

# The Java Supplier Functional Interface: Case Study ex12

**Douglas C. Schmidt**

**[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)**

**[www.dre.vanderbilt.edu/~schmidt](http://www.dre.vanderbilt.edu/~schmidt)**

**Professor of Computer Science**

**Institute for Software  
Integrated Systems**

**Vanderbilt University  
Nashville, Tennessee, USA**



# Learning Objectives in this Part of the Lesson

---

- Know how a Java Supplier is used in conjunction with a Java Optional object

```
Map<String, String> beingMap = new  
    HashMap<String, String>() { {  
        put("Demon", "Naughty");  
        put("Angel", "Nice");  
    } };
```

```
String being = ...;
```

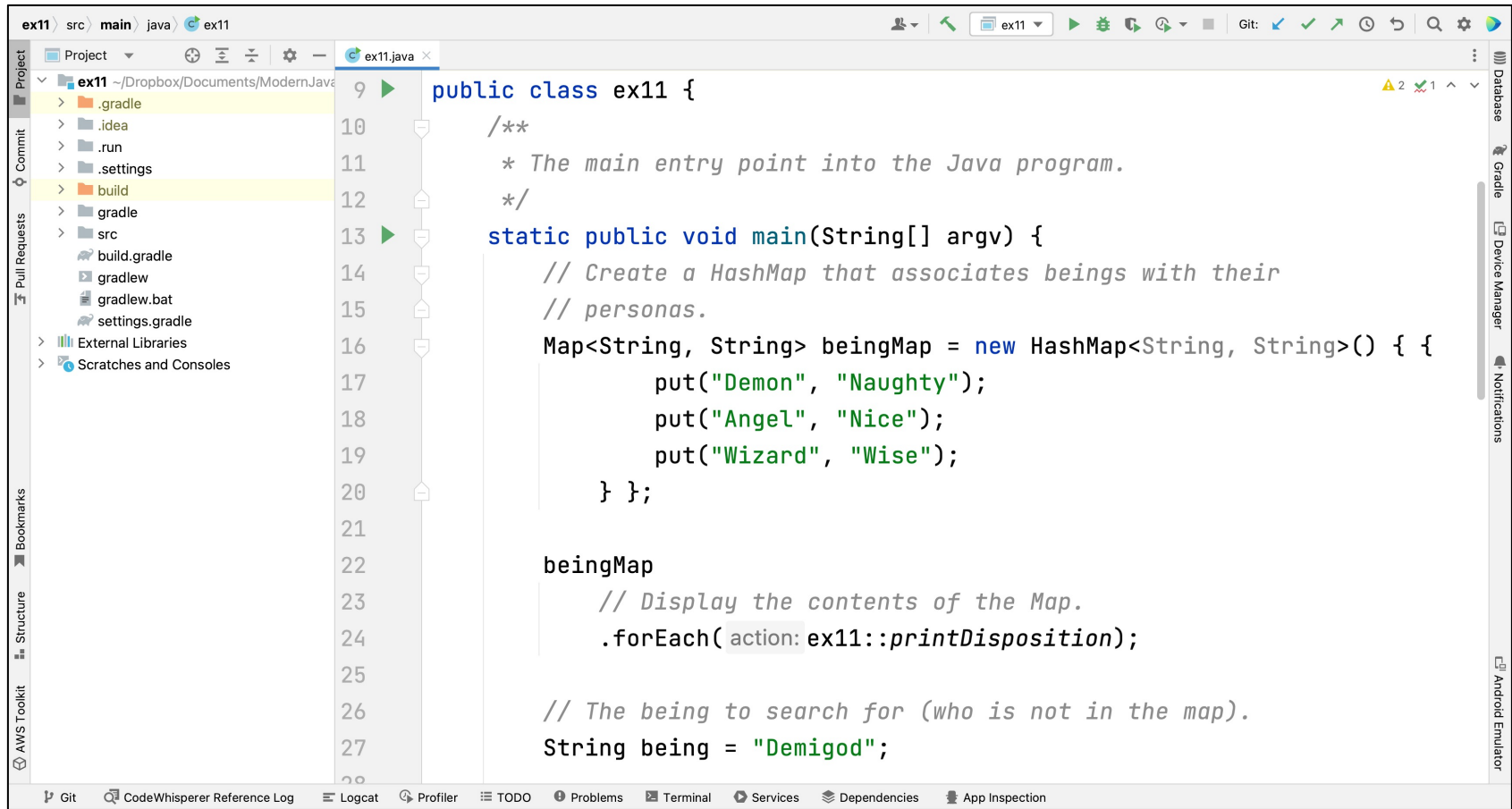
```
Optional<String> disposition =  
    Optional.ofNullable  
        (beingMap.get(being));
```

```
display(being, disposition  
        .orElseGet(() -> "unknown"));
```

---

# Applying the Java Supplier Functional Interface

# Applying the Java Supplier Functional Interface



```
9 public class ex11 {
10     /**
11      * The main entry point into the Java program.
12      */
13     static public void main(String[] argv) {
14         // Create a HashMap that associates beings with their
15         // personas.
16         Map<String, String> beingMap = new HashMap<String, String>() { {
17             put("Demon", "Naughty");
18             put("Angel", "Nice");
19             put("Wizard", "Wise");
20         } };
21
22         beingMap
23             // Display the contents of the Map.
24             .forEach( action: ex11::printDisposition);
25
26         // The being to search for (who is not in the map).
27         String being = "Demigod";
28     }
```

See [github.com/douglasraigschmidt/ModernJava/tree/main/FP/ex12](https://github.com/douglasraigschmidt/ModernJava/tree/main/FP/ex12)

---

# End of the Java Supplier Functional Interface: Case Study ex12