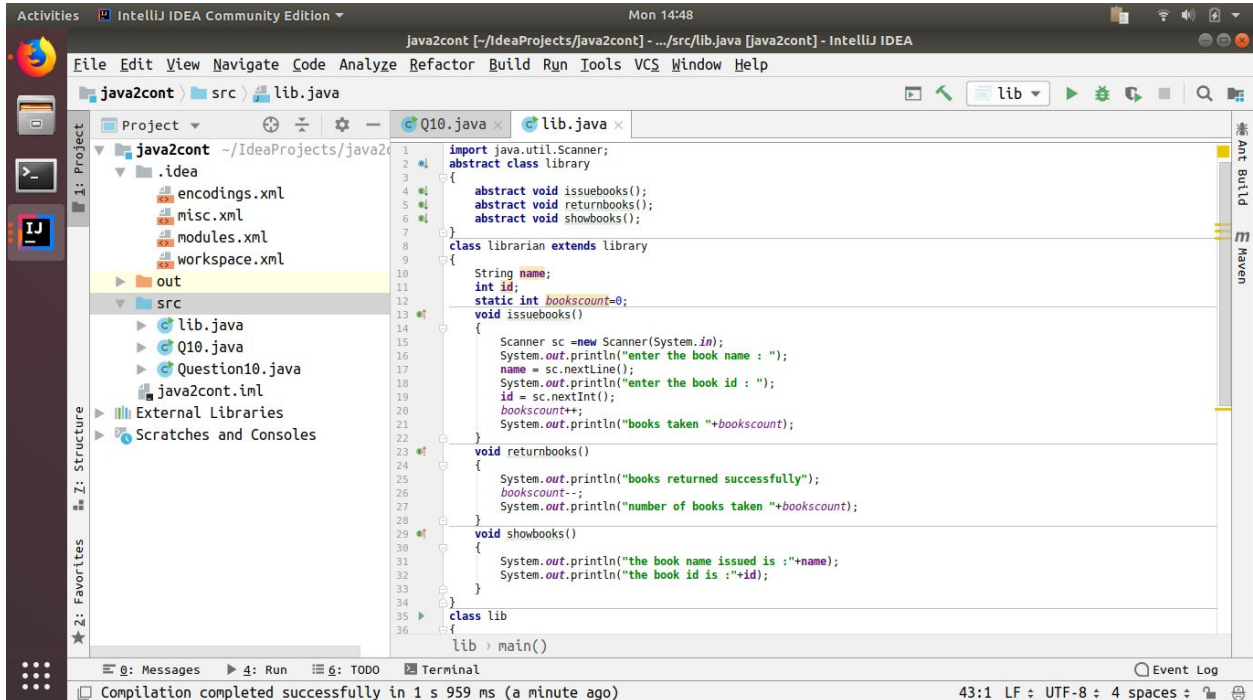


Java Assignment 2

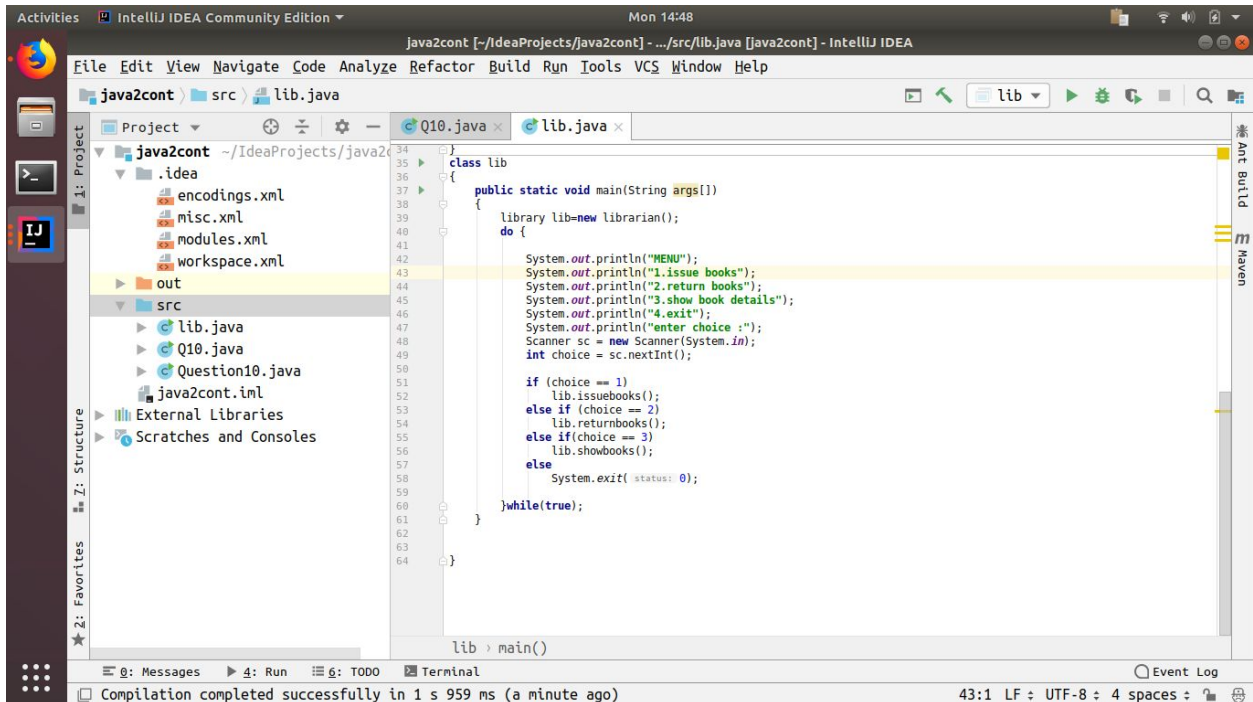
Note: Use Java keywords and concepts in problem solving.

