Code Quality

What is Clean Code?

"You know you are working on clean code when each routine you read turns out to be pretty much what you expected."
- Ward Cunningham

"Clean code can be read, and enhanced by a developer other than its original author."
- Dave Thomas

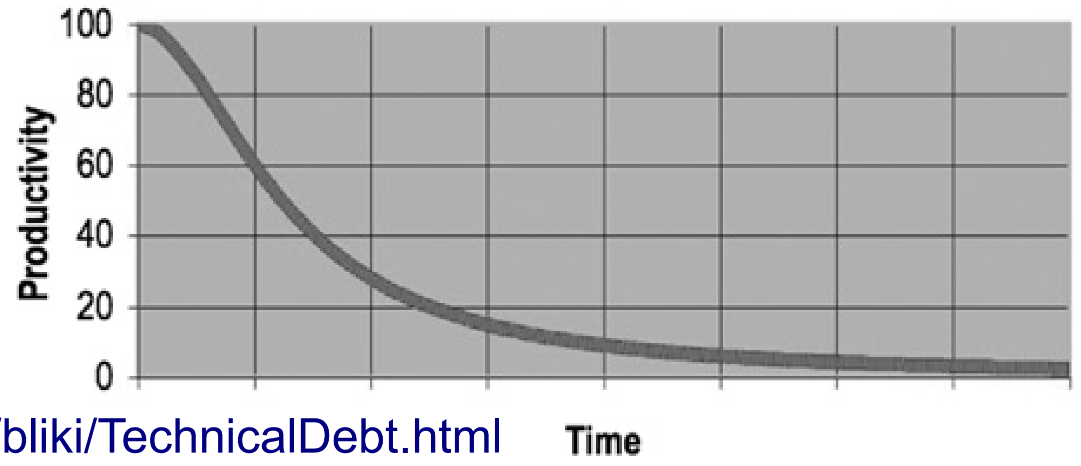
"Clean code always looks like it was written by someone who cares. There is nothing obvious that you can do to make it better."
- Michael Feathers

"Clean code is simple and direct. Clean code reads like well-written prose."
- Grady Booch

Clean Code ch1 / http://www.informit.com/articles/article.aspx?p=1235624

Technical Debt

"Doing things the quick and dirty way sets us up with a technical debt, which is similar to a financial debt. Like a financial debt, the technical debt incurs interest payments, which come in the form of the extra effort that we have to do in future development because of the quick and dirty design choice.
We can choose to continue paying the interest, or we can pay down the principal by refactoring the quick and dirty design into the better design. Although it costs to pay down the principal, we gain by reduced interest payments in the future."



http://martinfowler.com/bliki/TechnicalDebt.html
http://martinfowler.com/bliki/TechnicalDebtQuadrant.html
Clean Code ch1 / http://www.informit.com/articles/article.aspx?p=1235624
http://www.laputan.org/mud/

Refactoring

- "Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code yet improves its internal structure."
  -- Martin Fowler, *Refactoring*
- refactoring vs. **R**efactoring (a few seconds/minutes vs. days)
- DRY – Don't Repeat Yourself
- Code Smells
- Keeping the code at **top quality** is required for writing code using TDD. Otherwise bad code will grind it to a halt.
  (see SOLID principles: http://butunclebob.com/ArticleS.UncleBob.PrinciplesOfOod)
- A good regression test suite is needed. Run all tests after every change, to make sure that you broke nothing. Refactor only when all tests are green. Revert to last working state if you can't make the tests green – then try again with smaller steps.

http://www.refactoring.com/
http://c2.com/cgi/wiki?DontRepeatYourself
http://blog.objectmentor.com/articles/2007/07/20/whats-your-unit-of-measure
http://blog.objectmentor.com/articles/2008/07/21/tdd-is-how-i-do-it-not-what-i-do
http://randomactsofcoding.blogspot.com/2009/09/my-barriers-to-learning-tdd.html

Meaningful Names

- The name says what it is for, not what it is.
- Avoid encodings of any kind.
- Functions and variables have the same level of abstraction.
- Use pronounceable names.
- Shun names that disinform or confuse.
- Make context meaningful.
- Length of identifier matches scope.
- No lower-case L or upper-case o, ever.

```
int d; // elapsed time in days
```

$\longrightarrow$

```
int elapsedTimeInDays;
int daysSinceCreation;
int daysSinceModification;
int fileAgeInDays;
```

```
class DtaRcrd102 {
  private Date genymdhms;
  private Date modymdhms;
  private final String pszqint = "102";
...
```

$\longrightarrow$

```
class Customer {
  private Date generationTimestamp;
  private Date modificationTimestamp;
  private final String recordId = "102";
...
```

Clean Code ch2 / http://tottinge.blogsome.com/meaningfulnames/
http://agileinaflash.blogspot.com/2009/02/meaningful-names.html
http://butunclebob.com/ArticleS.TimOttinger.ApologizeIncode

Some Code Smells

- "In computer programming, code smell is any symptom in the source code of a program that possibly indicates a deeper problem."
  - Duplicated Code
  - Comments
  - Long Method
  - Large Class
  - Feature Envy
  - Divergent Change
  - Shotgun Surgery
  - Hard-to-Test Code
  - Fragile Test
  - Slow Tests

  etc.

http://www.codinghorror.com/blog/archives/000589.html
http://en.wikipedia.org/wiki/Code_smell
http://xunitpatterns.com/Test%20Smells.html

Demo: (Big) Refactoring Example

- Situation:

  An application server used *java.math.BigInteger* as the ID for entity objects which are stored in the object database. This is the Primitive Obsession code smell. It will need to be replaced with a new class (*EntityId*), in order to make its intent clearer and future refactorings easier.

- Codebase size:

  4400 SLOC production code

  7300 SLOC test code

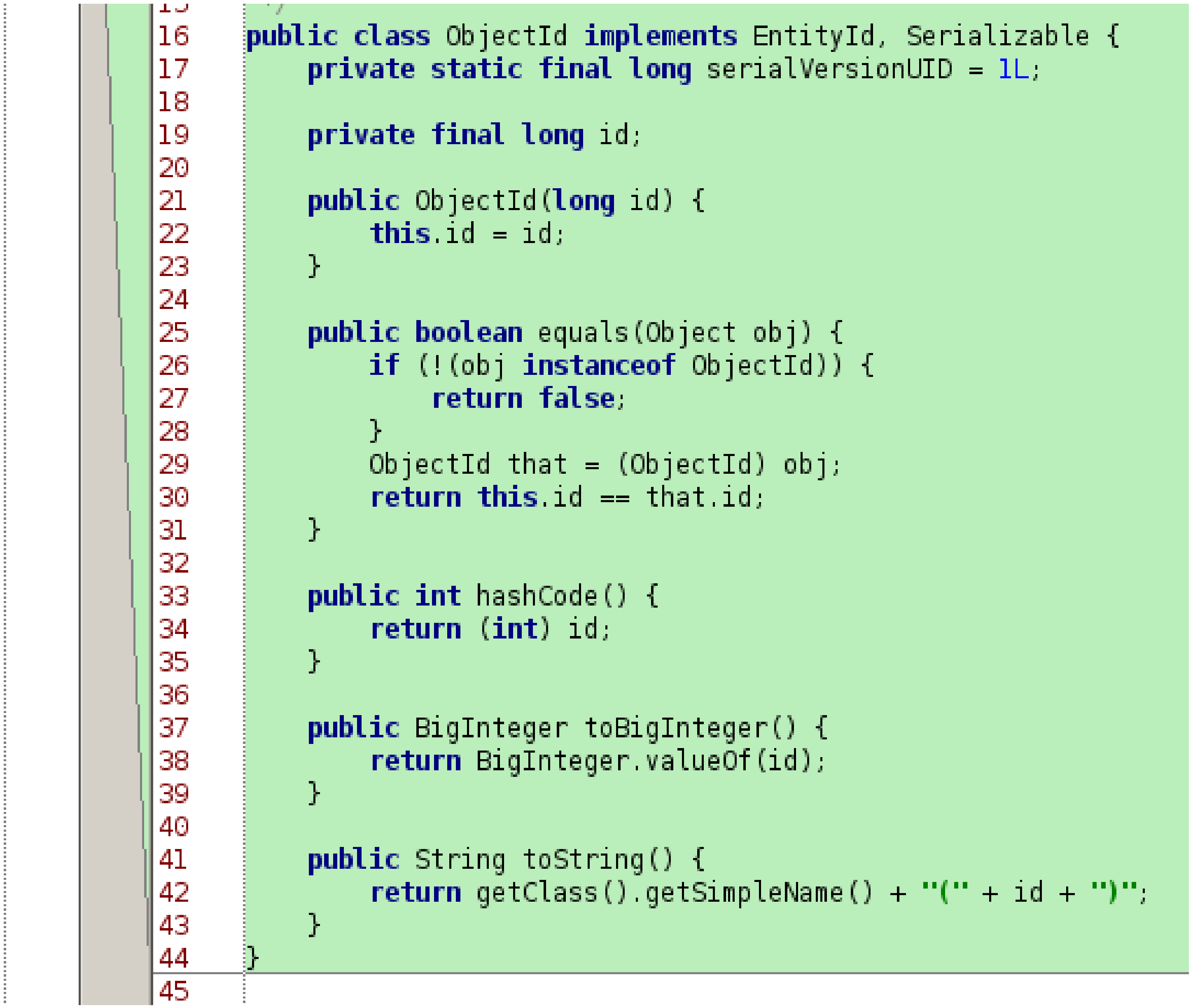
- Total time spent refactoring:

