



[Home](#)

[Java Core](#)

[Java SE](#)

[Java EE](#)

[Frameworks](#)

[IDEs](#)

[Servers](#)

[Coding](#)

[Books](#)

[Videos](#)

[Java Skills Test](#)



[Home](#) ▶ [IDEs](#) ▶ Eclipse

Search...

**File APIs**

Try Free  
30 Day Trial

DOC XLS PPT PDF PNG  
XML HTML EML RTF VSD  
BMP Barcode images...



ASPOSE



All kinds of Java hosting since 2010

**FREE** 30-DAY TRIAL  
**INSTANT** ACTIVATION  
**PRIVATE** JVM + cPanel  
**BEST** PRICE GUARANTEE



30%  
off

[The Pragmatic ...](#)

Paperback

~~\$49.99~~ **\$34.99**

33%  
off

[Clean Code: A...](#)

Paperback

~~\$49.99~~ **\$33.32**

27%  
off

[The Clean Coder: A...](#)

Paperback

~~\$44.99~~ **\$32.82**

## Latest Eclipse Books

**Eclipse 4 Application  
Development: The complet  
guide to Eclipse 4 RCP  
development (Volume 1)**

**Eclipse IDE: Eclipse IDE**

**based on Eclipse 4.2 and 4.  
(vogella series) (Volume 2)**

**Eclipse IDE (vogella)**

**Eclipse 4 Plug-in  
Development by Example:  
Beginner's Guide**

**Eclipse Rich Client Platform  
(2nd Edition)**

**Eclipse Plug-ins (3rd Edition)**

**Latest Eclipse Releases**

**Eclipse Luna (4.4)**

**Previous Eclipse Releases**

Eclipse Kepler (4.3)

Eclipse Juno (4.2)

Eclipse Indigo (3.7)

Eclipse Helios (3.6)

All Eclipse Releases

**Eclipse Downloads**

**Eclipse Luna (4.4)**

Eclipse Kepler (4.3)

Eclipse Juno (4.2)

Eclipse Indigo (3.7)

Eclipse Helios (3.6)

**Eclipse Documentation**

**Eclipse Luna (4.4)**

Eclipse Kepler (4.3)

Eclipse Juno (4.2)

Eclipse Indigo (3.7)

Eclipse Helios (3.6)

Last updated: 2014-12-14

---





# 保柏300萬保額個人醫保

保障住院及手術後護理費，住半私家房，入院免找數，提供墊底費選擇減低保費



## 25 Eclipse Shortcut Keys for Code Editing

Last Updated on 16 August 2015 | [Print](#) [Email](#)

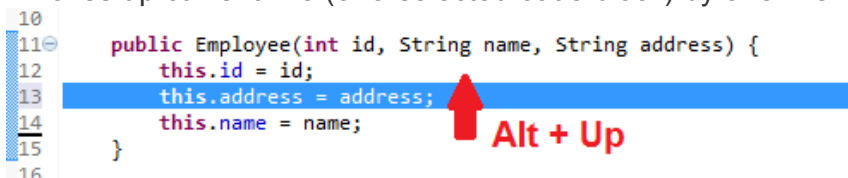
[Java Performance](#) [Clean Code](#) [The Clean Coder](#) [The Pragmatic Programmer](#)

When using an IDE, you cannot be more productive without using its shortcut keys frequently as your habit.

In this article, we summarize a list of shortcut keys which are useful for editing Java code in Eclipse IDE.

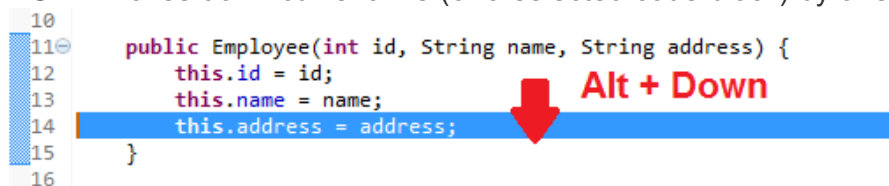
**NOTE:** Standard shortcuts are not covered, such as Ctrl + A (select all), Ctrl + Z (undo), etc.