1. Create Java classes having suitable attributes for Library management system. Use OOPs concepts in your design. Also try to use interfaces and abstract classes.



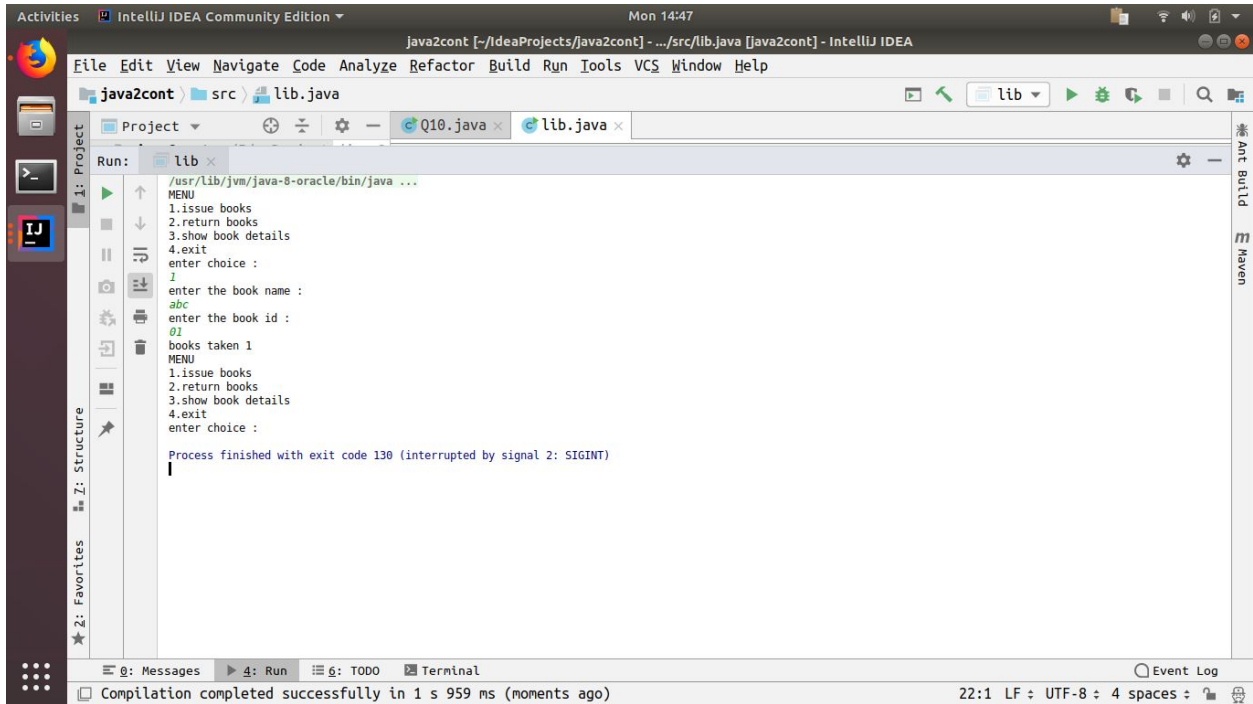
```
1 import java.util.Scanner;
2 abstract class Library
3 {
4     abstract void issuebooks();
5     abstract void returnbooks();
6     abstract void showbooks();
7 }
8 class Librarian extends Library
9 {
10     String name;
11     int id;
12     static int bookcount=0;
13     void issuebooks()
14     {
15         Scanner sc = new Scanner(System.in);
16         System.out.println("enter the book name : ");
17         name = sc.nextLine();
18         System.out.println("enter the book id : ");
19         id = sc.nextInt();
20         bookcount++;
21         System.out.println("books taken "+bookcount);
22     }
23     void returnbooks()
24     {
25         System.out.println("books returned successfully");
26         bookcount--;
27         System.out.println("number of books taken "+bookcount);
28     }
29     void showbooks()
30     {
31         System.out.println("the book name issued is :"+name);
32         System.out.println("the book id is :"+id);
33     }
34 }
35 class lib
36 {
37     lib > main()
```

Compilation completed successfully in 1 s 959 ms (a minute ago)