  2 hours

  (Demo: ~15 min, commit *entityid-refactor~3* onwards)

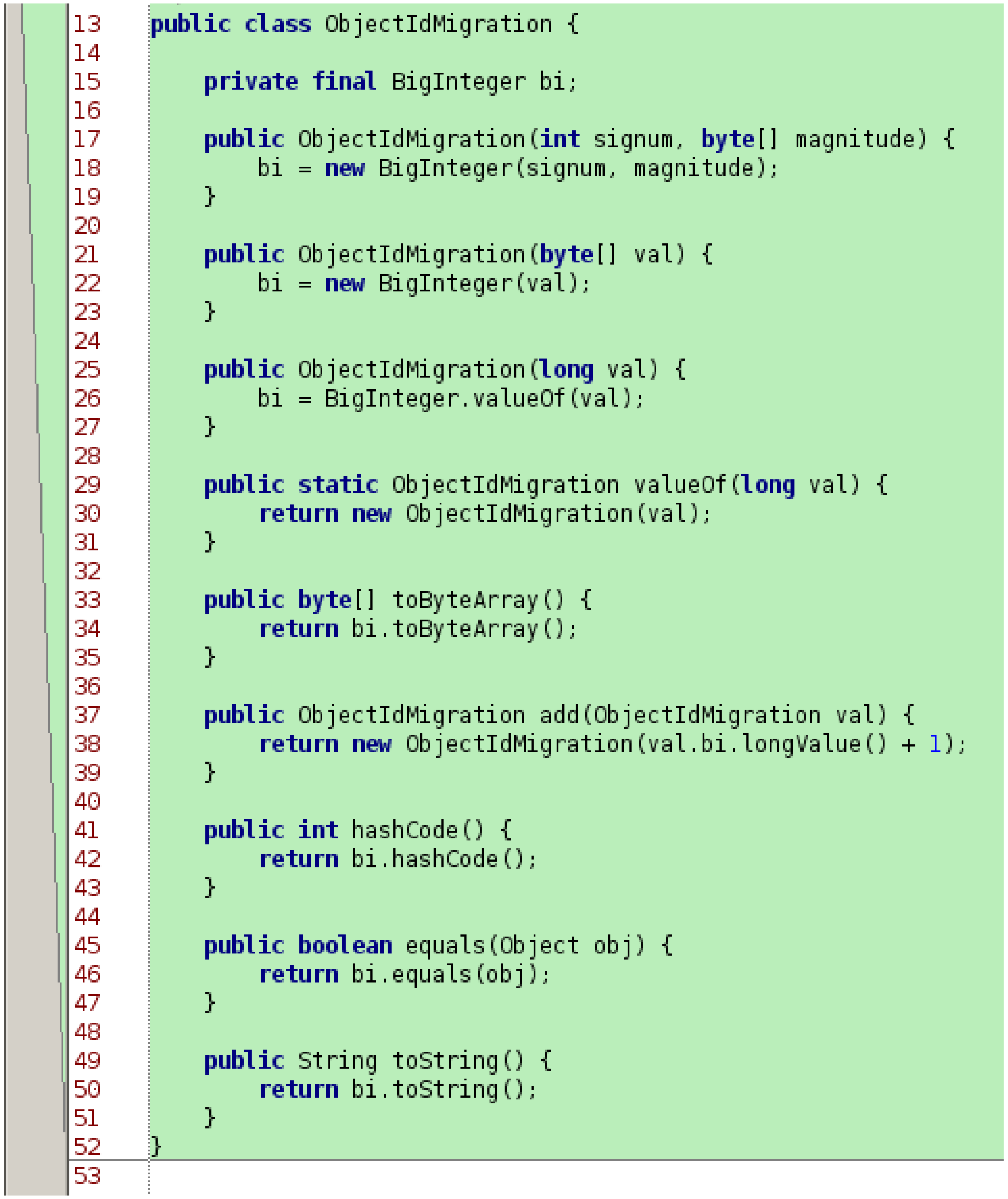
http://dimdwarf.sourceforge.net/ (commit *27ca4474* on 2009-08-13)
http://blogs.jetbrains.com/idea/2006/08/switching-between-api/

1. Create target class



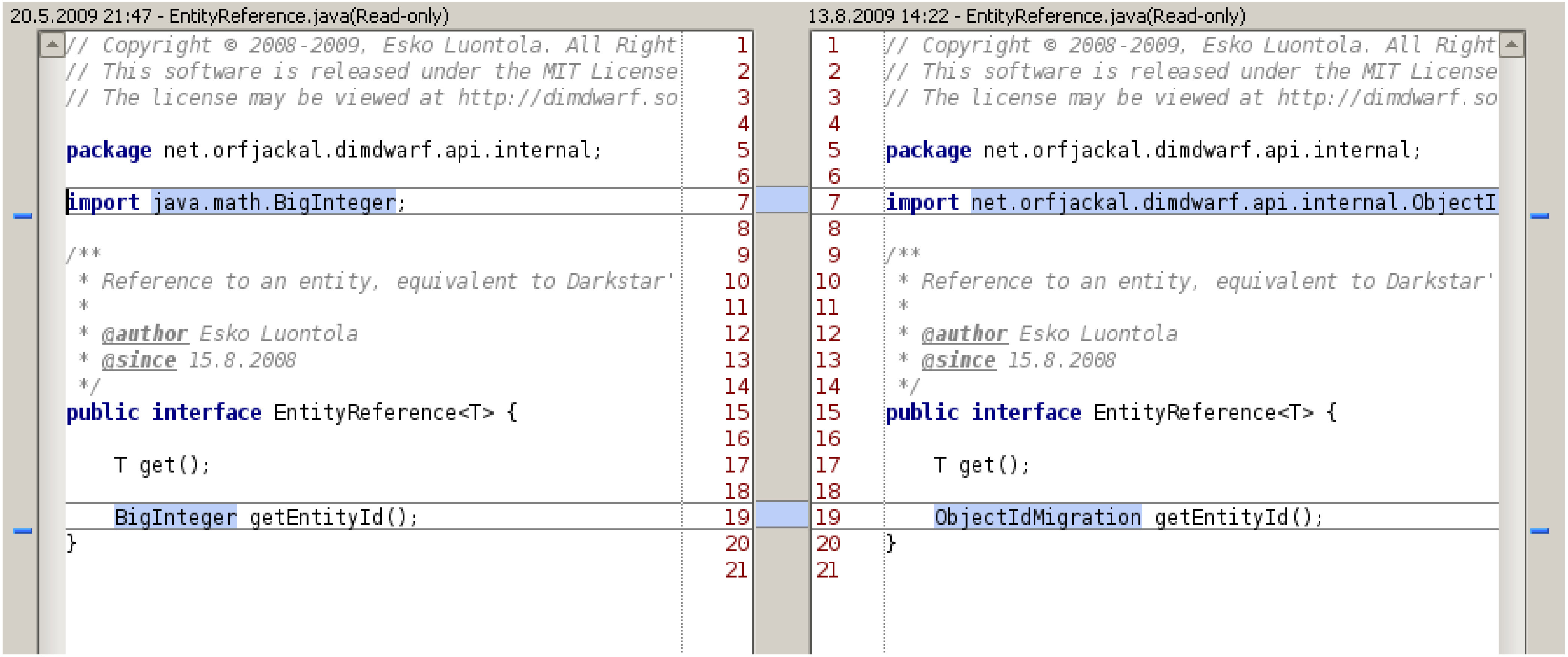
```
public class ObjectId implements EntityId, Serializable {
    private static final long serialVersionUID = 1L;

    private final long id;

    public ObjectId(long id) {
        this.id = id;
    }

    public boolean equals(Object obj) {
        if (!(obj instanceof ObjectId)) {
            return false;
        }
        ObjectId that = (ObjectId) obj;
        return this.id == that.id;
    }

    public int hashCode() {
        return (int) id;
    }

    public BigInteger toBigInteger() {
        return BigInteger.valueOf(id);
    }

    public String toString() {
        return getClass().getSimpleName() + "(" + id + ")";
    }
}
```