1. **Ctrl + D:** Deletes current line.
2. **Ctrl + Delete:** Deletes next word after the cursor.
3. **Ctrl + Shift + Delete:** Deletes from the cursor until end of line.
4. **Ctrl + Backspace:** Deletes previous word before the cursor.
5. **Shift + Ctrl + y:** Changes a selection to lowercase.
6. **Shift + Ctrl + x:** Changes a selection to uppercase.
7. **Alt + Up Arrow:** Moves up current line (or a selected code block) by one line:



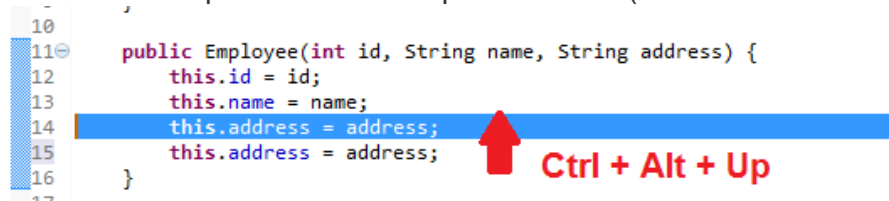
```
10
11 public Employee(int id, String name, String address) {
12     this.id = id;
13     this.address = address;
14     this.name = name;
15 }
16
```

8. **Alt + Down Arrow:** Moves down current line (or a selected code block) by one line:



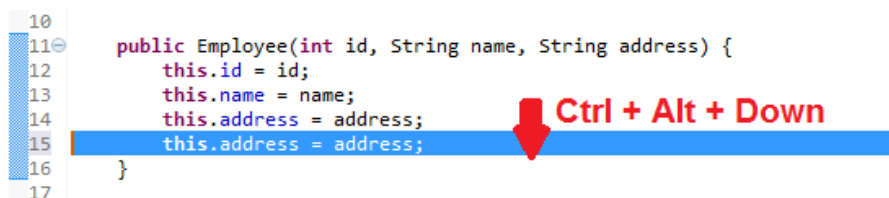
```
10
11 public Employee(int id, String name, String address) {
12     this.id = id;
13     this.name = name;
14     this.address = address;
15 }
16
```

9. **Ctrl + Alt + Up Arrow:** Copies and moves up current line (or a selected code block) by one line:



```
10
11 public Employee(int id, String name, String address) {
12     this.id = id;
13     this.name = name;
14     this.address = address;
15     this.address = address;
16 }
17
```

10. **Ctrl + Alt + Down Arrow:** Copies and moves down current line (or a selected code block) by one line:



```
10
11 public Employee(int id, String name, String address) {
12     this.id = id;
13     this.name = name;
14     this.address = address;
15     this.address = address;
16 }
17
```

11. **Shift + Enter:** Inserts a blank line after current line, regardless where the cursor is at the current line (very different from press **Enter** key alone):



```

10
11 public Employee(int id, String name, String address) {
12     this.id = id;
13     this.name = name;
14     this.address = address;
15 }
16
17

```

**Shift + Enter**

12. **Ctrl + Shift + Enter:** works similar to the **Shift + Enter**, but inserts a blank line just before the current line.
13. **Ctrl + Shift + O:** Organizes import statements by removing unused imports and sorts the used ones alphabetically. This shortcut also adds missing imports.

```

3 import java.security.NoSuchAlgorithmException;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Controller;
7 import org.springframework.web.bind.annotation.ModelAttribute;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RequestMethod;
10 import org.springframework.web.bind.annotation.ControllerAdvice;
11 import org.springframework.web.servlet.ModelAndView;
12
13 import java.util.List;
14 import java.util.ArrayList;
15
16 import com.ava.hr.dao.EmployeeDAO;
17 import com.ava.hr.dao.UserDAO;
18 import com.ava.hr.model.Employee;
19 import com.ava.hr.model.User;
20