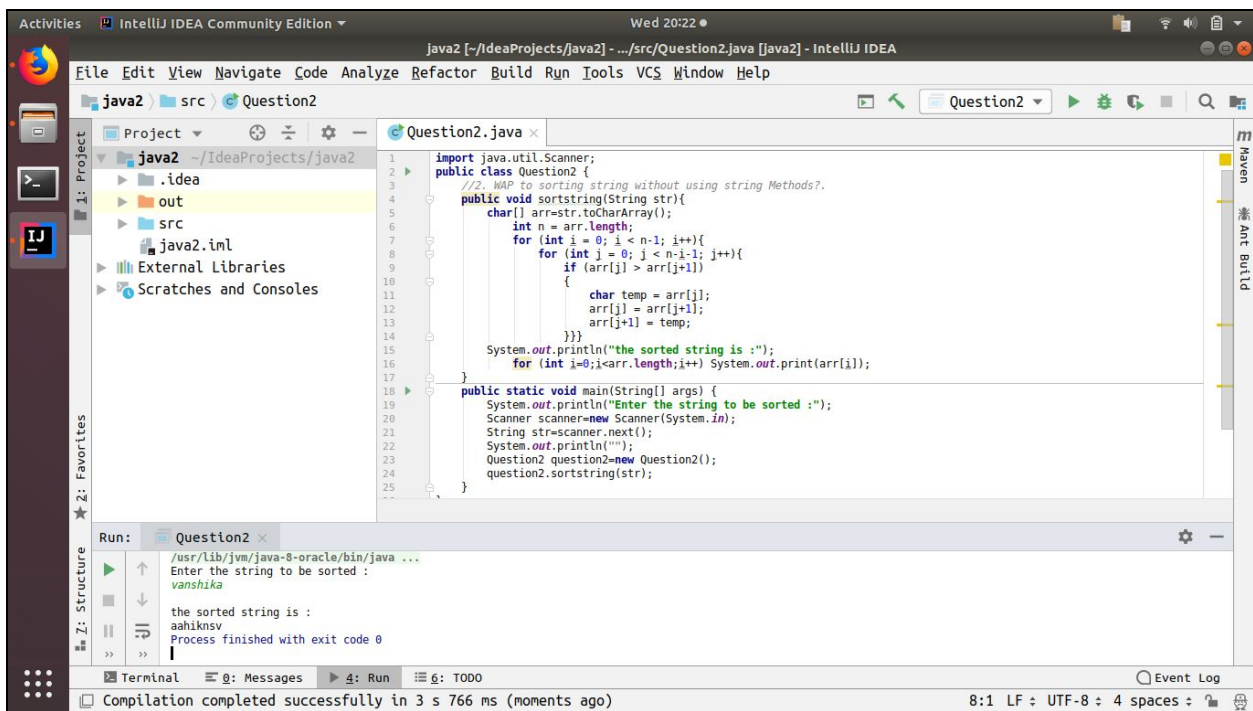


```
34 class lib
35 {
36     public static void main(String args[])
37     {
38         library lib=new Librarian();
39         do {
40             System.out.println("MENU");
41             System.out.println("1.issue books");
42             System.out.println("2.return books");
43             System.out.println("3.show book details");
44             System.out.println("4.exit");
45             System.out.println("enter choice :");
46             Scanner sc = new Scanner(System.in);
47             int choice = sc.nextInt();
48             if (choice == 1)
49                 lib.issuebooks();
50             else if (choice == 2)
51                 lib.returnbooks();
52             else if (choice == 3)
53                 lib.showbooks();
54             else
55                 System.exit( status: 0);
56         }while(true);
57     }
58 }
59
60 lib > main()
```

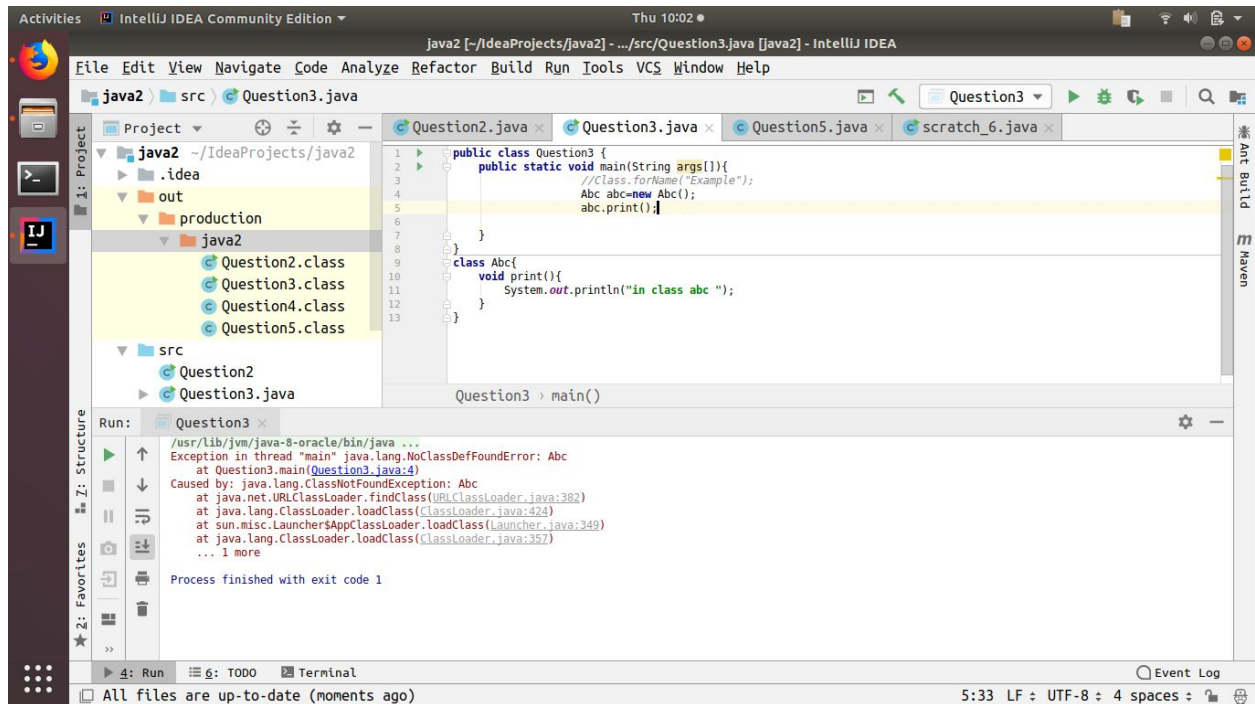
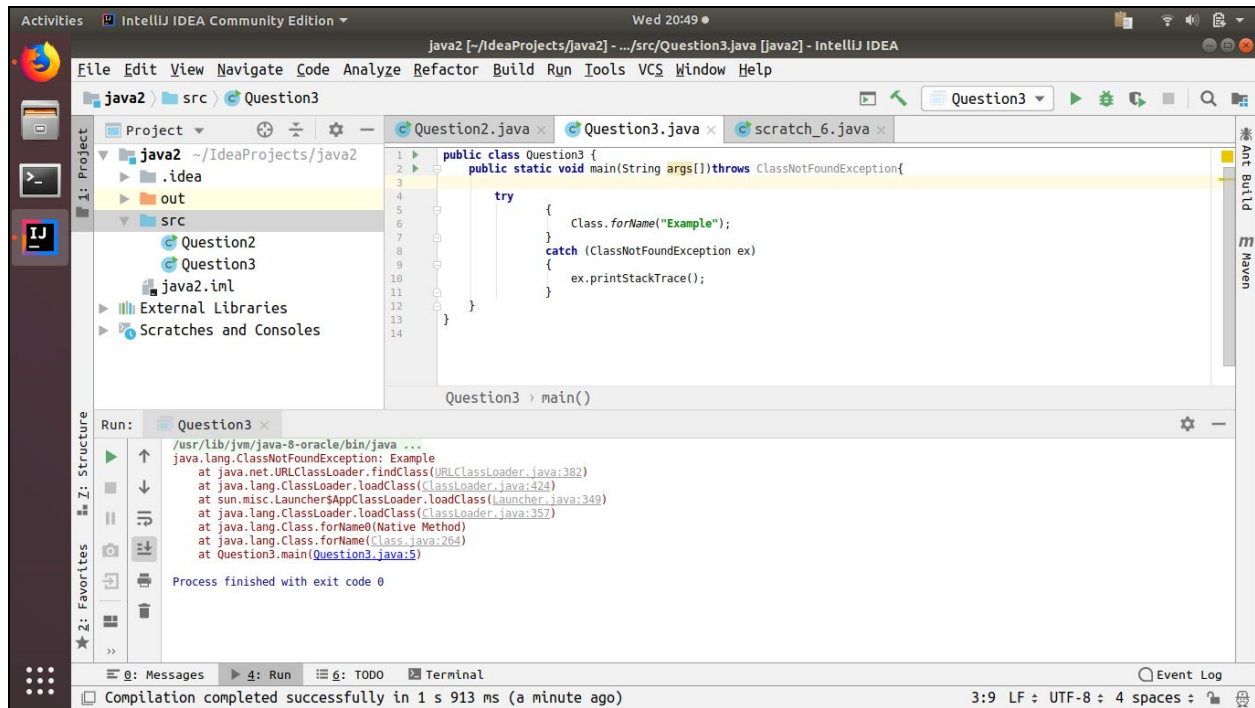
Compilation completed successfully in 1 s 959 ms (a minute ago)



