COMMENTS

Java clean code

Outline

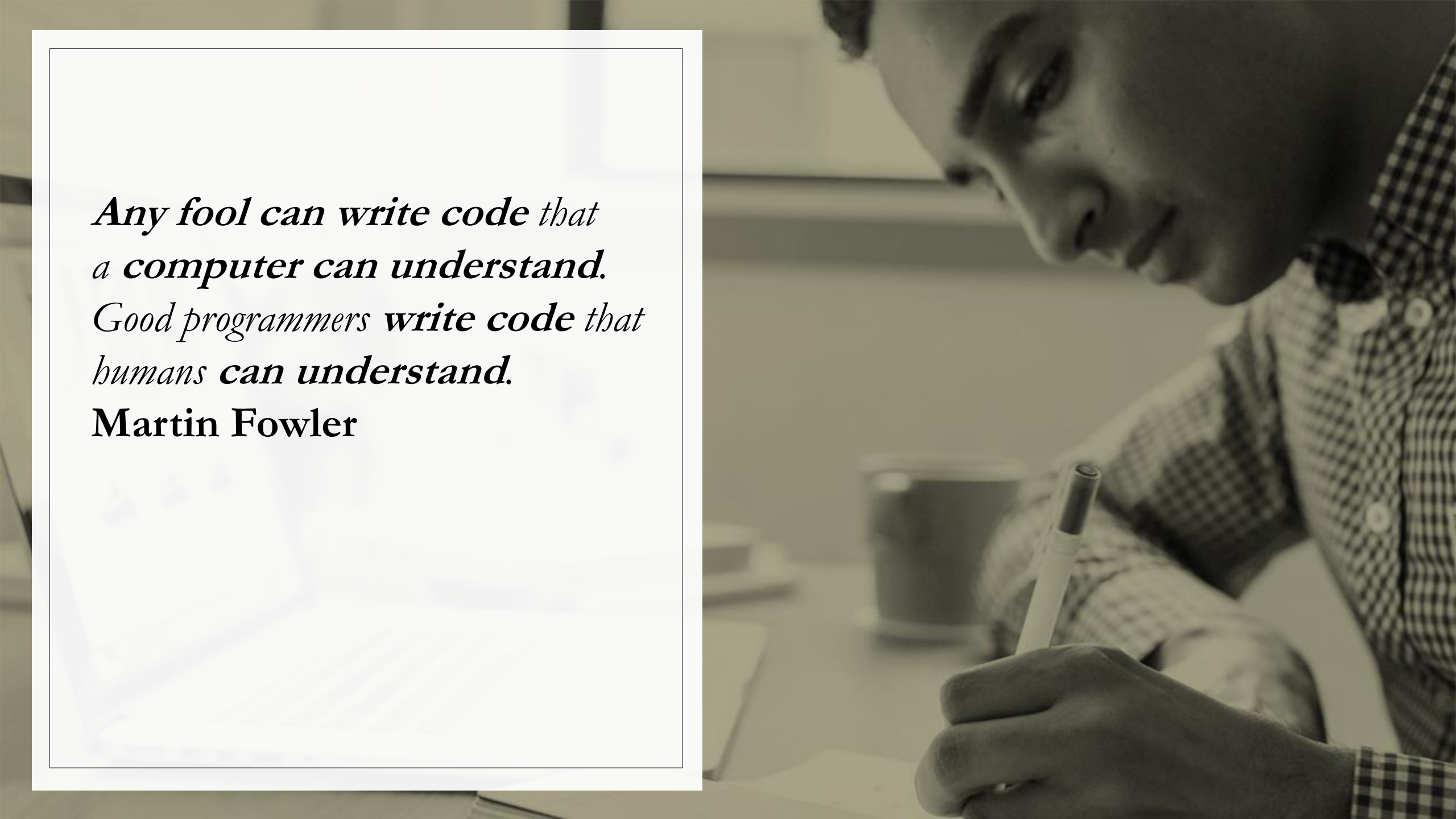
Lesson 1.
Clean code

Lesson 2.
Names

Lesson 3.
Methods

Lesson 4.
Classes

Lesson 5.
Comments



Any fool can write code *that a* ***computer can understand.*** *Good programmers* ***write code*** *that humans* ***can understand.***

Martin Fowler

To comment or not to comment?

Two types of programmers

- No comments
- Comment everything
- Find the balance

DRY principle

- Do not repeat yourself
- Do not copy and paste
- Multiple code similarity
- Do not write the same code repeatedly.

