

Program "Forum"

Inheritance is one of the key aspects of object-oriented programming. Using inheritance, you can extend the functionality of existing classes or change, as well as reduce code duplication.

One class can inherit the characteristics of another, namely its methods and variables.

Note: A class can only inherit members that are not marked with the private access modifier.

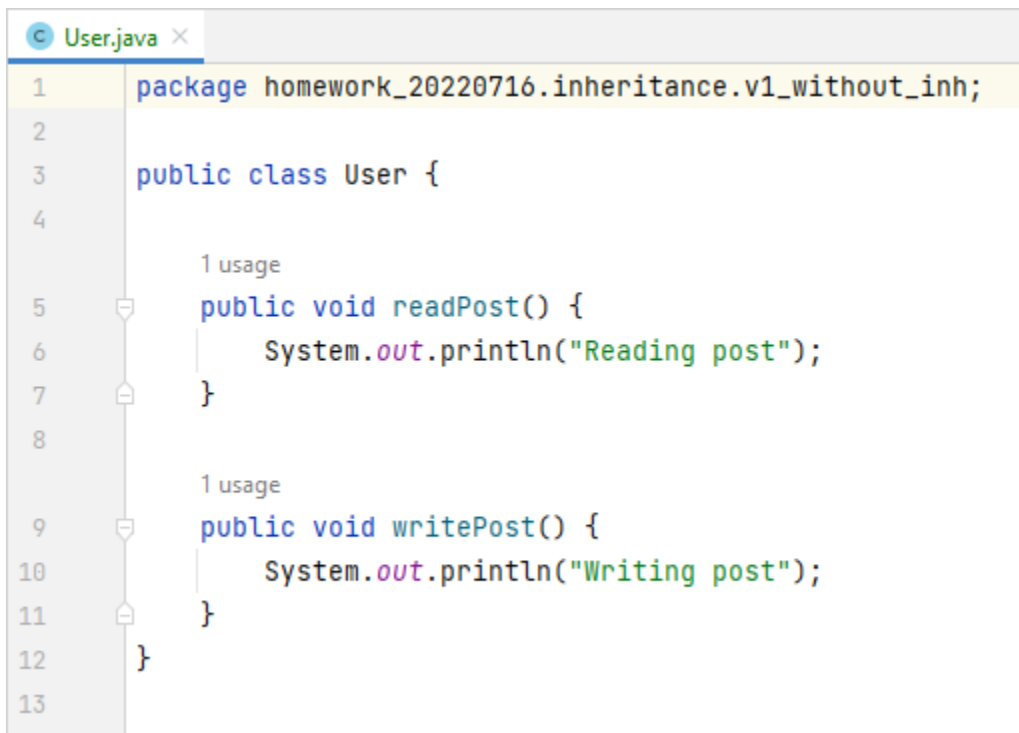
Consider a program that demonstrates inheritance using a forum as an example.

The forum has different types of accounts:

- User "User" - can read messages, write messages;
- Moderator "Moderator" - can delete messages and all available user options;
- Administrator "Administrator" - can block the user and all available moderator options;