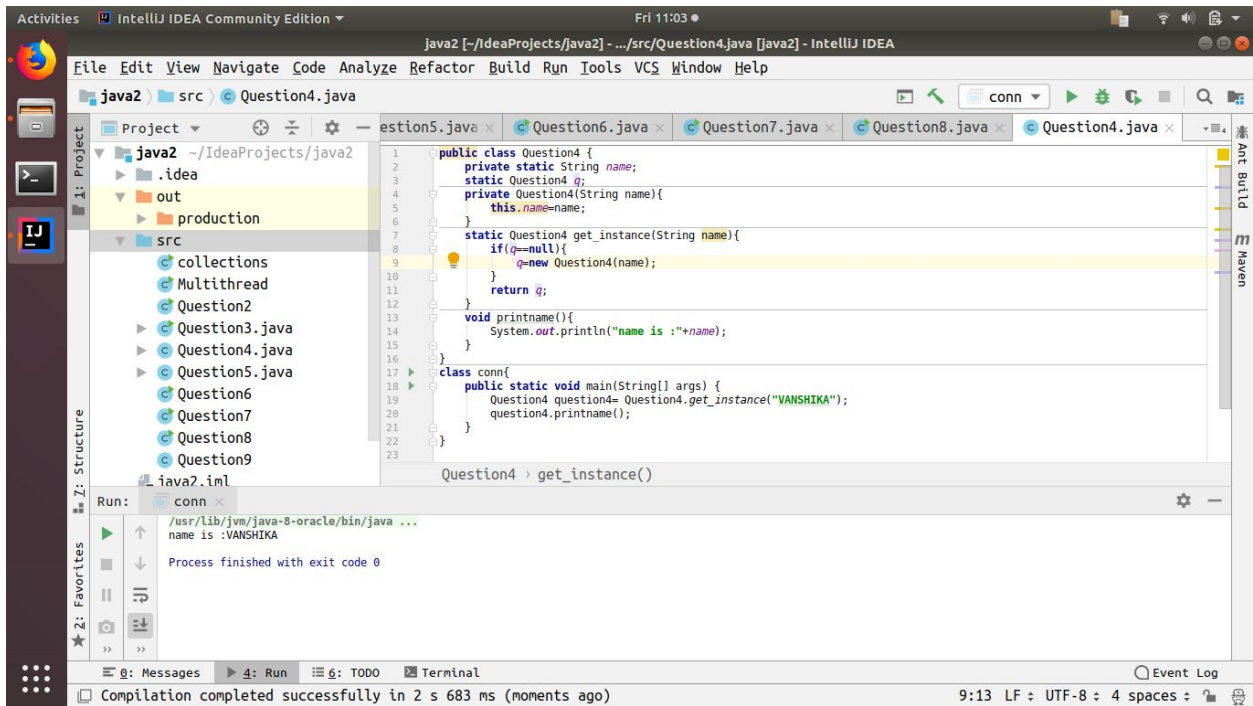
2. WAP to sorting string without using string Methods?.