2. Create temporary adapter class



```java
public class ObjectIdMigration {

    private final BigInteger bi;

    public ObjectIdMigration(int signum, byte[] magnitude) {
        bi = new BigInteger(signum, magnitude);
    }

    public ObjectIdMigration(byte[] val) {
        bi = new BigInteger(val);
    }

    public ObjectIdMigration(long val) {
        bi = BigInteger.valueOf(val);
    }

    public static ObjectIdMigration valueOf(long val) {
        return new ObjectIdMigration(val);
    }

    public byte[] toByteArray() {
        return bi.toByteArray();
    }

    public ObjectIdMigration add(ObjectIdMigration val) {
        return new ObjectIdMigration(val.bi.longValue() + 1);
    }

    public int hashCode() {
        return bi.hashCode();
    }

    public boolean equals(Object obj) {
        return bi.equals(obj);
    }

    public String toString() {
        return bi.toString();
    }
}
```

3. Migrate* all usages of the original class to the adapter (hundreds of usages)


20.5.2009 21:47 - EntityReference.java(Read-only)

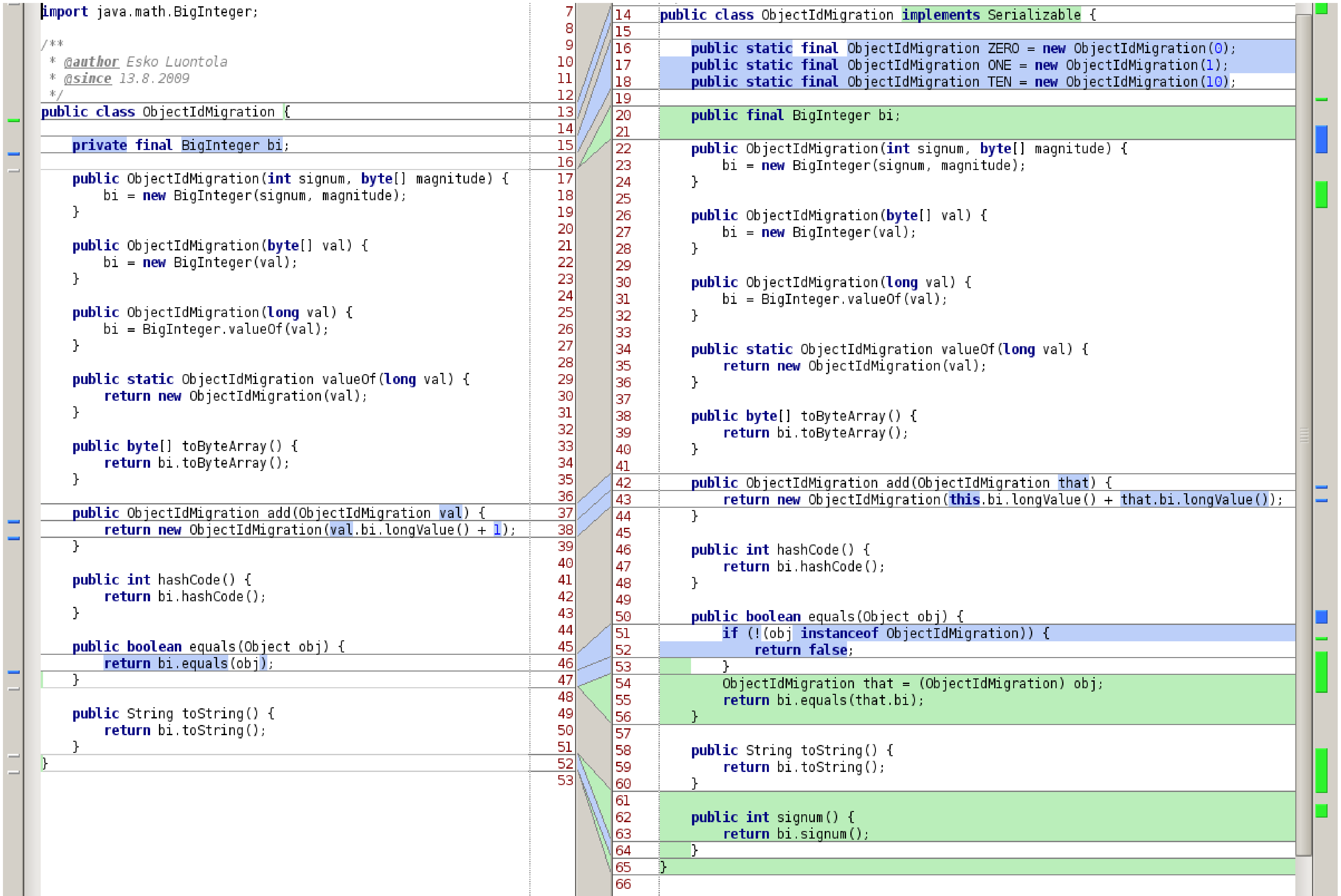
```
// Copyright © 2008-2009, Esko Luontola. All Right
// This software is released under the MIT License
// The license may be viewed at http://dimdwarf.so

package net.orfjackal.dimdwarf.api.internal;

import java.math.BigInteger;

/**
 * Reference to an entity, equivalent to Darkstar'
 *
 * @author Esko Luontola
 * @since 15.8.2008
 */
public interface EntityReference<T> {

    T get();

    BigInteger getEntityId();
}
```

13.8.2009 14:22 - EntityReference.java(Read-only)

```
// Copyright © 2008-2009, Esko Luontola. All Right
// This software is released under the MIT License
// The license may be viewed at http://dimdwarf.so

package net.orfjackal.dimdwarf.api.internal;

import net.orfjackal.dimdwarf.api.internal.ObjectI

/**
 * Reference to an entity, equivalent to Darkstar'
 *
 * @author Esko Luontola
 * @since 15.8.2008
 */
public interface EntityReference<T> {

    T get();

    ObjectIdMigration getEntityId();
}
```