Step #1

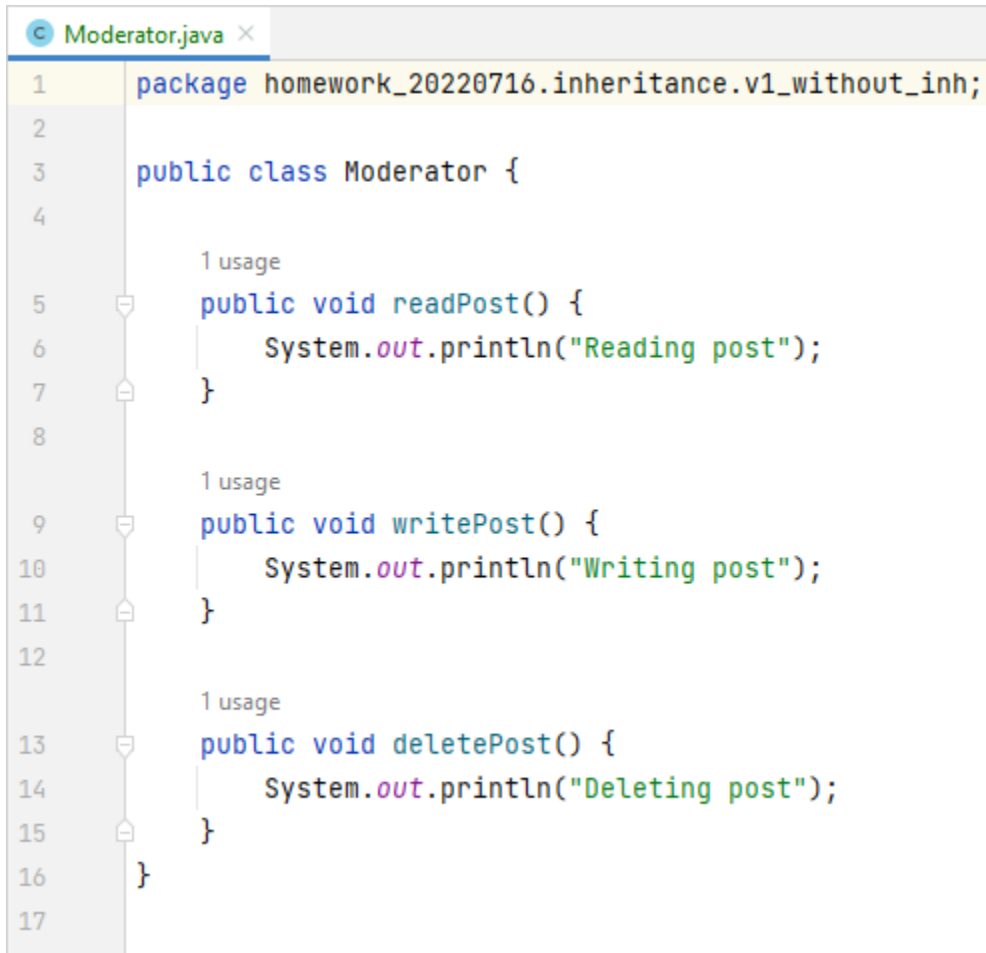
Let's create a class "User" with arbitrary logic for each of the methods:



```
1 package homework_20220716.inheritance.v1_without_inh;
2
3 public class User {
4
5     1 usage
6     public void readPost() {
7         System.out.println("Reading post");
8     }
9
10    1 usage
11    public void writePost() {
12        System.out.println("Writing post");
13    }
14 }
```

Step #2

Let's create a class "Moderator" with arbitrary logic that also performs the same actions as the user:



```
1 package homework_20220716.inheritance.v1_without_inh;
2
3 public class Moderator {
4
5     1 usage
6     public void readPost() {
7         System.out.println("Reading post");
8     }
9
10    1 usage
11    public void writePost() {
12        System.out.println("Writing post");
13    }
14
15    1 usage
16    public void deletePost() {
17        System.out.println("Deleting post");
18    }
19 }
```

Step #3

Let's create a class "Administrator" with arbitrary logic that also performs the same actions as the user and moderator:

```
Administrator.java x
1 package homework_20220716.inheritance.v1_without_inh;
2
3 public class Administrator {
4
5     1 usage
6     public void readPost() {
7         System.out.println("Reading post");
8     }
9
10    1 usage
11    public void writePost() {
12        System.out.println("Writing post");
13    }
14
15    1 usage
16    public void deletePost() {
17        System.out.println("Deleting post");
18    }
19
20    1 usage
21    public void blockUser() {
22        System.out.println("Blocking user");
23    }
24 }
```

Step #4

Let's demonstrate the work of the methods of each of the classes in the class "Main":

```
1 package homework_20220716.inheritance.v1_without_inh;
2
3 public class Main {
4     public static void main(String[] args) {
5
6         User user = new User();
7         Moderator moderator = new Moderator();
8         Administrator administrator = new Administrator();
9
10        user.readPost();
11        user.writePost();
12
13        System.out.println();
14
15        moderator.readPost();
16        moderator.writePost();
17        moderator.deletePost();
18
19        System.out.println();
20
21        administrator.readPost();
22        administrator.writePost();
23        administrator.deletePost();
24        administrator.blockUser();
25    }
26 }
27
```