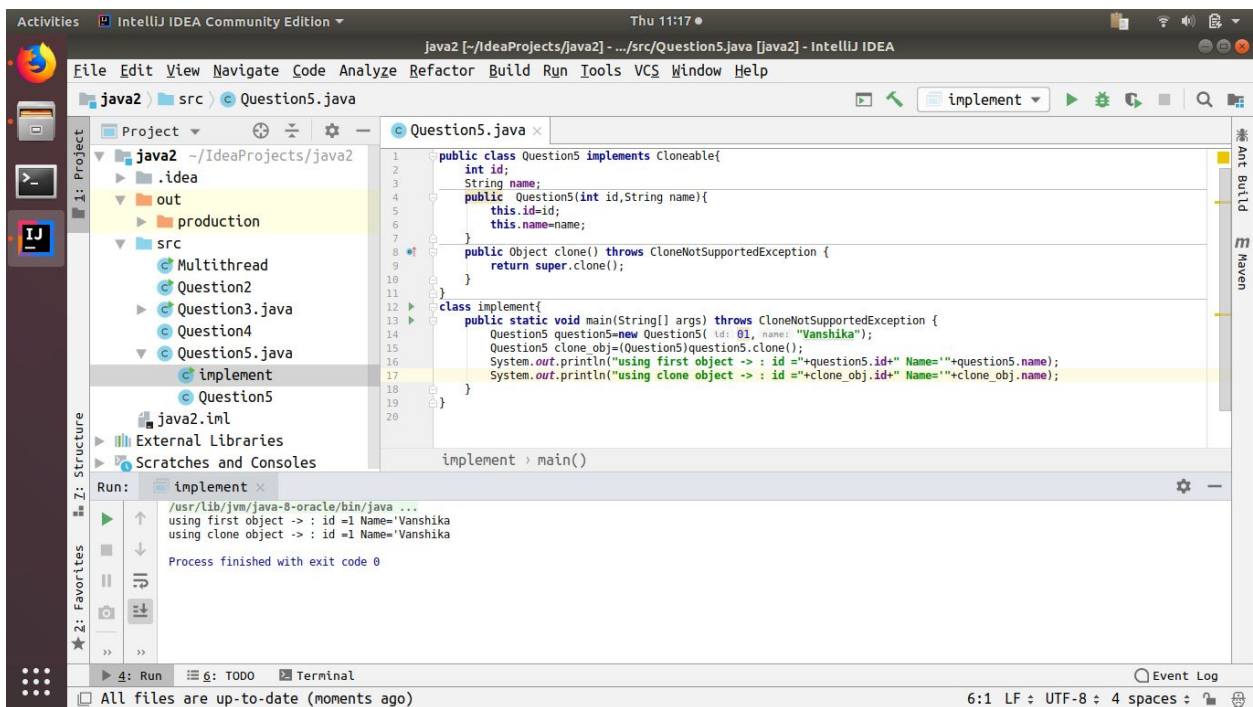
3. WAP to produce NoClassDefFoundError and ClassNotFoundException exception.



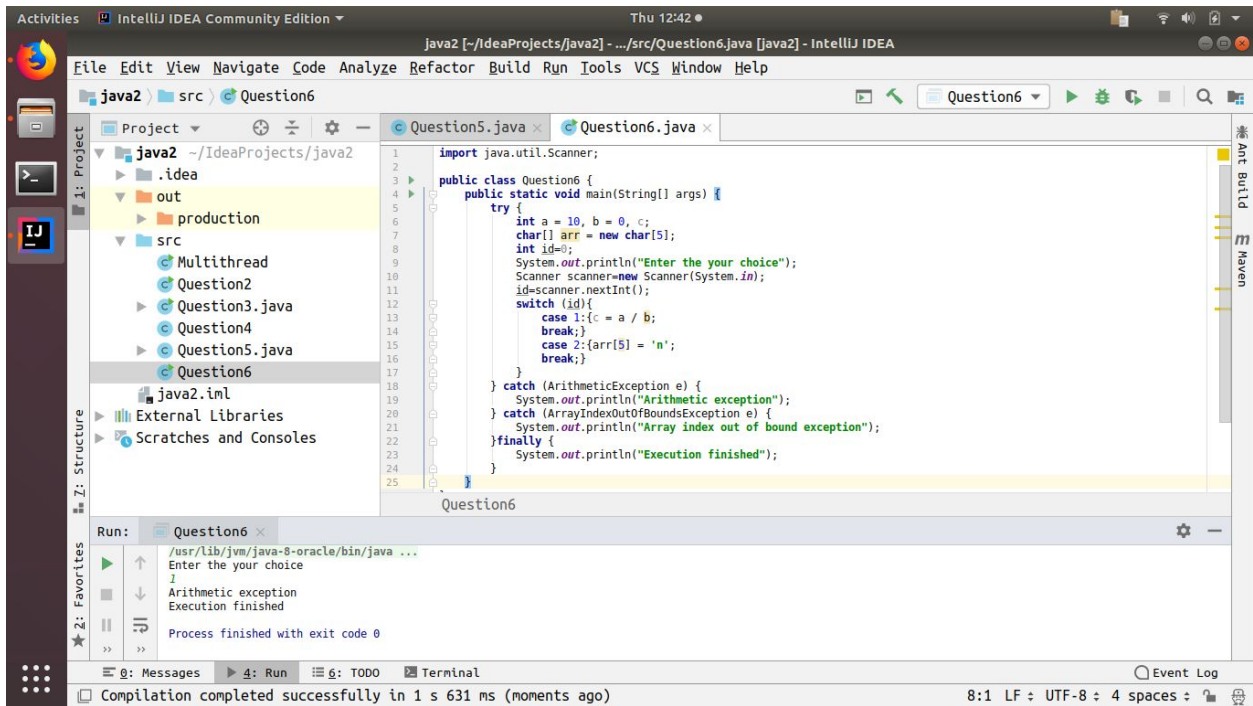
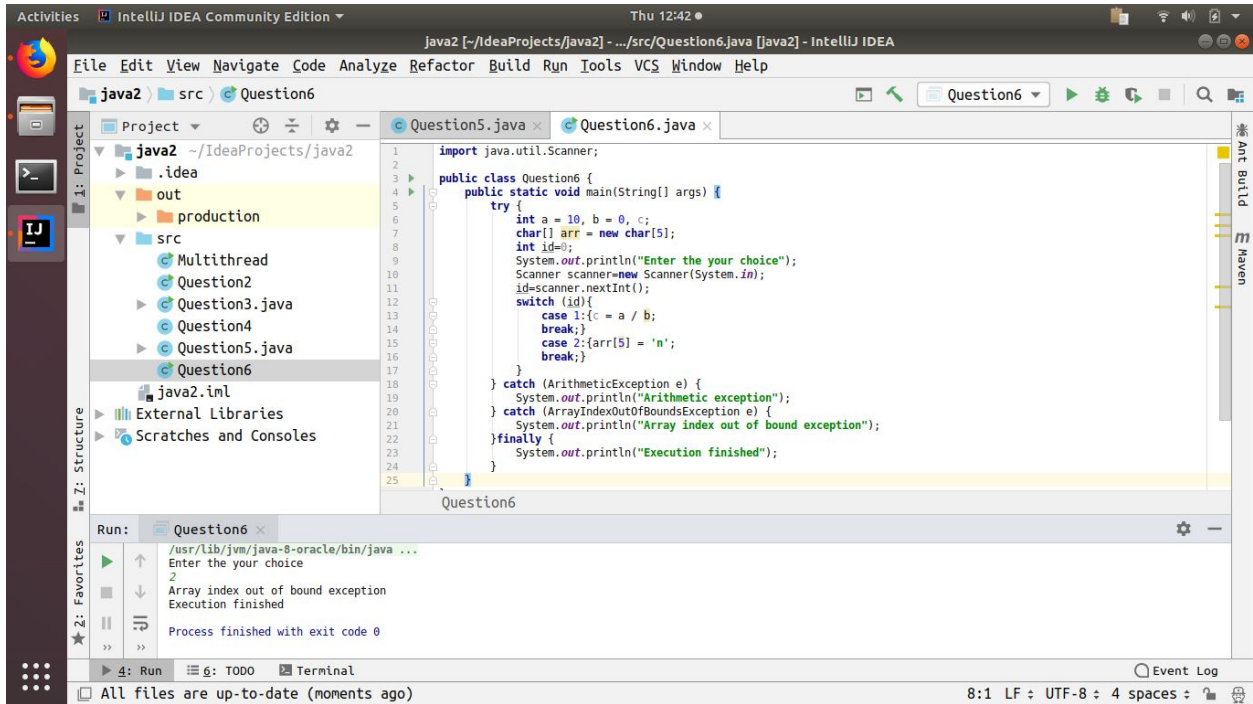
4. WAP to create singleton class.