* IDEA 8: *Refactor | Migrate*

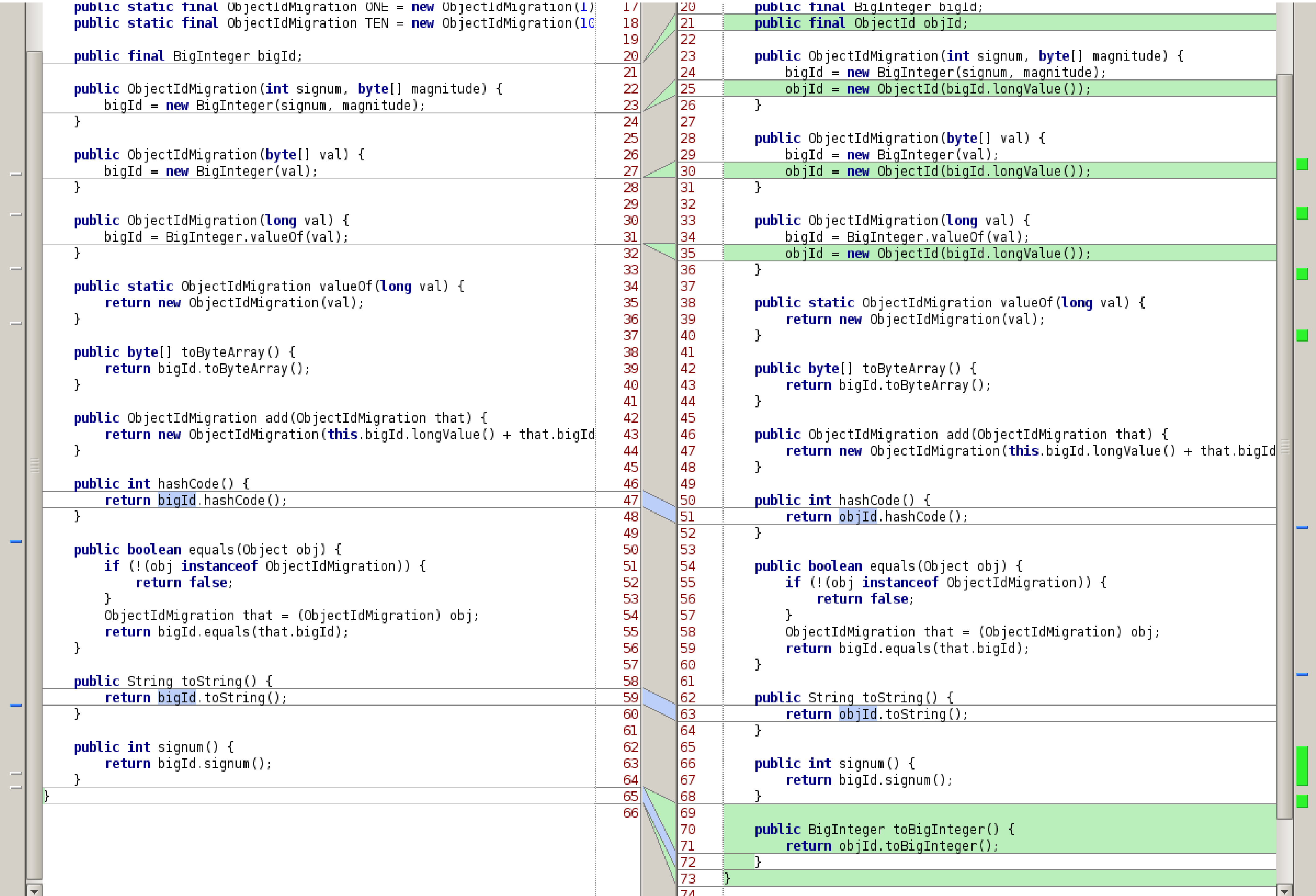
4. Add to the adapter class all methods and fields of the original class, until the project compiles and tests pass



```java
import java.math.BigInteger;

/**
 * @author Esko Luontola
 * @since 13.8.2009
 */
public class ObjectIdMigration {

    private final BigInteger bi;

    public ObjectIdMigration(int signum, byte[] magnitude) {
        bi = new BigInteger(signum, magnitude);
    }

    public ObjectIdMigration(byte[] val) {
        bi = new BigInteger(val);
    }

    public ObjectIdMigration(long val) {
        bi = BigInteger.valueOf(val);
    }

    public static ObjectIdMigration valueOf(long val) {
        return new ObjectIdMigration(val);
    }

    public byte[] toByteArray() {
        return bi.toByteArray();
    }

    public ObjectIdMigration add(ObjectIdMigration val) {
        return new ObjectIdMigration(val.bi.longValue() + 1);
    }

    public int hashCode() {
        return bi.hashCode();
    }

    public boolean equals(Object obj) {
        return bi.equals(obj);
    }

    public String toString() {
        return bi.toString();
    }
}
```

```java
public class ObjectIdMigration implements Serializable {

    public static final ObjectIdMigration ZERO = new ObjectIdMigration(0);
    public static final ObjectIdMigration ONE = new ObjectIdMigration(1);
    public static final ObjectIdMigration TEN = new ObjectIdMigration(10);

    public final BigInteger bi;

    public ObjectIdMigration(int signum, byte[] magnitude) {
        bi = new BigInteger(signum, magnitude);
    }

    public ObjectIdMigration(byte[] val) {
        bi = new BigInteger(val);
    }

    public ObjectIdMigration(long val) {
        bi = BigInteger.valueOf(val);
    }

    public static ObjectIdMigration valueOf(long val) {
        return new ObjectIdMigration(val);
    }

    public byte[] toByteArray() {
        return bi.toByteArray();
    }

    public ObjectIdMigration add(ObjectIdMigration that) {
        return new ObjectIdMigration(this.bi.longValue() + that.bi.longValue());
    }

    public int hashCode() {
        return bi.hashCode();
    }

    public boolean equals(Object obj) {
        if (!(obj instanceof ObjectIdMigration)) {
            return false;
        }
        ObjectIdMigration that = (ObjectIdMigration) obj;
        return bi.equals(that.bi);
    }

    public String toString() {
        return bi.toString();
    }

    public int signum() {
        return bi.signum();
    }
}
```

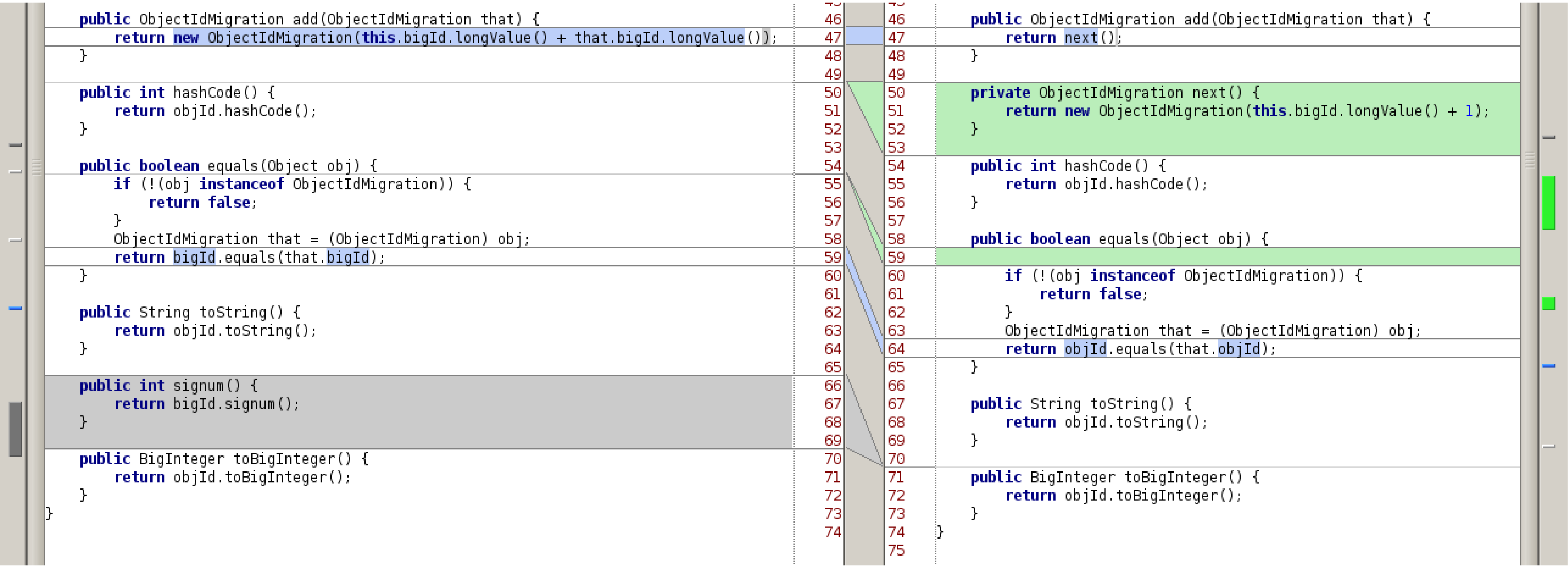
5. Put the target class next to the original class



```java
    public static final ObjectIdMigration ONE = new ObjectIdMigration(1)
    public static final ObjectIdMigration TEN = new ObjectIdMigration(10

    public final BigInteger bigId;

    public ObjectIdMigration(int signum, byte[] magnitude) {
        bigId = new BigInteger(signum, magnitude);
    }

    public ObjectIdMigration(byte[] val) {
        bigId = new BigInteger(val);
    }

    public ObjectIdMigration(long val) {
        bigId = BigInteger.valueOf(val);
    }

    public static ObjectIdMigration valueOf(long val) {
        return new ObjectIdMigration(val);
    }

    public byte[] toByteArray() {
        return bigId.toByteArray();
    }

    public ObjectIdMigration add(ObjectIdMigration that) {
        return new ObjectIdMigration(this.bigId.longValue() + that.bigId
    }

    public int hashCode() {
        return bigId.hashCode();
    }

    public boolean equals(Object obj) {
        if (!(obj instanceof ObjectIdMigration)) {
            return false;
        }
        ObjectIdMigration that = (ObjectIdMigration) obj;
        return bigId.equals(that.bigId);
    }

    public String toString() {
        return bigId.toString();
    }

    public int signum() {
        return bigId.signum();
    }
}
```

```java
    public final BigInteger bigId;
    public final ObjectId objId;

    public ObjectIdMigration(int signum, byte[] magnitude) {
        bigId = new BigInteger(signum, magnitude);
        objId = new ObjectId(bigId.longValue());
    }

    public ObjectIdMigration(byte[] val) {
        bigId = new BigInteger(val);
        objId = new ObjectId(bigId.longValue());
    }

    public ObjectIdMigration(long val) {
        bigId = BigInteger.valueOf(val);
        objId = new ObjectId(bigId.longValue());
    }

    public static ObjectIdMigration valueOf(long val) {
        return new ObjectIdMigration(val);
    }

    public byte[] toByteArray() {
        return bigId.toByteArray();
    }

    public ObjectIdMigration add(ObjectIdMigration that) {
        return new ObjectIdMigration(this.bigId.longValue() + that.bigId
    }

    public int hashCode() {
        return objId.hashCode();
    }

    public boolean equals(Object obj) {
        if (!(obj instanceof ObjectIdMigration)) {
            return false;
        }
        ObjectIdMigration that = (ObjectIdMigration) obj;
        return bigId.equals(that.bigId);
    }

    public String toString() {
        return objId.toString();
    }

    public int signum() {
        return bigId.signum();
    }

    public BigInteger toBigInteger() {
        return objId.toBigInteger();
    }
}
```