```

**Ctrl + Shift + O to remove these unused imports**

14. **Ctrl + Shift + M:** Adds a single import statement for the current error due to missing import. You need to place the cursor inside the error and press this shortcut:

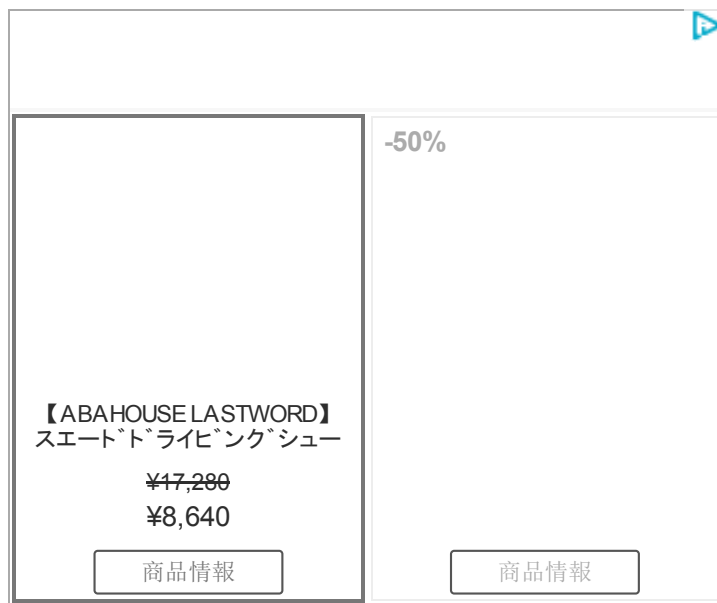
```

25 @RequestMapping(value="/")
26 public ModelAndView viewIndex() {
27     ModelAndView model = new ModelAndView("index");
28     model.addObject("user", new User());
29     return model;
30 }
31

```

**Ctrl + Shift + M to add import for the User class**

15. **Ctrl + Shift + F:** Formats a selected block of code or a whole source file. This shortcut is very useful when you want to format messy code to Java-standard code. Note that, if nothing is selected in the editor, Eclipse applies formatting for the whole file:



```
9 public class EmployeeDAO {
10     private SessionFactory sessionFactory;
11
12     public EmployeeDAO(SessionFactory sessionFactory) {
13         this.sessionFactory = sessionFactory;
14     }
15
16     @Transactional
17     public void save(Employee employee)
18     {
19         Session session = sessionFactory.getCurrentSession();
20
21         session.save(employee);
22     }
23
24 }
25
```

**Ctrl + Shift + F to format  
this messy code**

16. **Ctrl + I**: Corrects indentation for current line or a selected code block. This is useful as it helps you avoid manually using **Tab** key to correct the indentation:

```
26 @RequestMapping(value = "/")
27 public ModelAndView viewIndex() {
28     ModelAndView model = new ModelAndView("index");
29     model.addObject("user", new User());
30     return model;
31 }
```

**Press Ctrl + I  
to correct  
indentation of  
this line**

17. **Ctrl + /** or **Ctrl + 7**: Toggle single line comment. This shortcut adds single-line comment to current line or a block of code. Press again to remove comment. For example:

```

3 import org.hibernate.Session;
8
9 public class EmployeeDAO {
10     private SessionFactory sessionFactory;
11
12     public EmployeeDAO(SessionFactory sessionFactory) {
13         this.sessionFactory = sessionFactory;
14     }
15
16     // @Transactional
17     // public void save(Employee employee) {
18     //     Session session = sessionFactory.getCurrentSession();
19     //
20     //     session.save(employee);
21     // }
22
23 }
24