DRY bad example

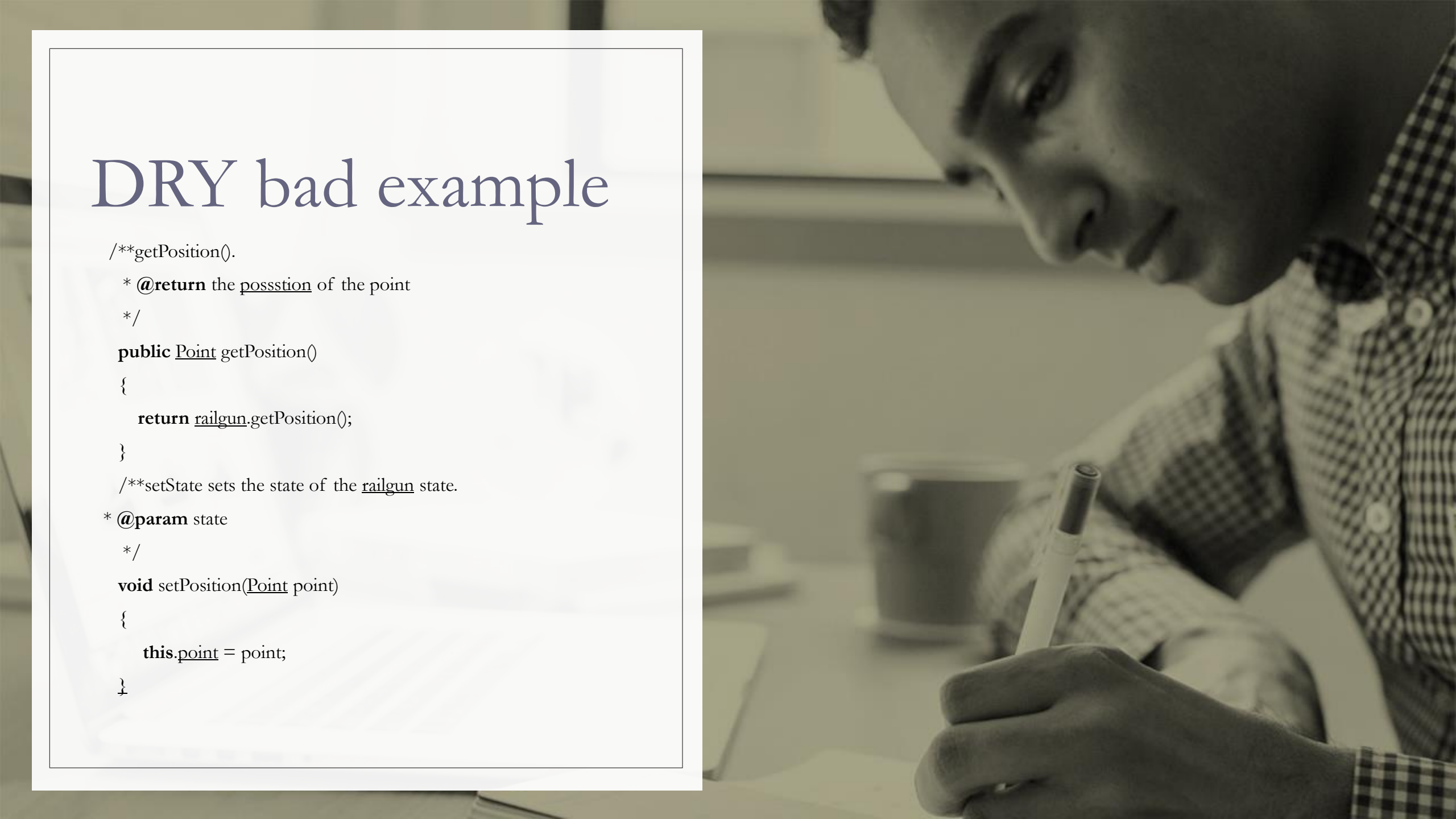
```
/**DamagedState is a damaged state.
*/
public class DamagedState implements RailgunState
{
   /**
    * railgun object of type Railgun.
    */
   Railgun railgun  ;
   /**
    * MAX_AMMO max ammonitions.
    */
   static int MAX_AMMO = 11;
```

DRY bad example

```
/**result what the result is.
 */
int result = 6;
/**
 * state object of type Railgun.
 */
RailgunState state ;
/**DamagedState default constructor.
 */
DamagedState()
{
}
```

