Java Lecture 9 — Annotations, Enums, JUnit, Files



Lector: Peter Manolov

Skype: nsghost1

E-mail: p.manolov@gmail.com

Facebook: https://www.facebook.com/pmanolov

Copyright © Pragmatic LLC

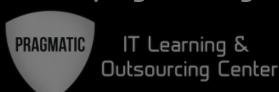
2013 - 201

www.pragmatic.bg

PRAGMATIC IT Learning & Outsourcing Center

Agenda

- Annotations
- Enums
- Unit Test
- Files



- metadata
- No direct effect on the code they annotate
- Uses
 - compiler information
 - compile-time and deployment-time processing
 - runtime processing

PRAGMATIC IT Learning & Outsourcing Center

Annotations

- Format
 - @Override
 - @SuppressWarnings(value = "unchecked")



Elements

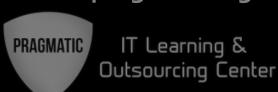
```
@Author(
   name = "Benjamin Franklin",
   date = "3/27/2003"
)
class MyClass() { ... }
```

```
@SuppressWarnings("unchecked")
void myMethod() { ... }
```

```
@SuppressWarnings(value = "unchecked")
void myMethod() { ... }
```

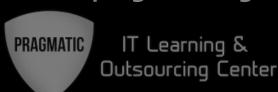


- Applications
 - Declaration of classes
 - Fields
 - Methods
 - Other As of Java 8
 - instance creation expression new @Interned MyObject()
 - type cast myString = (@NonNull String) str;
 - implements clause: class UnmodifiableList<T> implements @ReadOnly List<@Readonly T> {...}
 - thrown exception declaration void monitorTemperature() throws
 @Critical TemperatureException {...}



Declaration

```
@interface ClassPreamble {
  String author();
  String date();
   int currentRevision() default 1;
  String lastModified() default "N/A";
  String lastModifiedBy() default "N/A";
   // Note use of array
  String[] reviewers();
```



Use

```
@ClassPreamble (
   author = "John Doe",
   date = "3/17/2002",
   currentRevision = 6,
   lastModified = "4/12/2004",
   lastModifiedBy = "Jane Doe",
   // Note array notation
   reviewers = {"Alice", "Bob", "Cindy"}
public class Generation3List extends Generation2List {
```



- Predefined Annotations
 - @Deprecated
 - @Override
 - @SuppressWarnings
 - @SafeVarargs
 - @Test (jUnit)

www.pragmatic.bg

Annotations



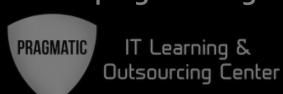
Annotations to annotations

@Retention

- RetentionPolicy.SOURCE
- RetentionPolicy.CLASS
- RetentionPolicy.RUNTIME

@Target

- ElementType.ANNOTATION_TYPE
- ElementType.CONSTRUCTOR
- ElementType.FIELD
- ElementType.LOCAL_VARIABLE
- ElementType.METHOD
- ElementType.PACKAGE
- ElementType.PARAMETER
- ElementType.TYPE



- A predefined list of constants
- Use when a variable can only take one of a small set of possible values
- Increase compile-time checking
- Reduce errors from passing invalid constants
- Document legal values

www.pragmatic.bg

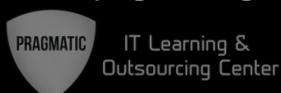


Example

```
enum ChessPiece {
  PAWN,
  KNIGHT,
  BISHOP,
  QUEEN,
  KING;
```

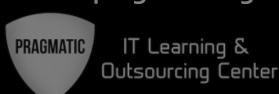


- Implicitly extend java.lang.Enum
- Type-safe
- Can get all possible values with MyEnum.values()
- Implicitly static and final cannot be changed
- Can be compared with "=="
- Cannot create instance with new (constructor is private)
- Can be declared inside and outside of class, but cannot be declared in a method



- Enums declared outside a class must not be static, final, abstract, protected or private
- Can contain constructors, methods, variables
- Constructors can have arguments and be overloaded
- Constructors can never be called directly in code

Unit Test



- Software Testing Method
- Tests Individual Units of code

 Unit Test vs Acceptance Test vs Integration Test vs Performance Test

JUnit

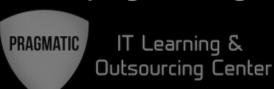


Test Case

Test Suite

```
//JUnit Suite Test
@RunWith(Suite.class)
@Suite.SuiteClasses({
   TestJunit1.class ,TestJunit2.class
})
public class JunitTestSuite {
}
```

Test Case



```
import org.junit.Test;
import static org.junit.Assert.assertEquals;
public class TestJunit1 {
 String message = "Robert";
 MessageUtil messageUtil = new MessageUtil(message);
  @Test
 public void testPrintMessage() {
   System.out.println("Inside testPrintMessage()");
   assertEquals(message, messageUtil.printMessage());
```



Test Case

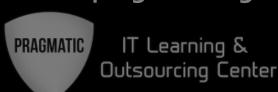
- @Before
- @After
- @BeforeClass
- @AfterClass
- @Test



Test Case

- Calculator
 - sum()
 - difference()
- CalculatorTest
 - testSum()
 - testDifference()

Files



- java.io.File
 - create new file
 - check if file exists
 - check if is directory
 - create temporary file
 - delete file
 - get full path
 - list files
 - read / write to file
 - others

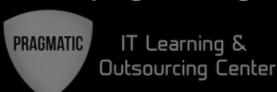




new File object

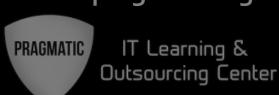
```
File file = new File("path_to_file");
file.exists();
file.getAbsolutePath();
file.createNewFile();
File.separator
file.listFiles()
```

Read from File



```
BufferedReader br = new BufferedReader(new
FileReader("file.txt"));
try {
    StringBuilder sb = new StringBuilder();
    String line = br.readLine();
    while (line != null) {
        sb.append(line);
        sb.append(System.lineSeparator());
        line = br.readLine();
    String everything = sb.toString();
 finally {
    br.close();
```

Read from File



Java 8 only

Write to File



```
public class Test {
            public static void main(String[] args) {
                         File file = new File("C:/temp/somefile.txt");
                         Writer output = null;
                         try {
                                     output = new BufferedWriter(new FileWriter(file));
                                     output.write("Something to write");
                         } catch (IOException e) {
                                     e.printStackTrace();
                         } finally {
                          if (output != null) {
                                      try {
                                                  output.close();
                                     } catch (IOException e) {
                                                  e.printStackTrace();
```

Write to File