```

**Ctrl + / or Ctrl + 7**

18. **Ctrl + Shift + /**: Adds block comment to a selection.

```

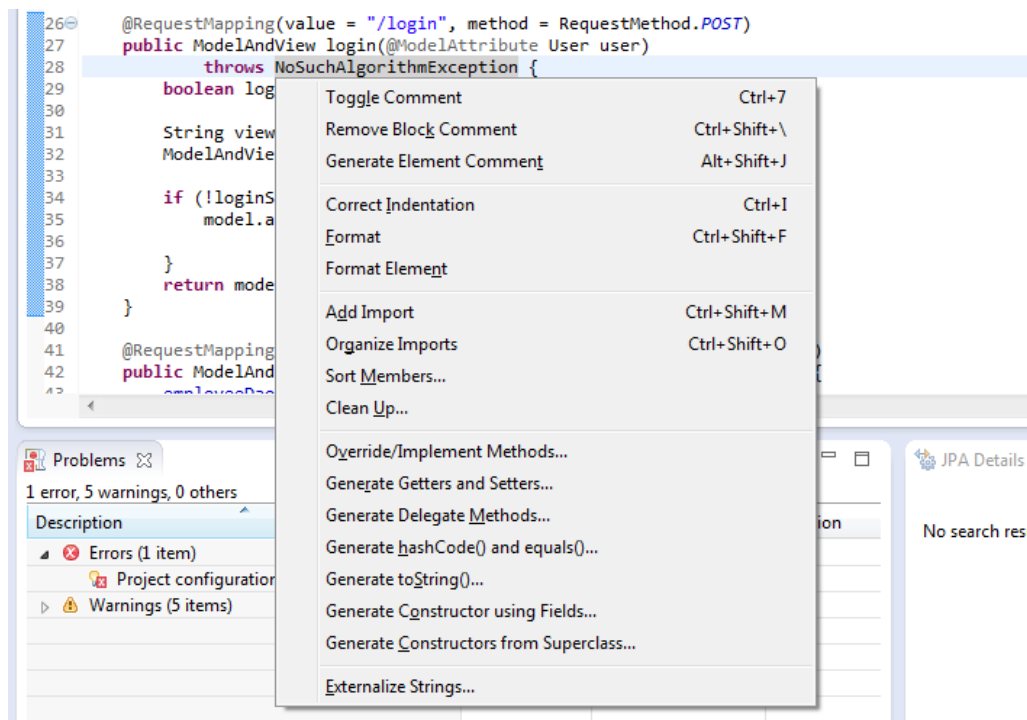
9 public class EmployeeDAO {
10     private SessionFactory sessionFactory;
11
12     public EmployeeDAO(SessionFactory sessionFactory) {
13         this.sessionFactory = sessionFactory;
14     }
15
16     /* @Transactional
17     public void save(Employee employee) {
18         Session session = sessionFactory.getCurrentSession();
19
20         session.save(employee);
21     }*/
22
23 }
24

```

**Ctrl + Shift + / to add a block comment**

19. **Ctrl + Shift + \**: Removes block comment.