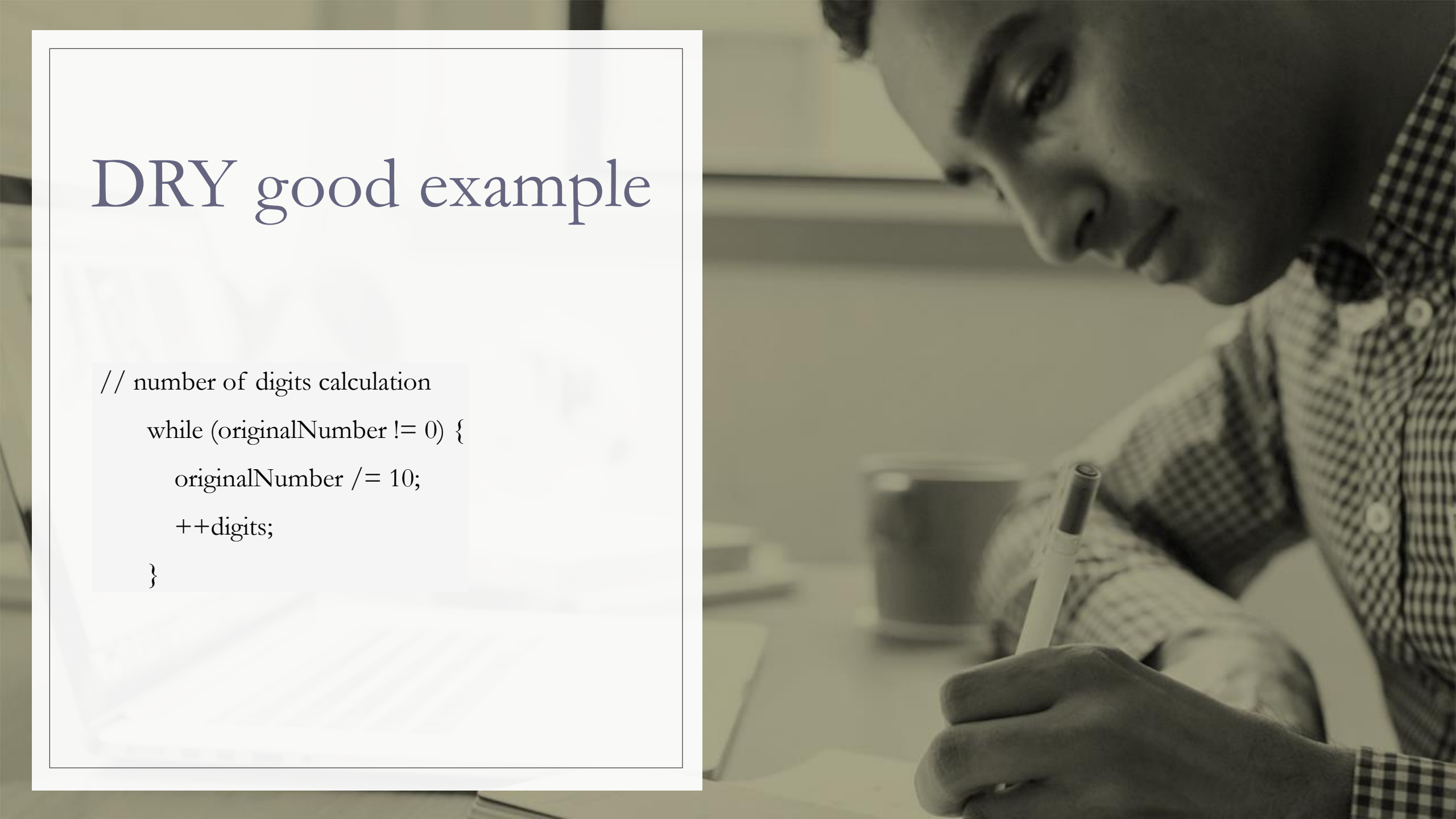
DRY bad example

```
/**getPosition().
 * @return the possstion of the point
 */
public Point getPosition()
{
    return railgun.getPosition();
}
/**setState sets the state of the railgun state.
* @param state
 */
void setPosition(Point point)
{
    this.point = point;
}
```



