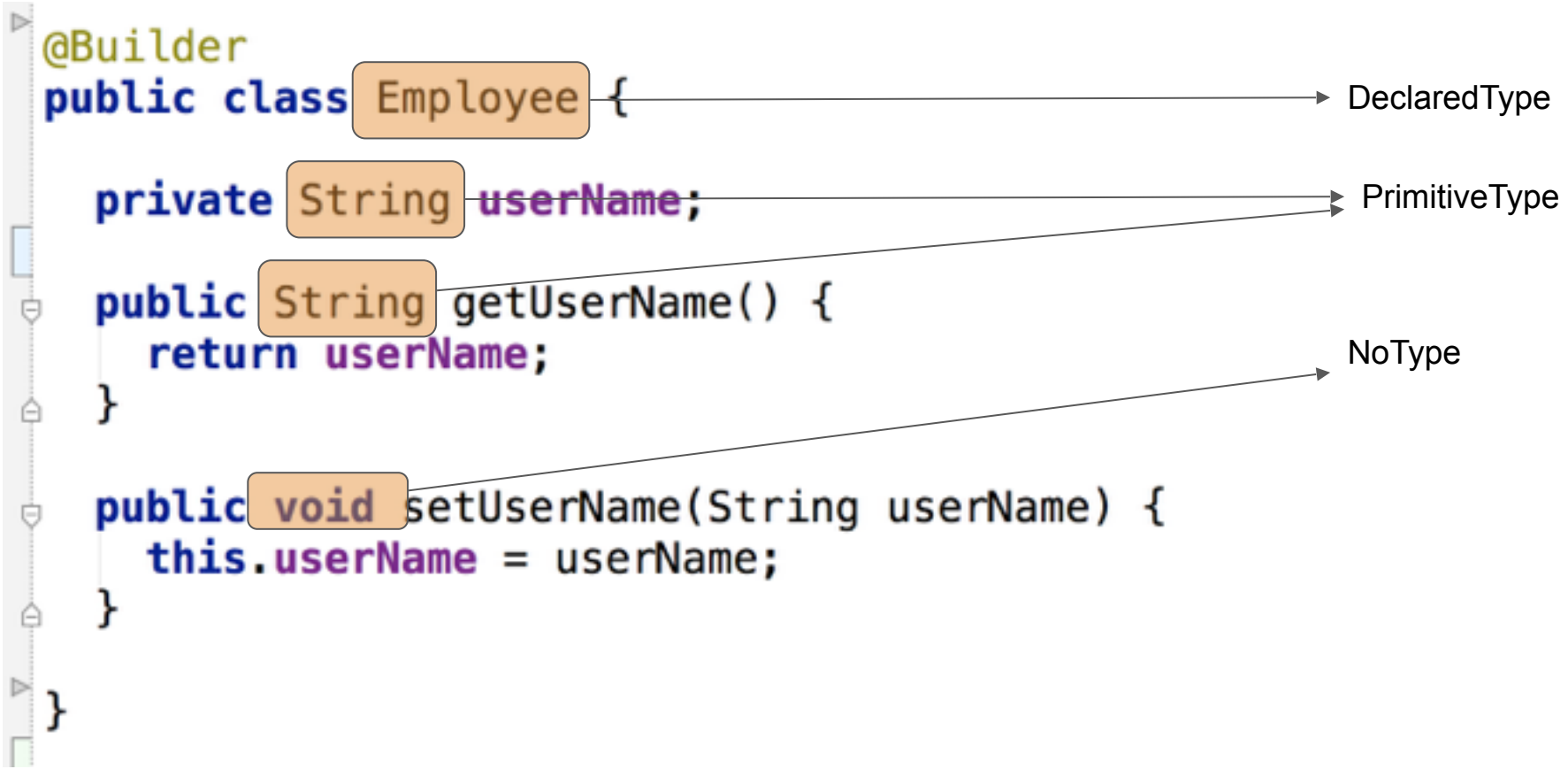
Element allows

- getEnclosedElements() -> To get children
- getEnclosingElement() -> To get Parent

processingEnv.getElementUtils() help work with elements

- elementUtils.getTypeElement(“test.User”) gives TypeElement of said class
- elementUtils.getPackageOf(element) gives package for any Element

Note: Use element.getKind() == ElementKind.CLASS etc. Don’t rely on instanceOf


@Builder
public class Employee {
private String userName;
public String getUserName() {
return userName;
}
public void setUserName(String userName) {
this.userName = userName;
}
}
DeclaredType
PrimitiveType
NoType

Types

- Holds the Type representation of element (element.asType())
- TypeMirror (javax.lang.model.type)
  - DeclaredType
  - PrimitiveType, ReferenceType, NullType
  - ExecutableType
- processingEnv.getTypeUtils() helps working with Type
  - to check if one type extends other or for inner classes
  - to work with Generics

Note: Use TypeMirror.getKind to find to which to cast to

Guess Element or Type

Method’s name

- executableElement.getName()

Method’s return type

- executableElement.getReturnType()

Class of a field

- variableElement.asType()

Field name

- variableElement.getSimpleName()

Exceptions thrown by a method

- executableElement.getThrownTypes()

A methods parameter name

- executableElement.getParameters()

Custom AnnotationProcessor

```java
@SupportedAnnotationTypes(
    "com.solvevolve.atp.annotationprocessing.Builder"
)
@SupportedSourceVersion(SourceVersion.RELEASE_7)
@AutoService(Processor.class)
public class BuilderAnnotationProcessor extends AbstractProcessor {

  @Override
  public synchronized void init(ProcessingEnvironment processingEnv) {
  }

  @Override
  public boolean process(Set<? extends TypeElement> annotations,
                         RoundEnvironment roundEnv) {
    return false;
  }

}
```

