The Java BiFunction Functional Interface: Case Study ex11

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software Integrated Systems

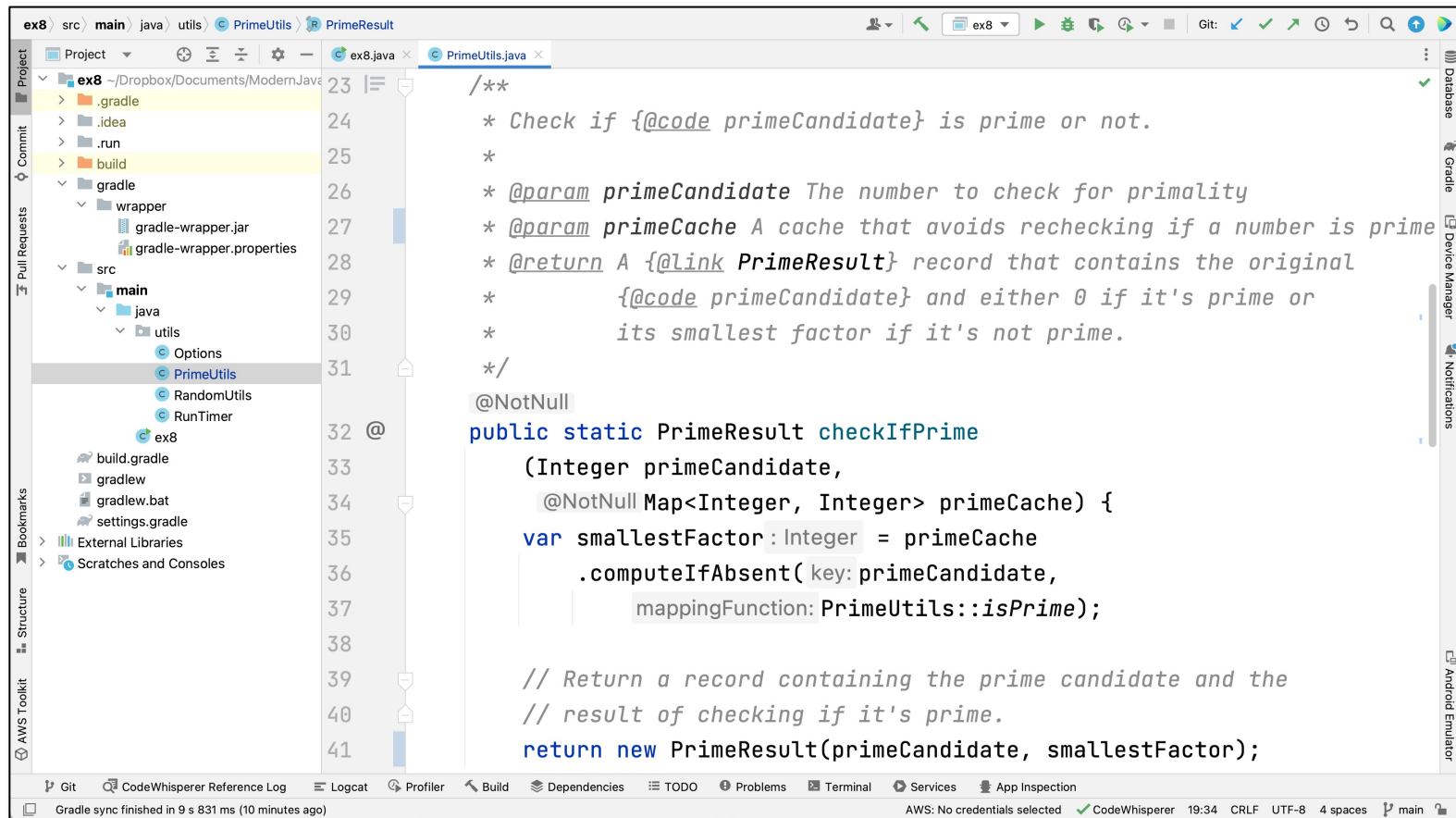**Vanderbilt University Nashville, Tennessee, USA**

Learning Objectives in this Part of the Lesson

- Know how the Java BiFunction functional interface can be used to replace all the values of all keys in a ConcurrentHashMap

```
var stoogeMap = makeMap();

System.out.println(stoogeMap);

for (Map.Entry<String, Integer>
        entry : stoogeMap.entrySet())
        entry.setValue
            (entry.getValue() - 30);

System.out.println(stoogeMap);

stoogeMap = makeMap();

stoogeMap.replaceAll
            ((k, v) -> v - 30);
```

See docs.oracle.com/javase/8/docs/api/java/util/function/BiFunction.html

Applying the Java BiFunction Functional Interface

Applying the Java BiFunction Functional Interface


```java
/**
 * Check if {@code primeCandidate} is prime or not.
 *
 * @param primeCandidate The number to check for primality
 * @param primeCache A cache that avoids rechecking if a number is prime
 * @return A {@link PrimeResult} record that contains the original
 *         {@code primeCandidate} and either 0 if it's prime or
 *         its smallest factor if it's not prime.
 */
@NotNull
public static PrimeResult checkIfPrime
    (Integer primeCandidate,
     @NotNull Map<Integer, Integer> primeCache) {
    var smallestFactor : Integer = primeCache
        .computeIfAbsent( key: primeCandidate,
            mappingFunction: PrimeUtils::isPrime);


    // Return a record containing the prime candidate and the
    // result of checking if it's prime.
    return new PrimeResult(primeCandidate, smallestFactor);
```

See github.com/douglascraigschmidt/ModernJava/tree/main/FP/ex11

End of the Java BiFunction Functional Interface: Case Study ex10