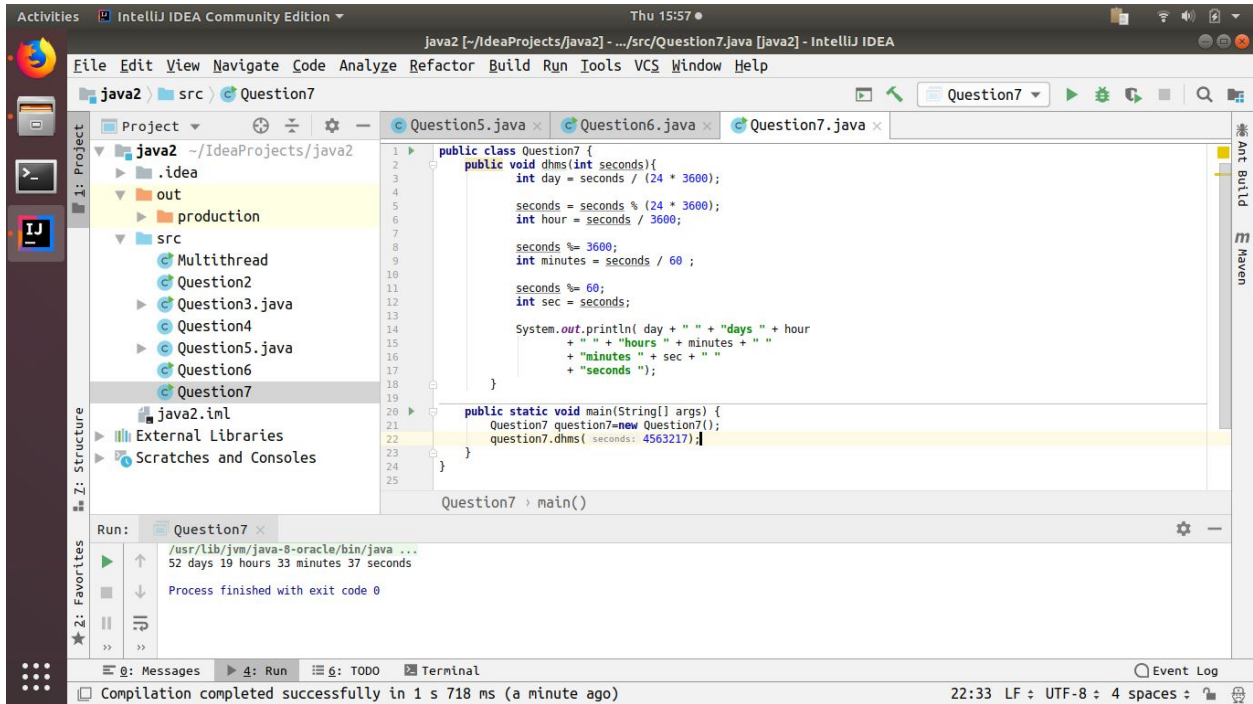
5. WAP to show object cloning in java using cloneable and copy constructor both.