RoundEnvironment

```
public interface RoundEnvironment {
    /**...*/
    boolean processingOver();

    /**...*/
    boolean errorRaised();

    /**...*/
    Set<? extends Element> getRootElements();

    /**...*/
    Set<? extends Element> getElementsAnnotatedWith(TypeElement a);

    /**...*/
    Set<? extends Element> getElementsAnnotatedWith(Class<? extends Annotation> a);
}
```

Use to get Elements you care about

Builder Annotation process()

```
@Override
public boolean process(Set<? extends TypeElement> annotations, RoundEnvironment roundEnv) {
  if (!roundEnv.processingOver() && !annotations.isEmpty()) {
    TypeElement builderAnnotation = annotations.iterator().next();
    Set<? extends Element> elementsAnnotated = roundEnv.getElementsAnnotatedWith(builderAnnotation);
    for (Element element : elementsAnnotated) {
      if(isTopLevelClass(element)) {
        BuilderModel model = constructModel((TypeElement) element);
        generateBuilder(model);
      }
      else {
        this.messager.printMessage(Diagnostic.Kind.ERROR,
                                   "Applicable only to top level Classes", element);
      }
    }
  }
  return false;
}

private boolean isTopLevelClass(Element element) {
  return element.getKind() == ElementKind.CLASS
         || element.getEnclosingElement().getKind() == ElementKind.PACKAGE;
}
```

Builder Annotation constructModel()

```
private BuilderModel constructModel(TypeElement typeElement) {
  BuilderModel builderModel = new BuilderModel();
  builderModel.annotatedElement = typeElement;
  builderModel.annotatedPackage = elementUtils.getPackageOf(typeElement);

  for (Element element : typeElement.getEnclosedElements()) {
    if (element.getKind() == ElementKind.METHOD && isSetterMethod((ExecutableElement) element)) {
      builderModel.setterMethods.add((ExecutableElement) element);
    }
  }
  return builderModel;
}

private boolean isSetterMethod(ExecutableElement executableElement) {
  return executableElement.getReturnType().getKind() == TypeKind.VOID
         &&
         executableElement.getSimpleName().subSequence(0, 3).equals("set")
         &&
         executableElement.getParameters().size() == 1;
}
```

```
private void generateBuilder(BuilderModel model) {

  TypeSpec.Builder builder = TypeSpec.classBuilder(model.getNameOfClassToBeBuilt())
      .addModifiers(Modifier.PUBLIC)
      .addField(ClassName.get(model.annotatedElement.asType()), "holder", Modifier.PRIVATE)
      .addMethod(getBuildMethodSpec(model));

  ClassName builderClassName =
     ClassName.get(model.getPackageName(), model.getNameOfClassToBeBuilt());
  for (ExecutableElement setterMethodElement : model.setterMethods) {
    MethodSpec setter = getSetterMethodSpec(builderClassName, setterMethodElement);
    builder.addMethod(setter);
  }
  JavaFile javaFile = JavaFile.builder(model.getPackageName(), builder.build()).build();
  try {
    javaFile.writeTo(filer);
  } catch (IOException e) {
    e.printStackTrace();
  }
}
```

```
private MethodSpec getBuildMethodSpec(BuilderModel model) {
  return MethodSpec.methodBuilder("build")
      .addModifiers(Modifier.PUBLIC)
      .returns(ClassName.get(model.annotatedElement.asType()))
      .addStatement("return holder")
      .build();
}

private MethodSpec getSetterMethodSpec(ClassName builderClassName,
                                       ExecutableElement setterMethodElement) {
  String methodName = setterMethodElement.getSimpleName().toString();
  VariableElement variableElement = setterMethodElement.getParameters().get(0);
  String paramName = variableElement.getSimpleName().toString();
  TypeMirror paramType = variableElement.asType();
  //builder methods
  return MethodSpec.methodBuilder(paramName)
      .addModifiers(Modifier.PUBLIC)
      .returns(builderClassName)
      .addParameter(ClassName.get(paramType), paramName)
      .addStatement("this.holder.$L($L)", methodName, paramName)
      .addStatement("return this")
      .build();
}
```

JavaPoet

```
package com.example.helloworld;

public final class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello, JavaPoet!");
  }
}
```

```
MethodSpec main = MethodSpec.methodBuilder("main")
    .addModifiers(Modifier.PUBLIC, Modifier.STATIC)
    .returns(void.class)
    .addParameter(String[].class, "args")
    .addStatement("$T.out.println($S)", System.class, "Hello, JavaPoet!")
    .build();

TypeSpec helloWorld = TypeSpec.classBuilder("HelloWorld")
    .addModifiers(Modifier.PUBLIC, Modifier.FINAL)
    .addMethod(main)
    .build();

JavaFile javaFile = JavaFile.builder("com.example.helloworld", helloWorld)
    .build();

javaFile.writeTo(System.out);
```

Template for annotation processing

- Loop through annotated elements
  - roundEnv.getElementsAnnotatedWith()
- Traverse elements
  - element.getEnclosingElement()
  - element.getEnclosedElements()
- Use elements and types as necessary
- Communicate needed info back via messenger
- Construct a logical Model of the data needed
- Using the model generate output files via JavaPoet

Testing

Using libraries like compile-testing and truth from Google

```
@Test
public void processorTest() {

  JavaFileObject beanFile = JavaFileObjects.forResource("Employee.java");
  JavaFileObject builderFile = JavaFileObjects.forResource("EmployeeBuilder.java");
  assertAbout(javaSource())
          .that(beanFile)
          .processedWith(Collections.singleton(new BuilderAnnotationProcessor()))
          .compilesWithoutError()
          .and()
          .generatesSources(builderFile);
}
```

Thank you