The result of the program:

```
homework_20220716.inheritance.v1_without_inh.Main x
C:\Users\laurfial\.jdk\temurin-17.0.2\bin
Reading post
Writing post

Reading post
Writing post
Deleting post

Reading post
Writing post
Deleting post
Blocking user

Process finished with exit code 0
```

It can be seen that the main conditions of the program are met, but now there are a number of shortcomings in the code: the logic of the methods is the same, regardless of which class performs it, therefore this is code duplication. If it is necessary to change the logic of one of the methods, for example, a post entry, then it must be changed in all classes.

To reduce code duplication, the existing functionality is supplemented using the inheritance mechanism.

Step #5

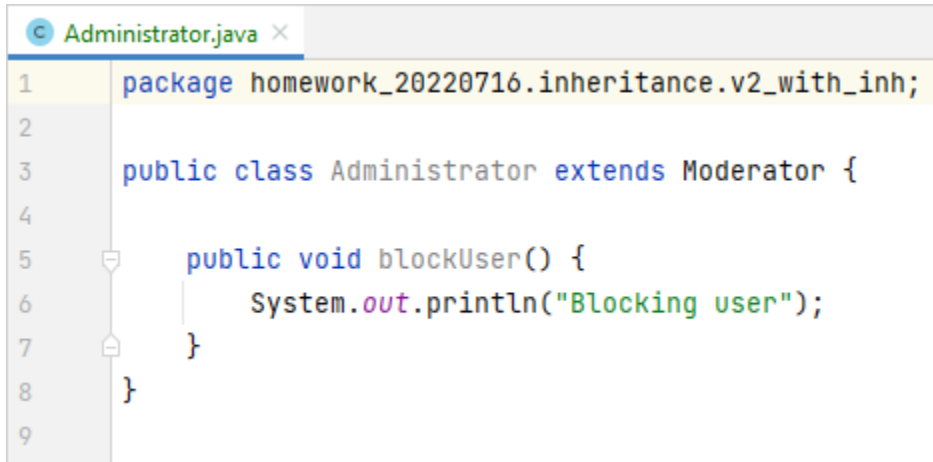
Let's change the code so that the "Moderator" logic is inherited from the "User" class:

```
Moderator.java x
1 package homework_20220716.inheritance.v2_with_inh;
2
3 public class Moderator extends User {
4
5     public void deletePost() {
6         System.out.println("Deleting post");
7     }
8 }
9
```

The result of the program has not changed. Please note that the logic of work has not changed, but the amount of code has become less.

Step #6

We inherit the logic of the "Moderator" class in the "Administrator" class:



```
Administrator.java x
1 package homework_20220716.inheritance.v2_with_inh;
2
3 public class Administrator extends Moderator {
4
5     public void blockUser() {
6         System.out.println("Blocking user");
7     }
8 }
9
```

The result of the program has not changed. The logic of work remains the same.

Step #7

Suppose the class "Administrator" writes a post with a different logic. In this case, the behavior of the method can be overridden:

```
C Administrator.java x
1 package homework_20220716.inheritance.v2_with_inh;
2
3 public class Administrator extends Moderator {
4
5     3 usages
6     @Override
7     public void writePost() {
8         System.out.println("Administrator: writing post");
9     }
10
11     1 usage
12     public void blockUser() {
13         System.out.println("Blocking user");
14     }
15 }
```

The result of the program:

```
homework_20220716.inheritance.v2_with_inh.Main x
C:\Users\laurfial\.jdk\temurin-17.0.2\bin
Reading post
Writing post

Reading post
Writing post
Deleting post

Reading post
Administrator: writing post
Deleting post
Blocking user

Process finished with exit code 0
```

It can be seen that the logic of work has changed only in the "Administrator" class. If necessary, you can override methods at any point in the hierarchy, but in this case, the changes will affect all classes - heirs.