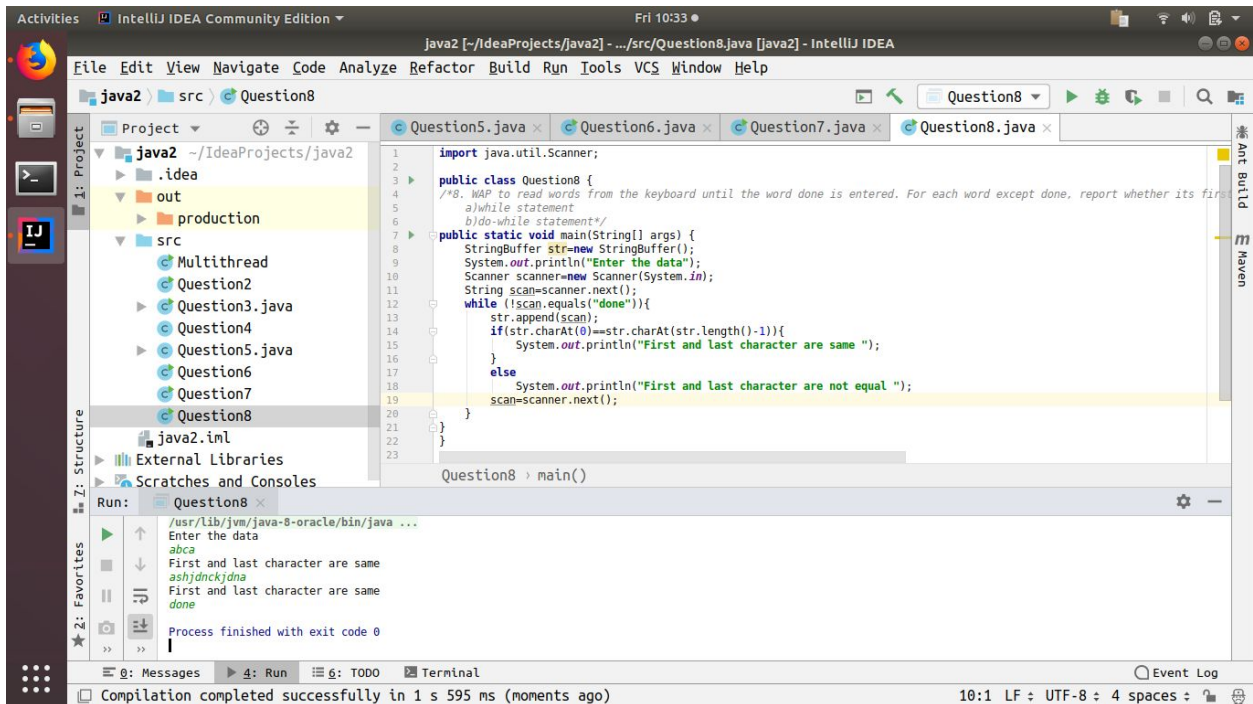
6. WAP showing try, multi-catch and finally blocks.



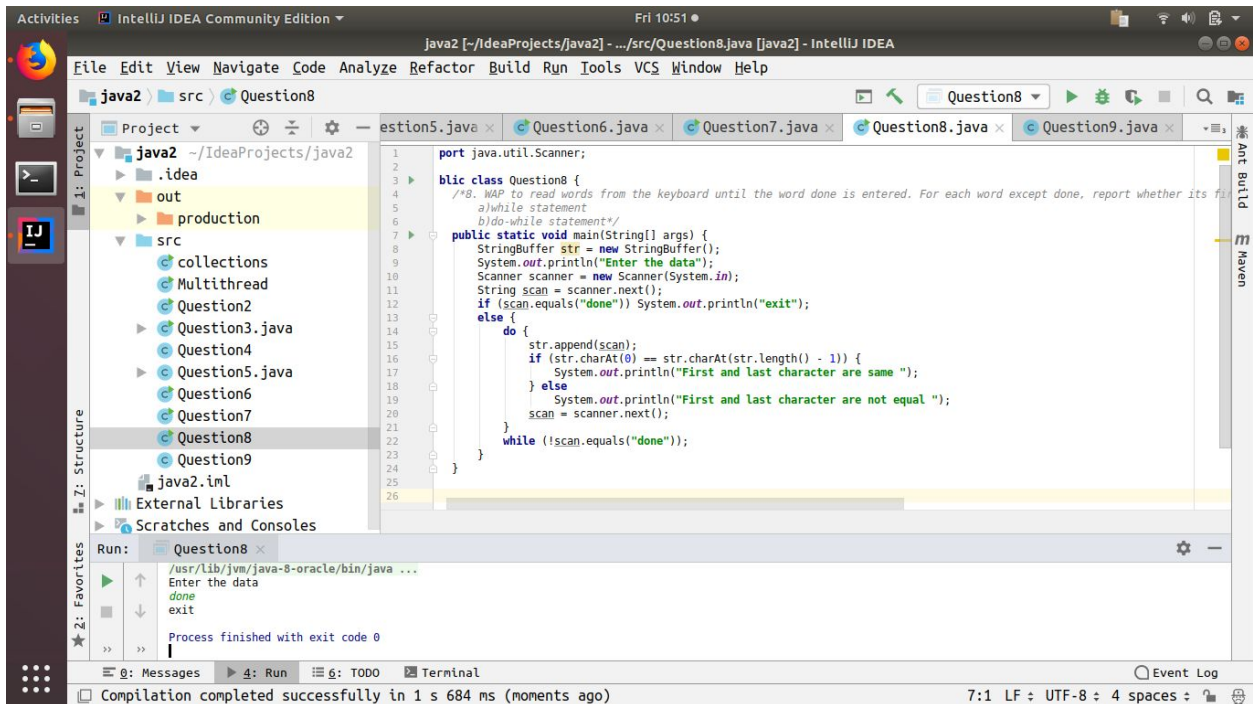
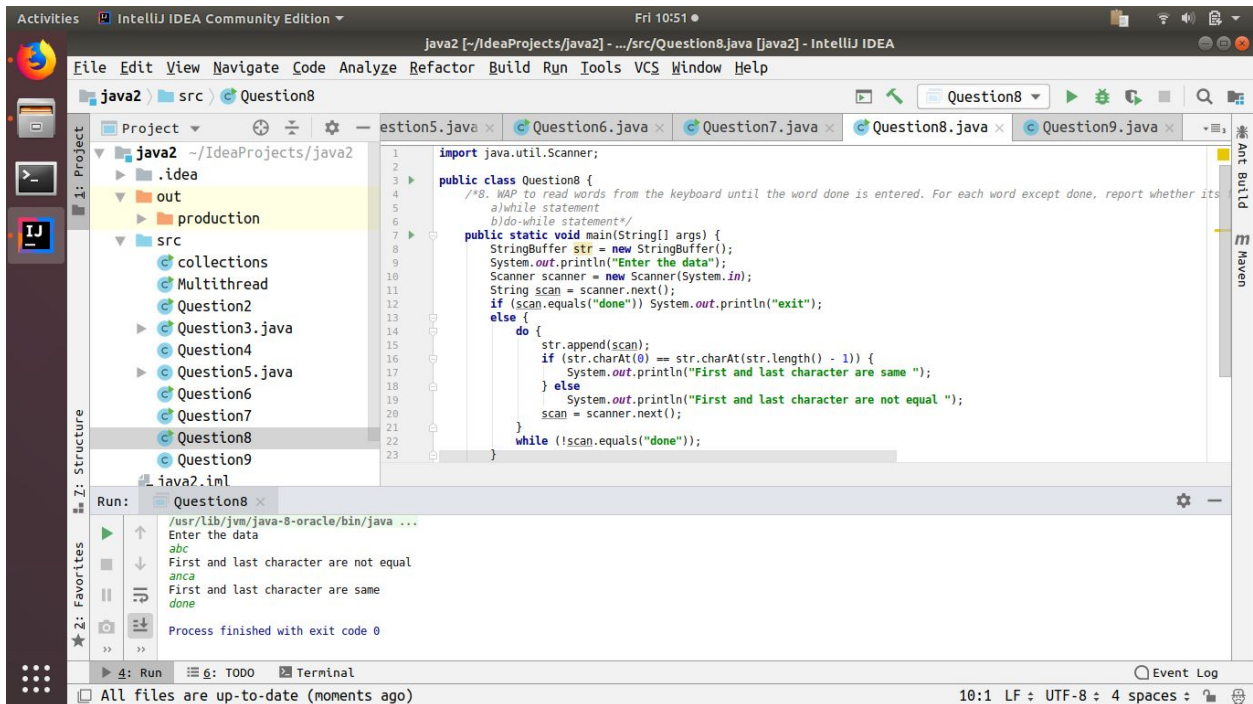
7. WAP to convert seconds into days, hours, minutes and seconds.