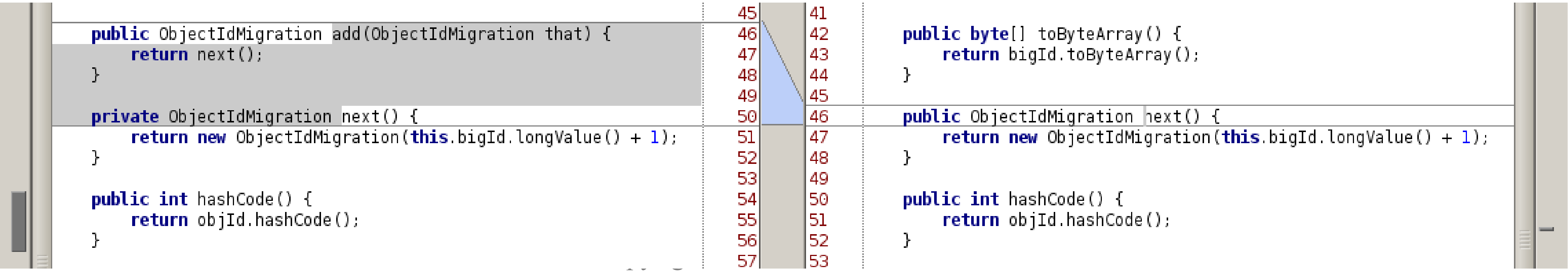
6. Delegate original method to target method



```java
    public ObjectIdMigration add(ObjectIdMigration that) {
        return new ObjectIdMigration(this.bigId.longValue() + that.bigId.longValue());
    }

    public int hashCode() {
        return objId.hashCode();
    }

    public boolean equals(Object obj) {
        if (!(obj instanceof ObjectIdMigration)) {
            return false;
        }
        ObjectIdMigration that = (ObjectIdMigration) obj;
        return bigId.equals(that.bigId);
    }

    public String toString() {
        return objId.toString();
    }

    public int signum() {
        return bigId.signum();
    }

    public BigInteger toBigInteger() {
        return objId.toBigInteger();
    }
}
```

```java
    public ObjectIdMigration add(ObjectIdMigration that) {
        return next();
    }

    private ObjectIdMigration next() {
        return new ObjectIdMigration(this.bigId.longValue() + 1);
    }

    public int hashCode() {
        return objId.hashCode();
    }

    public boolean equals(Object obj) {

        if (!(obj instanceof ObjectIdMigration)) {
            return false;
        }
        ObjectIdMigration that = (ObjectIdMigration) obj;
        return objId.equals(that.objId);
    }

    public String toString() {
        return objId.toString();
    }

    public BigInteger toBigInteger() {
        return objId.toBigInteger();
    }
}
```

7. Inline* original method

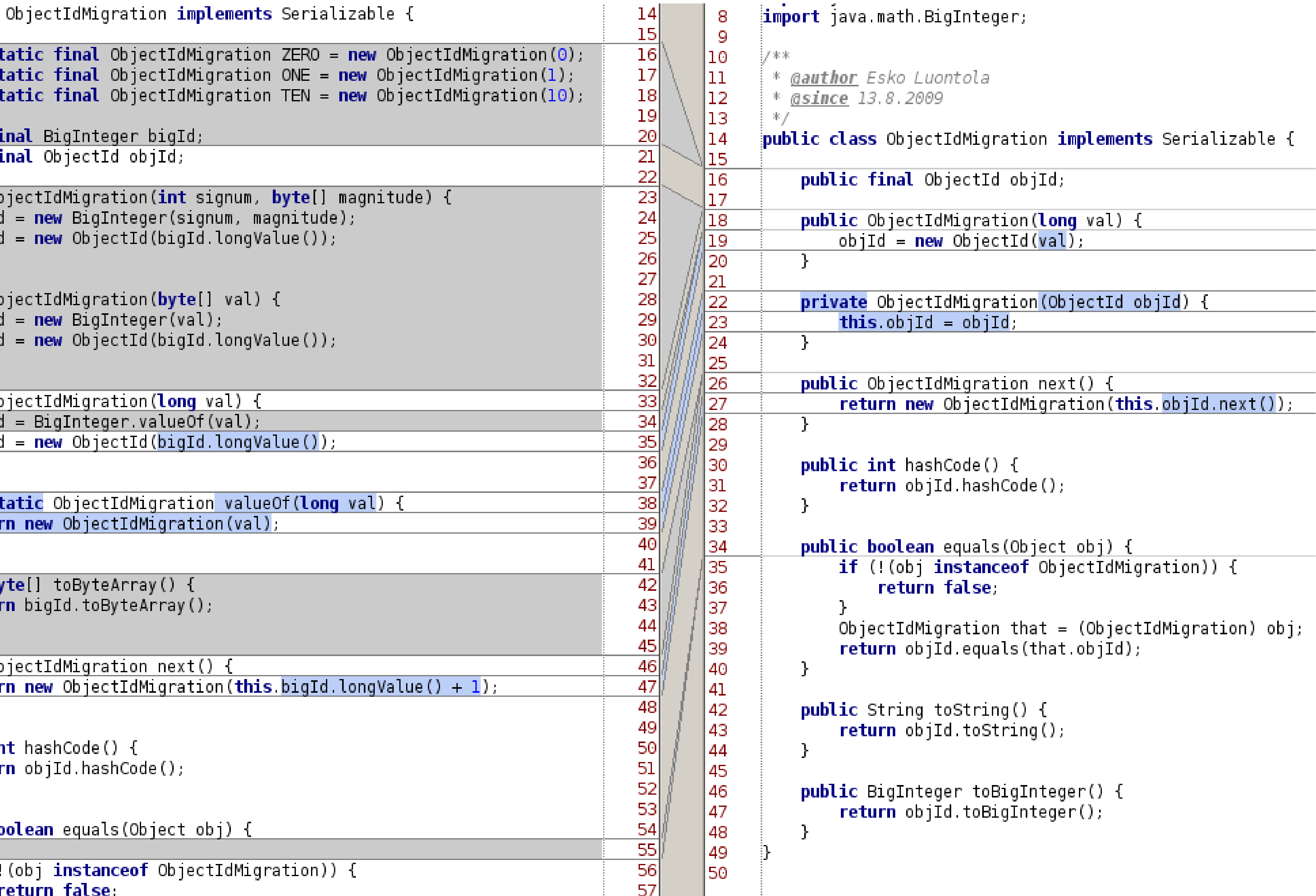


```java
    public ObjectIdMigration add(ObjectIdMigration that) {
        return next();
    }

    private ObjectIdMigration next() {
        return new ObjectIdMigration(this.bigId.longValue() + 1);
    }

    public int hashCode() {
        return objId.hashCode();
    }
```

```java
    public byte[] toByteArray() {
        return bigId.toByteArray();
    }

    public ObjectIdMigration next() {
        return new ObjectIdMigration(this.bigId.longValue() + 1);
    }

    public int hashCode() {
        return objId.hashCode();
    }
```