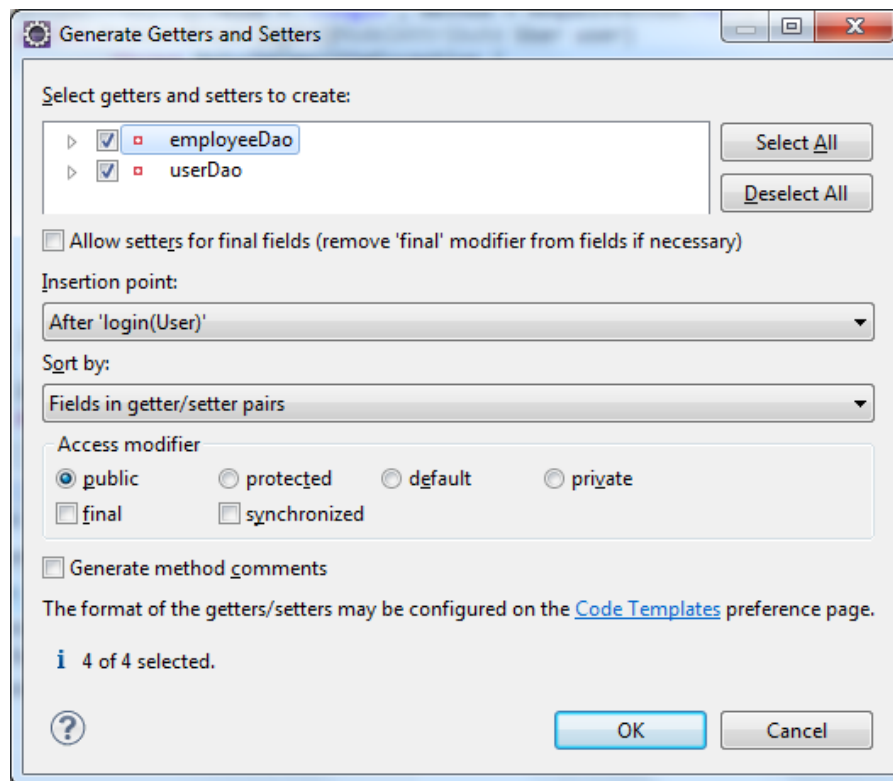
20. **Alt + Shift + S**: Shows context menu that lists possible actions for editing code:



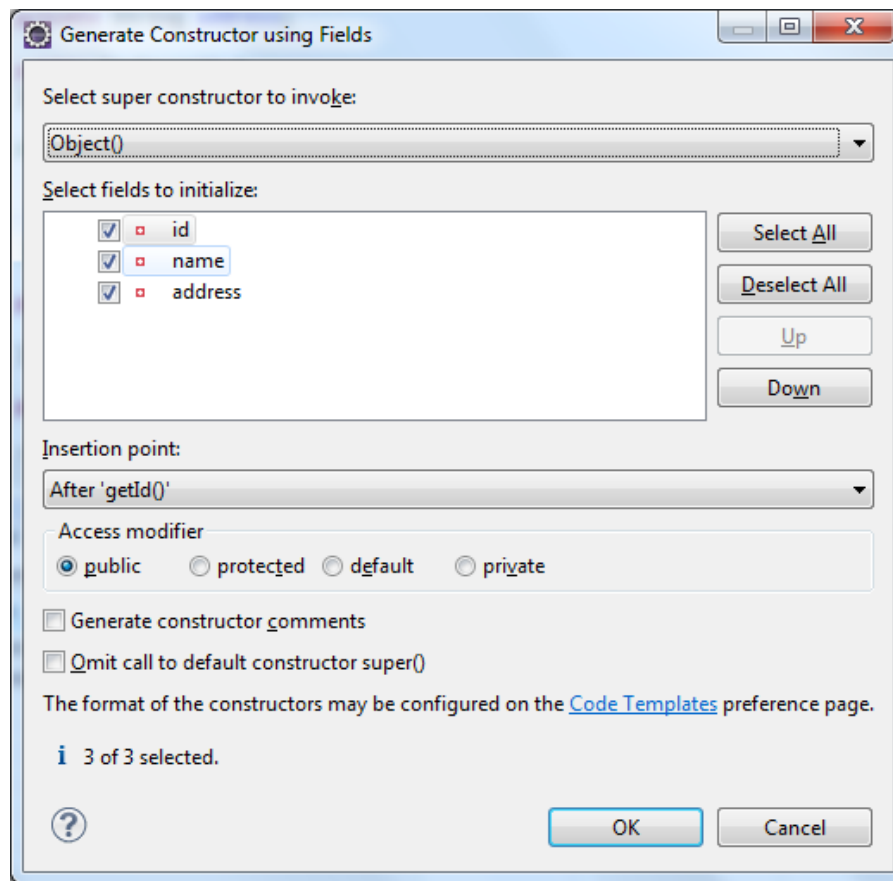
From this context menu, you can press another letter (according to the underscore letters in the names) to access the desired functions.