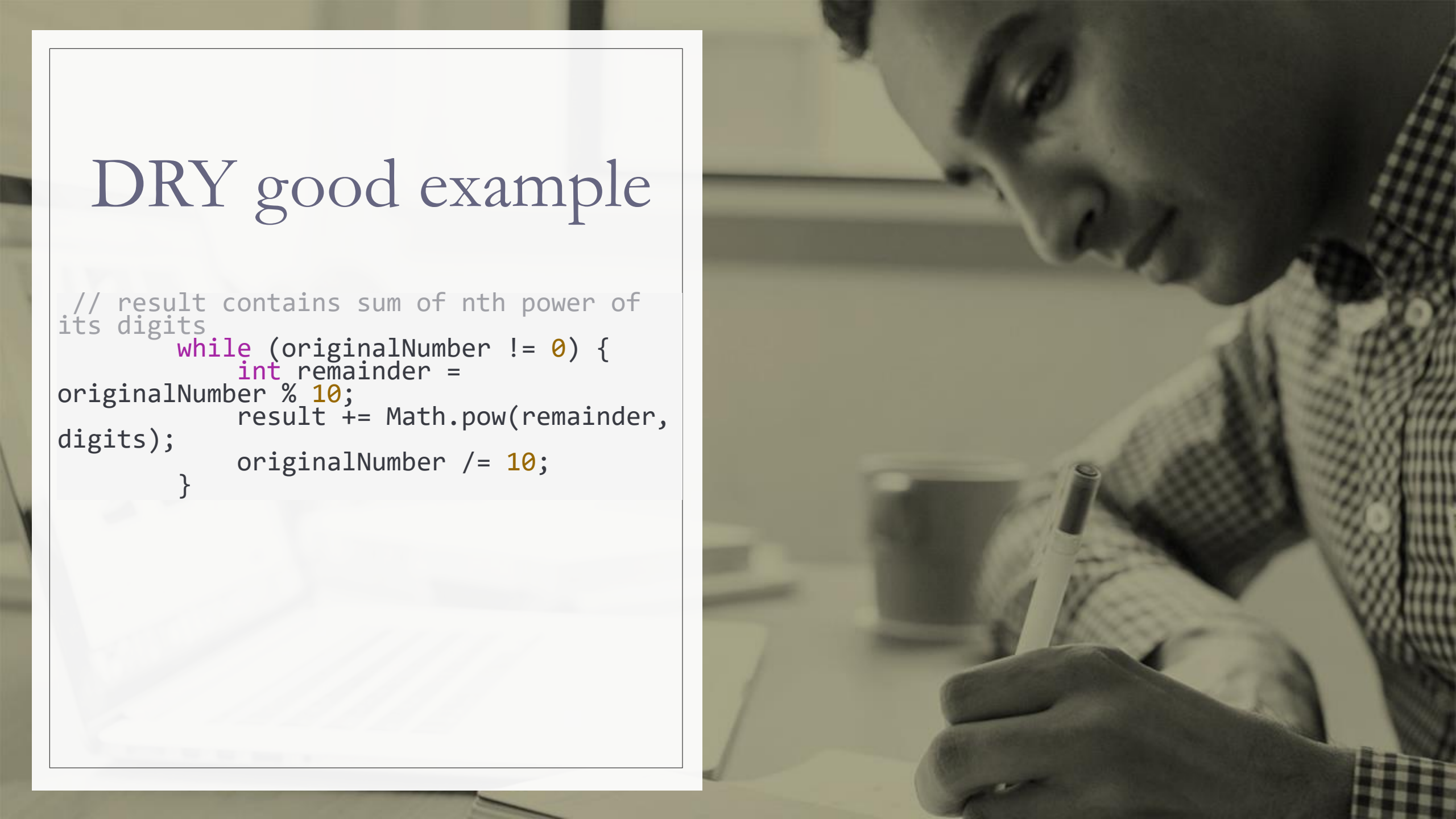
DRY good example

```
// number of digits calculation
      while (originalNumber != 0) {
         originalNumber /= 10;
         ++digits;
      }
```



DRY good example

```
 // result contains sum of nth power of
its digits
        while (originalNumber != 0) {
            int remainder =
originalNumber % 10;
            result += Math.pow(remainder,
digits);
            originalNumber /= 10;
        }
```



GNU: General Public License

```
<one line to give the program's name and a brief idea of what it does.>
Copyright (C) <year>  <name of author>

This program is free software: you can redistribute it and/or modify
it under the terms of the GNU General Public License as published by
the Free Software Foundation, either version 3 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program.  If not, see <https://www.gnu.org/licenses/>.
```