* IDEA 8: *Refactor* | *Inline*

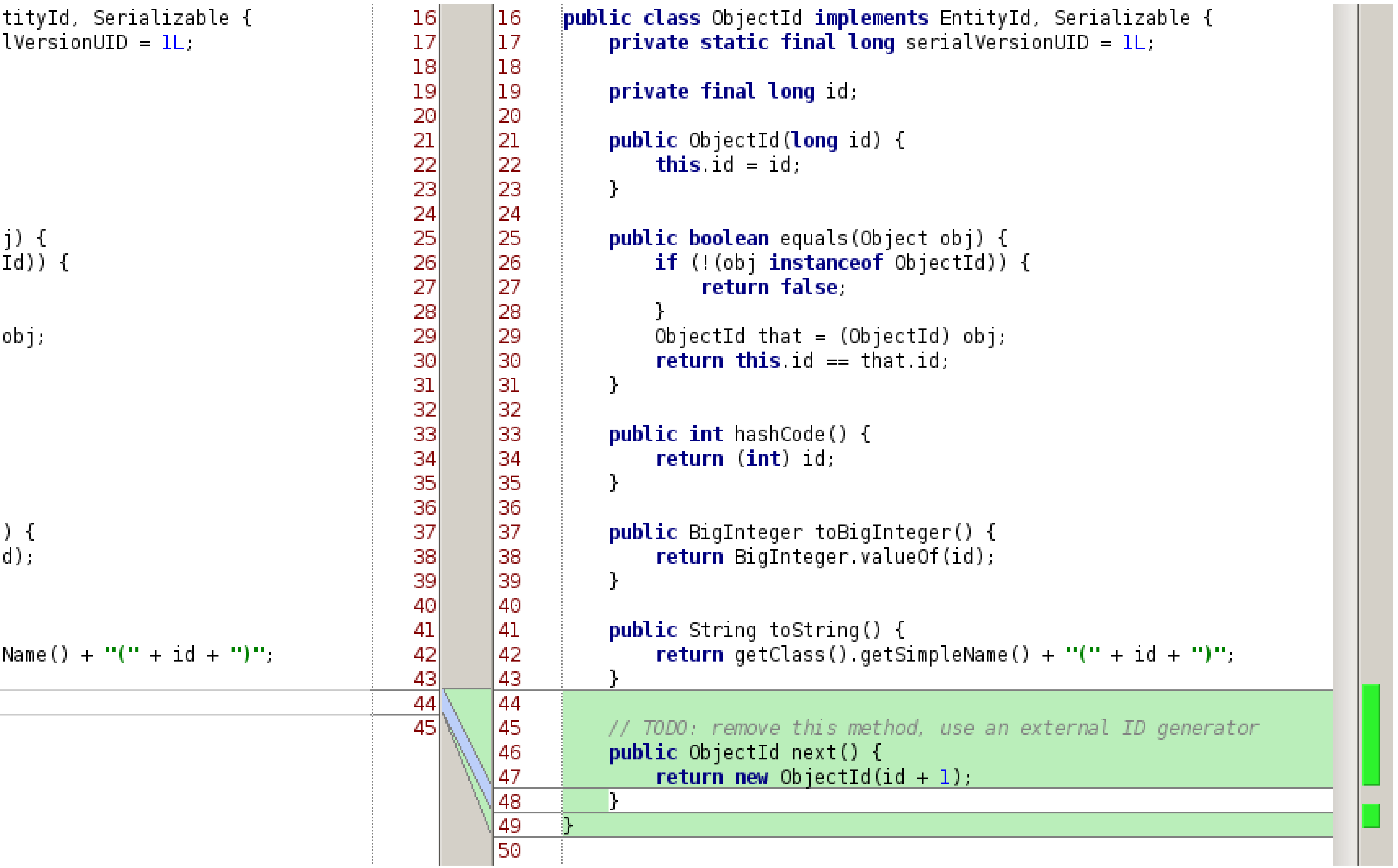
8.1 Repeat until the adapter and target classes have the same methods



```
 ObjectIdMigration implements Serializable {

tatic final ObjectIdMigration ZERO = new ObjectIdMigration(0);
tatic final ObjectIdMigration ONE = new ObjectIdMigration(1);
tatic final ObjectIdMigration TEN = new ObjectIdMigration(10);

inal BigInteger bigId;
inal ObjectId objId;

jectIdMigration(int signum, byte[] magnitude) {
d = new BigInteger(signum, magnitude);
d = new ObjectId(bigId.longValue());


jectIdMigration(byte[] val) {
d = new BigInteger(val);
d = new ObjectId(bigId.longValue());


jectIdMigration(long val) {
d = BigInteger.valueOf(val);
d = new ObjectId(bigId.longValue());


tatic ObjectIdMigration valueOf(long val) {
rn new ObjectIdMigration(val);


yte[] toByteArray() {
rn bigId.toByteArray();


jectIdMigration next() {
rn new ObjectIdMigration(this.bigId.longValue() + 1);


nt hashCode() {
rn objId.hashCode();


oolean equals(Object obj) {

!(obj instanceof ObjectIdMigration)) {
return false;
```

```
import java.math.BigInteger;

/**
 * @author Esko Luontola
 * @since 13.8.2009
 */
public class ObjectIdMigration implements Serializable {

    public final ObjectId objId;

    public ObjectIdMigration(long val) {
        objId = new ObjectId(val);
    }

    private ObjectIdMigration(ObjectId objId) {
        this.objId = objId;
    }

    public ObjectIdMigration next() {
        return new ObjectIdMigration(this.objId.next());
    }

    public int hashCode() {
        return objId.hashCode();
    }

    public boolean equals(Object obj) {
        if (!(obj instanceof ObjectIdMigration)) {
            return false;
        }
        ObjectIdMigration that = (ObjectIdMigration) obj;
        return objId.equals(that.objId);
    }

    public String toString() {
        return objId.toString();
    }

    public BigInteger toBigInteger() {
        return objId.toBigInteger();
    }
}
```