30% off	<a href="#">The Pragmatic ...</a> Andrew Hunt, David ... Paperback <del>\$49.99</del> <b>\$34.99</b>	33% off	<a href="#">Clean Code: A ...</a> Robert C. Martin Paperback <del>\$49.99</del> <b>\$33.32</b>	27% off	<a href="#">The Clear</a> Robert C. M Paperback <del>\$44.99</del> <b>\$3</b>
------------	---	------------	---	------------	--

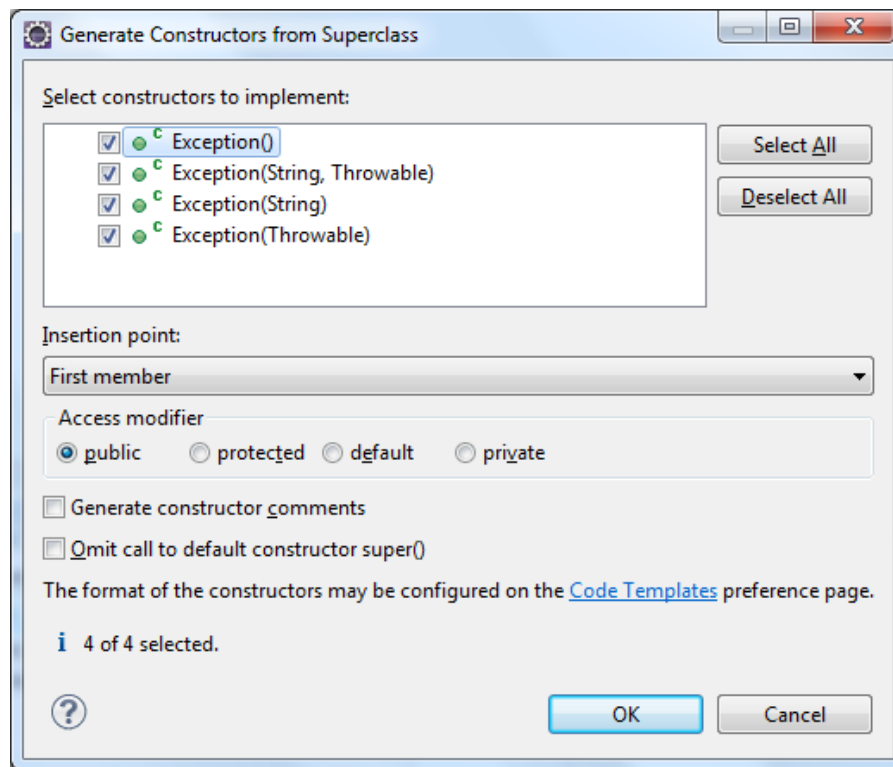
21. **Alt + Shift + S, R:** Generates getters and setters for fields of a class. This is a very handy shortcut that helps us generate getter and setter methods quickly. The following dialog appears:



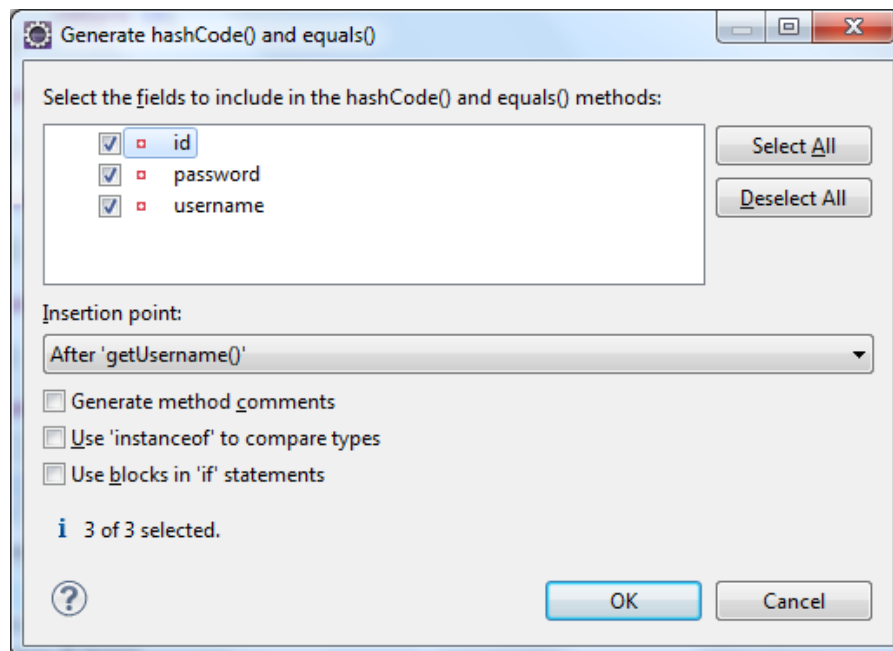
22. **Alt + Shift + S, O:** Generates constructor using fields. This shortcut is very useful when you want to generate code for a constructor that takes class' fields as its parameters. The following dialog appears:



23. **Alt + Shift + S, C:** Generates Constructors from Superclass. A common example for using this shortcut is when creating a custom exception class. In this case, we need to write some constructors similar to the `Exception` superclass. This shortcut brings the *Generate Constructors from Superclass* dialog which allows us to choose the constructors to be implemented in the subclass:



24. **Alt + Shift + S, H:** Generates `hashCode()` and `equals()` methods, typically for a JavaBean/POJO class. The class must have non-static fields. This shortcut brings the *Generate hashCode() and equals()* dialog as below:



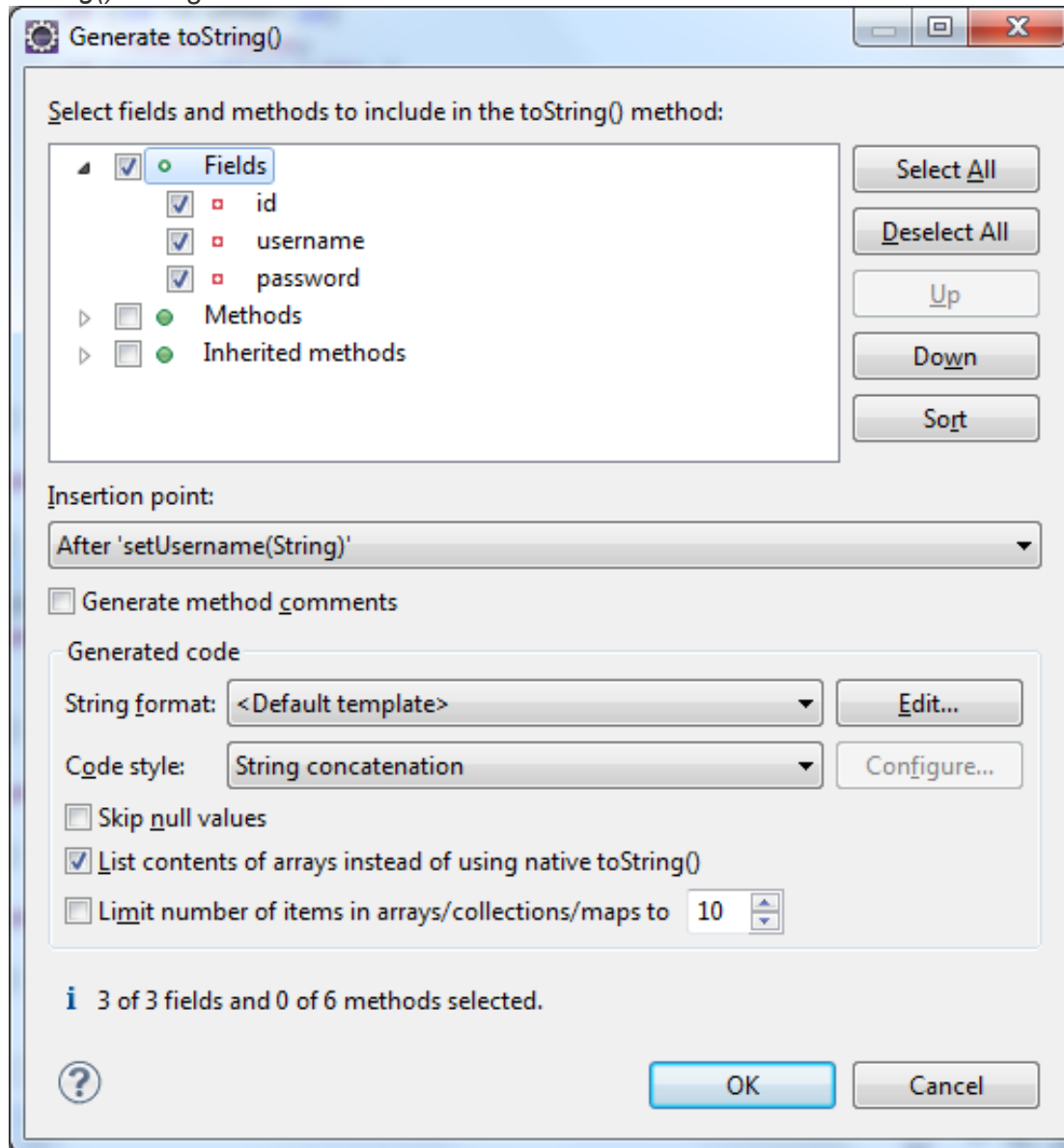
Select the fields to be used in `hashCode()` and `equals()` method, and then click OK. We got the following result (example):

```
@Override
public int hashCode() {
    final int prime = 31;
    int result = 1;
    result = prime * result + id;
    result = prime * result
        + ((password == null) ? 0 : password.hashCode());
    result = prime * result
        + ((username == null) ? 0 : username.hashCode());
    return result;
}

@Override
public boolean equals(Object obj) {
    if (this == obj)
        return true;
    if (obj == null)
        return false;
    if (getClass() != obj.getClass())
        return false;
    User other = (User) obj;
    if (id != other.id)
        return false;
    if (password == null) {
        if (other.password != null)
            return false;
    } else if (!password.equals(other.password))
        return false;
    if (username == null) {
        if (other.username != null)
            return false;
    } else if (!username.equals(other.username))
        return false;
    return true;
}
```



25. **Alt + Shift + S, S:** Generates `toString()` method. This shortcut comes in handy if we want to override `toString()` method that returns information of relevant fields of the class. This brings the *Generate toString()* dialog as below:



And here's sample of a generated `toString()` method:

```
@Override
public String toString() {
    return "User [id=" + id + ", username=" + username + ", password="
        + password + " ]";
}
```

**Do you want to be expert in Java programming?** If you do, why not join our mailing list to get advices from the professionals everyday? Just click here: <http://newsletter.codejava.net> - It's FREE, Quick and Awesome!

Add comment

---

☐ Notify me of follow-up comments





Type the text

[Privacy & Terms](#)

Send

## Comments

#1 **parham** 2015-10-14 09:15  
tnx guys i love you

0

[Quote](#)

[Refresh comments list](#)

[RSS feed for comments to this post](#)

JComments

[About](#) [Advertise](#) [Contribute](#) [Contact](#) [Terms of Use](#) [Privacy Policy](#) [Site Map](#) [Newsletter](#)



Copyright © 2012 - 2015 by [www.codejava.net](http://www.codejava.net)