GNU: General Public License

```
27
28 /**
29  * <p>This class represents one share in a company. The class is an implementation
30  * of the singleton pattern. This means that there is one unique Share instance
31  * stored in a map per named key. The key to the map is the unique three letter
32  * designation of the company e.g. BBC</p>
33  *
34  * <p>This program is part of AJP-P3-2012-2013-SOLUTION.</p>
35  *
36  * <p>AJP-P3-2012-2013-SOLUTION is free software: you can redistribute it and/or
37  * modify it under the terms of the GNU General Public License as published by
38  * the Free Software Foundation, either version 3 of the License, or (at your
39  * option) any later version.</p>
40  *
41  * <p>This program is distributed in the hope that it will be useful, but
42  * WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or
43  * FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more
44  * details.</p>
45  *
46  * <p>You should have received a copy of the GNU General Public License along
47  * with this program. If not, see http://www.gnu.org/licenses/.</p>
48  *
49  * <p>Copyright Kiril Anastasov L1087591@tees.ac.uk 17-Oct-2012 </p>
50  */
51
```

Good comments

- Ownership and licenses
- //TODO
- Complex algorithms

Intent or Apology

```
//intent  Driver is working
if(driver.status == 1) {
}
If(driver.status ==
status.Working) {
}


// sorry I did not feel like
refactoring this code
```

Bad comments

- Documented code
- Describing behavior
- Justifying you programming

Documented code

UserDto.java

```java
package com.kirilanastasov.reservationservices.dto.model.user;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Getter
@Setter
@Accessors(chain = true)
@NoArgsConstructor
@ToString
@JsonInclude(value = JsonInclude.Include.NON_NULL)
@JsonIgnoreProperties(ignoreUnknown = true)
public class UserDto {
    private String email;
    private String password;
    private String firstName;
    private String lastName;
    private String mobileNumber;
    private boolean isAdmin;
    private Set<RoleDto> roles;

//    public String getFullName() {
//      if(firstName!= null)
//      return firstName + lastName;
//    }

    public String getFullName() {
        return firstName != null ? firstName.concat(" ").concat(lastName) : "";
    }
}

```

Describe behavior

```
// If this happens, somebody's been screwing around with the database definitions and
// has removed the restriction that a given alarm may have only one entry in the
// notifications table.  Bad maintenance programmer!  Bad!  No biscuit!


// If an alert is active on our side but inactive on theirs, that might mean
// they closed the alert.  (Or that we just haven't told them about it yet.)  The
// logic comes later; for now, we'll just compile it in a list.


// If we know for a fact that an alarm isn't getting through, we're going to whine pretty
// aggressively about it until it gets fixed.
```

Bad comments

```
30
31  // the main entry method for the program, which
32  // the java command line program will execute.
33  public static void main(String[] args) {
34      // prompt the user for their name using System.out, which
35      // is a PrintStream class. The PrintStream class has a
36      // method called println, which will output the text
37      // passed to the console (so that the user can see it)
38      // and then print a newline.
39      System.out.println("Welcome to my program! What is your name? ");
40
41      // this makes a new scanner, which can read from
42      // STDIN, located at System.in. The scanner lets us look
43      // for tokens, aka stuff the user has entered.
44      Scanner sc = new Scanner(System.in);
45
46      // this will create a new string variable called "name"
47      // and set the value to whatever the user types next
48      String name = sc.nextLine();
49
50      // finally we use println again, and string
51      // concatenation, with the + operator, to
52      // output or (sic) message!!
53      System.out.println("Hello " + name + " ! ");
54  }
55
```

Bad comments

```
30
31      while(isActive)
32      {
33          if(isActive)
34          {
35          }
36          else // !isActive
37          {
38          }
39      } // end while( isActive )
40
```

Checkstyle and PMD

Stay consistent

- Default Constructors
- Clear variables: firstName, lastName, color
- Comments are for documentaion

Structure

- Self-explanatory code
- Comments are deceiving.
- Code evolves comments do not

Formatting

- Consistency for the team
- File sizes should be small
- Same indentation style

Comments Lesson Summary

- Comments should be rare
- Comments should not be mandatory
- Consistent Formatting

Course Progress

Lesson 1	Lesson 2	Lesson 3	Lesson 4	Lesson 5
Clean code	Names	Methods	Classes	Comments


THANK YOU!