JAVA Programming

Annotations

Overview

- Annotations
- Pre-built annotations
- Reflection
- When to use?

Annotations

- Annotations
  - Declaratively provide meta information
    - Indicate a method is a remote method
    - Meta information about a JavaBean
    - Indicate a method is deprecated
    - …
  - Tools use the annotations to generate code or use the annotation in some other way.
  - Tools rely on reflection API to discover and use the annotations.

Annotations

- Pre-built annotations
  - @Override
    - Compiler checks that method overrides a base class method
    - Applicable to methods and available in source code
  - @Deprecated
    - Compiler warns when used
    - Applicable to all elements and available at runtime
  - @SuppressWarning
    - Compiler suppresses warnings for annotated type
    - Applicable to all program elements and available in source code

Annotations

- Pre-built annotations for use in Annotation definitions (thus on Annotation types):
  - @Documented
    - To be documented by javadoc or similar tools
  - @Inherited
    - If a ***class*** is annotated with such an Annotation then its sub-classes wil also be annotated (implicitly)
  - @Retention
    - Determines if an Annotation is only kept in source code, class code or even at runtime.
  - @Target
    - On what kind of elements is Annotation applicable.

Annotations

- Pre-built annotations continued:
  - @Target
    - For which elements is the annotation valid.
      - TYPE, FIELD, METHOD, PARAMETER, CONSTRUCTOR, LOCAL_VARIABLE, ANNOTATION_TYPE, PACKAGE
    - May specify several targets in an array:
      - @Target(value={ FIELD, METHOD })
  - @Retention
    - Where do Annotations exist

SOURCE	Discarded by compiler
CLASS	Recorded in Class file but not retained in VM
RUNTIME	Annotation also retained by VM

Annotations

- Annotations
  - Are defined as interfaces prefixed with @.
  - Each method represents an element of the annotation type
  - Three Annotation types:

• Marker:	Annotation without arguments
• Single Member:	Annotation with one argument Implicitly named *value*.
• Normal:	Annotation with multiple arguments

Annotations

- Example : Author Annotation

Define:

```
import java.lang.annotation.*;

@Retention(value=RetentionPolicy.RUNTIME)
@Target(ElementType.METHOD)
@Inherited()
public @interface Author
{
     String value() default "Unknown";
}
```

Or:

```
import java.lang.annotation.*;
import static java.lang.annotation.RetentionPolicy.*;
import static java.lang.annotation.ElementType.*;

@Retention(RUNTIME)
@Target(METHOD)
@Inherited()
public @interface Author
{
     String value() default "Unknown";
}
```

Annotations

- Write tool that uses the Annotation (uses reflection)

```
public class AuthorUtil
{
    public static void showAuthors(Class<?> c)
    {
        if ( c.isAnnotationPresent(Author.class))
        {
            System.out.println("Author " + c.getName() + ": "
                    + c.getAnnotation(Author.class).value());
        }

        for (Method m : c.getMethods())
        {
            if (m.isAnnotationPresent(Author.class))
            {
               System.out.println("Author " + m.getName()+ ": "
                         + m.getAnnotation(Author.class).value());
            }
        }
    }
}
```

Annotations

- Annotate your code with annotations:

```
@Author(value="Paul Kramer")
public class GamePlayer
{
    @Author("Timo Kramer")
    public void startGame()
    {    //  Some action                       }

    @Author("Menno Kramer")
    public void joinGame()
    {    //  Some action                       }
}

public class GamePlayerEx extends GamePlayer
{
    @Override()
    public void startGame()
    {    //  override base method  }
}
```

Annotations

- Use tool on annotated classes.

```
public class ShowAuthors
{
    public static void main(String[] args)
    {
        System.out.println("---GamePlayer--");
        AuthorUtil.showAuthors(GamePlayer.class);

        System.out.println("--GamePlayerEx--");
        AuthorUtil.showAuthors(GamePlayerEx.class);
    }
}
```

```
---GamePlayer--
Author demo.GamePlayer: Paul Kramer
Author startGame: Timo Kramer
Author joinGame: Menno Kramer
--GamePlayerEx--
Author demo.GamePlayerEx: Paul Kramer
Author joinGame: Menno Kramer
```

Annotations

- When to use:
  - Annotate your code with pre-built annotations when appropriate
  - Annotations may be a strategic weapon for fulfilling the *ease of development* promise
  - Annotations may ease EJB development
  - Unlikely to build many Annotations yourself (because you have to provide a tool as well)

Lab: Annotations