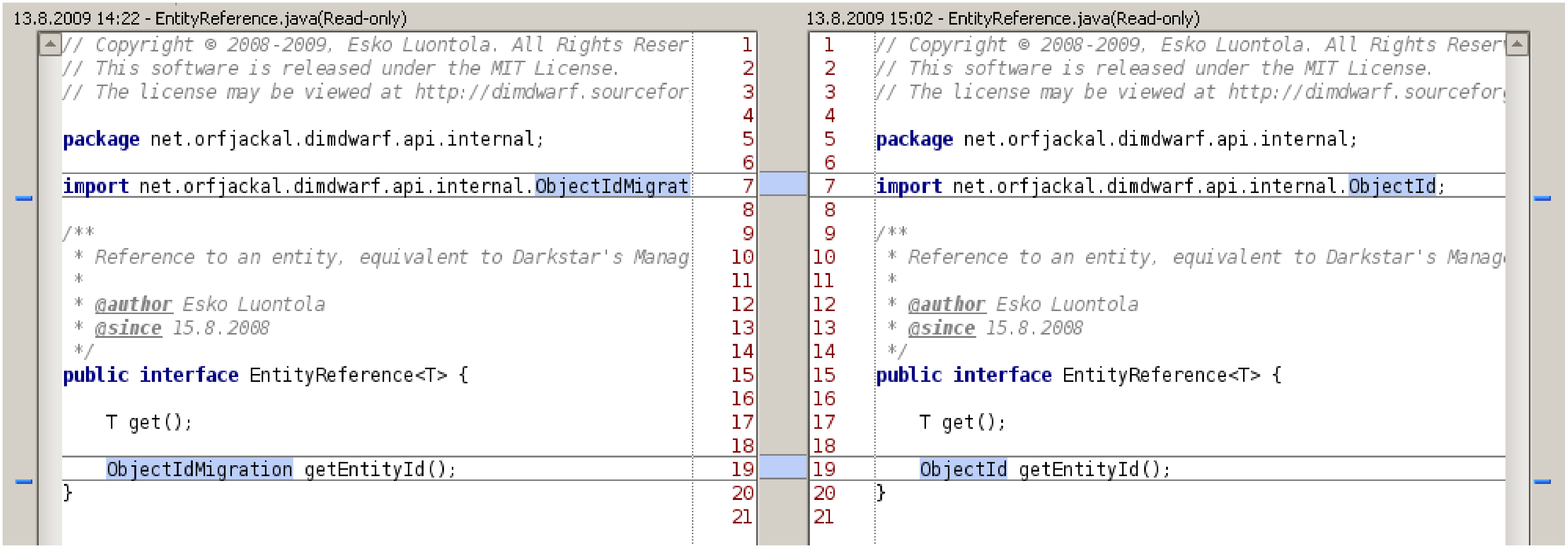
8.2 Repeat until the adapter and target classes have the same methods



```
tityId, Serializable {
lVersionUID = 1L;







j) {
Id)) {


obj;







) {
d);




Name() + "(" + id + ")";
```

```java
public class ObjectId implements EntityId, Serializable {
    private static final long serialVersionUID = 1L;

    private final long id;

    public ObjectId(long id) {
        this.id = id;
    }

    public boolean equals(Object obj) {
        if (!(obj instanceof ObjectId)) {
            return false;
        }
        ObjectId that = (ObjectId) obj;
        return this.id == that.id;
    }

    public int hashCode() {
        return (int) id;
    }

    public BigInteger toBigInteger() {
        return BigInteger.valueOf(id);
    }

    public String toString() {
        return getClass().getSimpleName() + "(" + id + ")";
    }

    // TODO: remove this method, use an external ID generator
    public ObjectId next() {
        return new ObjectId(id + 1);
    }
}

```

9. Migrate* all usages of the adapter class to the target (hundreds of usages)



13.8.2009 14:22 - EntityReference.java(Read-only)

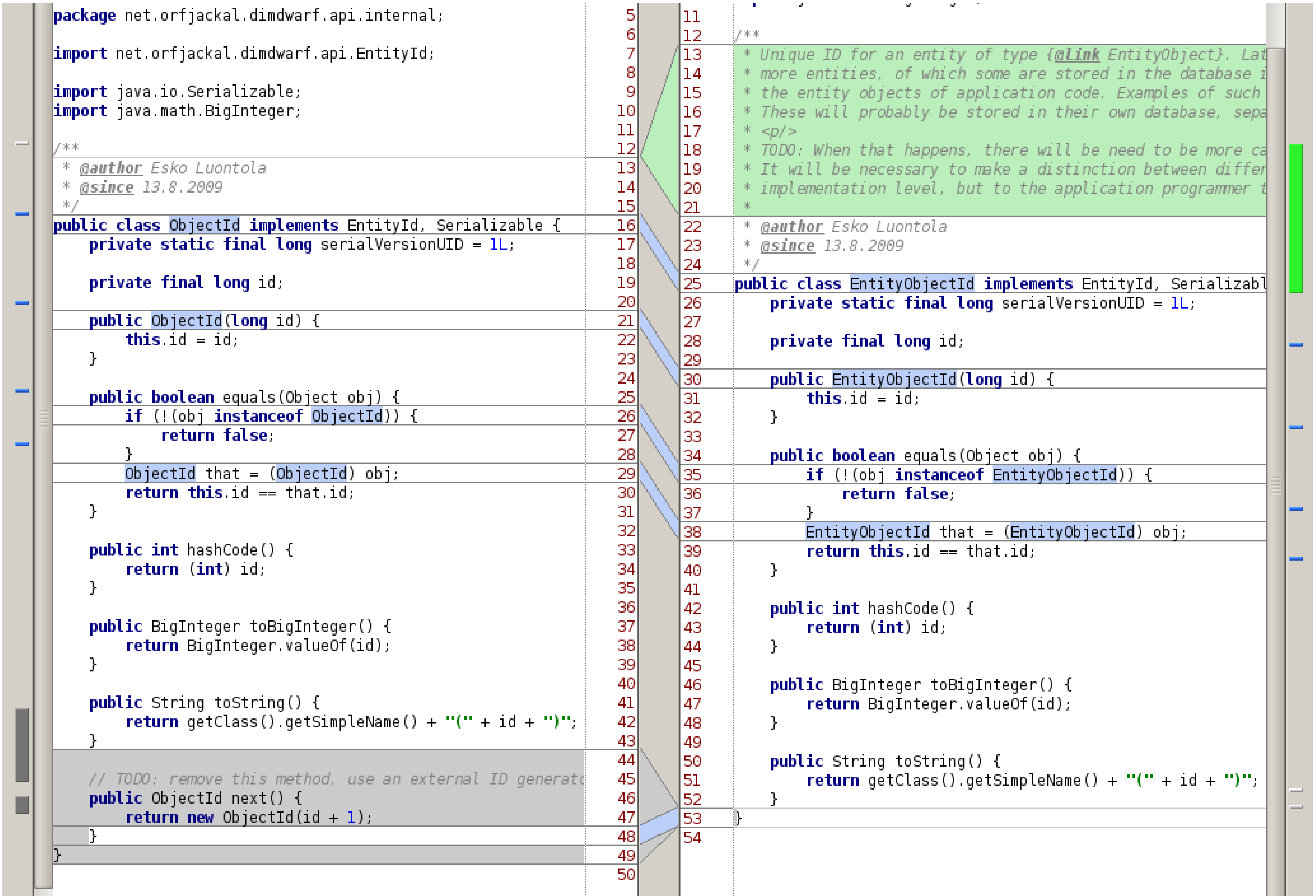
```
// Copyright © 2008-2009, Esko Luontola. All Rights Reser
// This software is released under the MIT License.
// The license may be viewed at http://dimdwarf.sourcefor

package net.orfjackal.dimdwarf.api.internal;

import net.orfjackal.dimdwarf.api.internal.ObjectIdMigrat

/**
 * Reference to an entity, equivalent to Darkstar's Manag
 *
 * @author Esko Luontola
 * @since 15.8.2008
 */
public interface EntityReference<T> {

    T get();

    ObjectIdMigration getEntityId();
}
```

13.8.2009 15:02 - EntityReference.java(Read-only)

```
// Copyright © 2008-2009, Esko Luontola. All Rights Reser
// This software is released under the MIT License.
// The license may be viewed at http://dimdwarf.sourcefor

package net.orfjackal.dimdwarf.api.internal;

import net.orfjackal.dimdwarf.api.internal.ObjectId;

/**
 * Reference to an entity, equivalent to Darkstar's Manag
 *
 * @author Esko Luontola
 * @since 15.8.2008
 */
public interface EntityReference<T> {

    T get();

    ObjectId getEntityId();
}
```

* IDEA 8: *Refactor* | *Migrate*

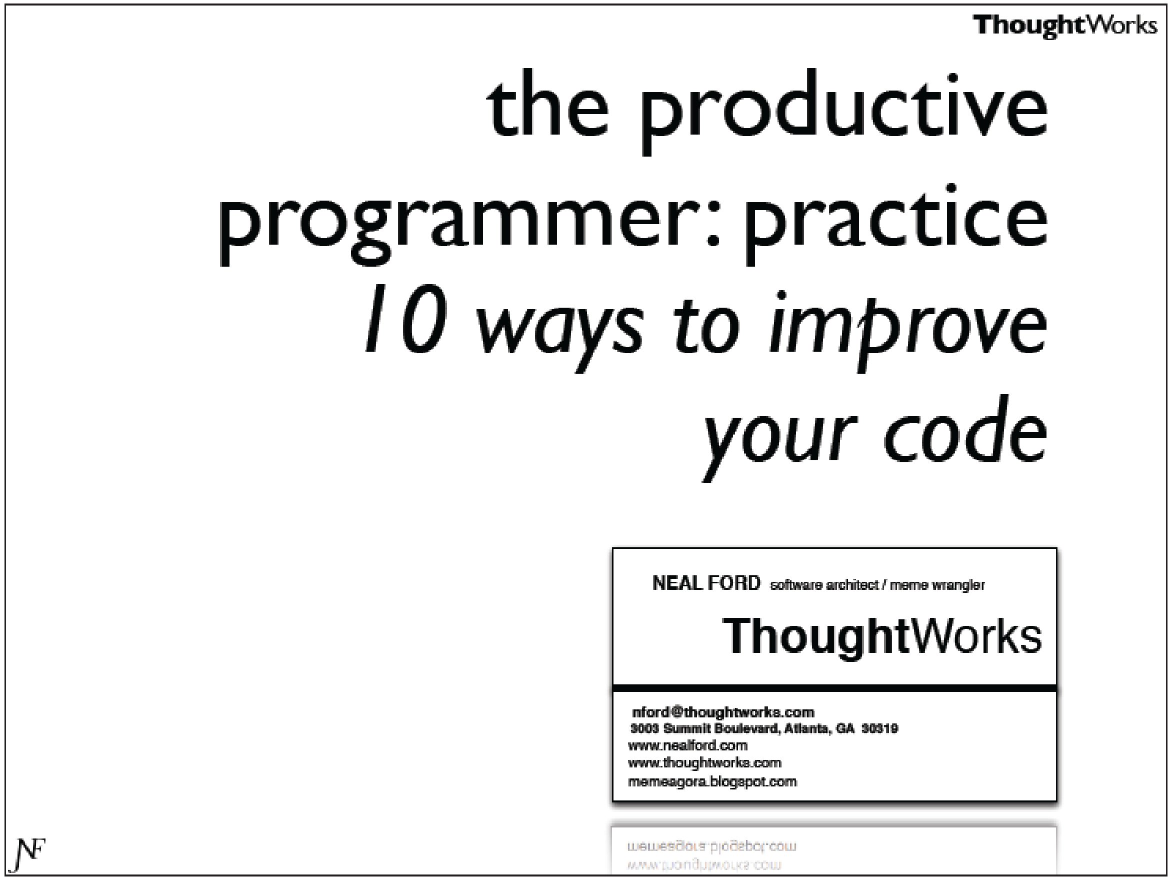
10. Delete the adapter class and do final cleanups



```
package net.orfjackal.dimdwarf.api.internal;

import net.orfjackal.dimdwarf.api.EntityId;

import java.io.Serializable;
import java.math.BigInteger;

/**
 * @author Esko Luontola
 * @since 13.8.2009
 */
public class ObjectId implements EntityId, Serializable {
    private static final long serialVersionUID = 1L;

    private final long id;

    public ObjectId(long id) {
        this.id = id;
    }

    public boolean equals(Object obj) {
        if (!(obj instanceof ObjectId)) {
            return false;
        }
        ObjectId that = (ObjectId) obj;
        return this.id == that.id;
    }

    public int hashCode() {
        return (int) id;
    }

    public BigInteger toBigInteger() {
        return BigInteger.valueOf(id);
    }

    public String toString() {
        return getClass().getSimpleName() + "(" + id + ")";
    }

    // TODO: remove this method, use an external ID generat
    public ObjectId next() {
        return new ObjectId(id + 1);
    }
}
```

```
/**
 * Unique ID for an entity of type {@link EntityObject}. Lat
 * more entities, of which some are stored in the database i
 * the entity objects of application code. Examples of such
 * These will probably be stored in their own database, sepa
 * <p/>
 * TODO: When that happens, there will be need to be more ca
 * It will be necessary to make a distinction between differ
 * implementation level, but to the application programmer t
 *
 * @author Esko Luontola
 * @since 13.8.2009
 */
public class EntityObjectId implements EntityId, Serializabl
    private static final long serialVersionUID = 1L;

    private final long id;

    public EntityObjectId(long id) {
        this.id = id;
    }

    public boolean equals(Object obj) {
        if (!(obj instanceof EntityObjectId)) {
            return false;
        }
        EntityObjectId that = (EntityObjectId) obj;
        return this.id == that.id;
    }

    public int hashCode() {
        return (int) id;
    }

    public BigInteger toBigInteger() {
        return BigInteger.valueOf(id);
    }

    public String toString() {
        return getClass().getSimpleName() + "(" + id + ")";
    }
}
```

10 Ways to Improve Your Code


ThoughtWorks

the productive programmer: practice

10 ways to improve your code

NEAL FORD software architect / meme wrangler

ThoughtWorks

nford@thoughtworks.com
3003 Summit Boulevard, Atlanta, GA 30319
www.nealford.com
www.thoughtworks.com
memeagora.blogspot.com

NF

http://www.infoq.com/presentations/10-Ways-to-Better-Code-Neal-Ford (59 min)
http://qconsf.com/sf2008/file?path=/qcon-sanfran-2008/slides//NealFord_10_Ways_to_Improve_Your_Code.pdf

Summary of *10 Ways to...*

1. Composed Method
   - "Divide your program into methods that perform one identifiable task. Keep all of the operations in a method at the same level of abstraction. This will naturally result in programs with many small methods, each a few lines long."
   - Benefits:
     - shorter methods easier to test
     - method names become documentation
     - large number of very cohesive methods
     - discover reusable assets that you didn't know were there

Summary of *10 Ways to...*

2. Test-Driven *Design*
   - Benefits:
     - first consumer
     - think about how the rest of the world uses this class
     - creates consumption awareness
     - forces mocking of dependent objects
     - naturally creates composed method
     - cleaner metrics (e.g. cyclometric complexity)

Summary of *10 Ways to...*

3. Static Analysis
4. Good Citizenship
   - "How classes react to one another in a civilized society."
   - Example: static methods
     - good: stateless utility methods
     - bad: mixing state + static (e.g. the evil singleton pattern)
5. YAGNI – You Ain't Gonna Need It
   - Discourages gold plating:
     - build the simplest thing that we need right now
     - don't indulge in speculative development
       - increases software entropy
       - only saves time if you can guarantee you won't have to change it later
       - leads to (the evil version of) frameworks

Summary of *10 Ways to...*

6. Question Authority
   - angry monkeys: test names (camel case hard to read)
   - non-intuitive: pair programming (15% slower, 15% fewer defects; real-time code review)
7. SLAP - Single Level of Abstraction Principle
   - "Keep all lines of code in a method at the same level of abstraction."
     - jumping abstraction layers makes code hard to understand
     - composed method → SLAP
     - refactor to slap, even if it means single-line methods
8. Polyglot Programming
   - "Leveraging existing *platforms* with *languages* targeted at specific problems and applications."

Summary of *10 Ways to...*

9. Every Nuance
   - "If you are going to spend time primarily in a language, it makes sense to learn all the nuances of that language – all the little back alleys and strange little places where you hardly ever go."
10. Anti-Objects
    - "The metaphor of objects can go too far by making us try to create objects that are too much inspired by the real world."
    - "Take something that appears to be the opposite of what you think you should be writing code about, and see if that yields a simpler solution to your problem."
      - "Pacman smell": intelligence not in the ghost, but in the maze

Course Material

- *Clean Code*, chapter 1: Clean Code
  = http://www.informit.com/articles/article.aspx?p=1235624
- *Clean Code*, chapter 2: Meaningful Names
  ≈ http://tottinge.blogsome.com/meaningfulnames/
  ≈ http://agileinaflash.blogspot.com/2009/02/meaningful-names.html
- http://martinfowler.com/bliki/TechnicalDebt.html
- http://c2.com/cgi/wiki?DontRepeatYourself
- http://butunclebob.com/ArticleS.TimOttinger.ApologizeIncode
- http://www.codinghorror.com/blog/archives/000589.html
- http://xunitpatterns.com/Test%20Smells.html
- http://www.infoq.com/presentations/10-Ways-to-Better-Code-Neal-Ford