8. WAP to read words from the keyboard until the word done is entered. For each word except done, report whether its first character is equal to its last character. For the required loop, use a
a)while statement



b)do-while statement



9. Design classes having attributes for furniture where there are wooden chairs and tables, metal chairs and tables. There are stress and fire tests for each products.

Activities IntelliJ IDEA Community Edition Fri 12:06

Java2 [-/IdeaProjects/java2] - .../src/Furniture.java [java2] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

java2 src Furniture

Table.java Chair.java Furniture.java

```
1 import java.util.Scanner;
2
3 public class Furniture {
4     public static void main(String[] args) {
5         System.out.println("Enter the type of furniture - metal/wooden");
6         Scanner scanner=new Scanner(System.in);
7         String property=scanner.next();
8         Chair chair=new Chair(property);
9         Table table=new Table(property);
10        System.out.println("Enter the furniture type and test you want to perform -");
11        System.out.println("for chair - stress test - 1");
12        System.out.println("for chair - fire test - 2");
13        System.out.println("for table - stress test - 3");
14        System.out.println("for table - fire test - 4");
15        int choice=scanner.nextInt();
16        switch (choice){
17            case 1:{
18                chair.StressTest();
19                break;
20            }
21            case 2:{
22                chair.FireTest();
23                break;
24            }
25            case 3:{
26                table.StressTest();
27                break;
28            }
29            case 4:{
30                table.FireTest();
31                break;
32            }
33            default:
34            {
35                System.out.println("No choice entered ");
36            }
37        }
38    }
39 }
```

Furniture > main()

6: TODO Terminal Event Log

14:28 LF UTF-8 4 spaces

Activities IntelliJ IDEA Community Edition Fri 12:06

Java2 [-/IdeaProjects/java2] - .../src/Chair.java [java2] - IntelliJ IDEA

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

java2 src Chair

Table.java Chair.java Furniture.java

[java2] ~/IdeaProjects/java2/src/Chair.java

```
1 //table class having property - metal/wooden and methods - stress test and fire test
2
3 public class Chair {
4     String property;
5     public Chair(String property) { this.property = property; }
6     public void StressTest() {
7         System.out.println("Enter the load applied ");
8         Scanner scanner = new Scanner(System.in);
9         int load = scanner.nextInt();
10        if (property.equals("wooden")) {
11            if (load < 100) {
12                System.out.println("Stress test passed ");
13            } else {
14                System.out.println("Stress test failed"); }
15        }
16        if (property.equals("metal")) {
17            if (load < 200) {
18                System.out.println("Stress test passed ");
19            } else {
20                System.out.println("Stress test failed"); } }
21    }
22    public void FireTest() {
23        System.out.println("Enter the time for which furniture is burned in minutes");
24        Scanner scanner = new Scanner(System.in);
25        int time = scanner.nextInt();
26        if (property.equals("wooden")) {
27            if (time < 10) {
28                System.out.println("Stress test passed ");
29            } else {
30                System.out.println("Stress test failed");
31            }
32        }
33        if (property.equals("metal")) {
34            if (time < 20) {
35                System.out.println("Stress test passed ");
36            } else {
37                System.out.println("Stress test failed"); } }
38    }
39 }
```

6: TODO Terminal Event Log

37:1 LF UTF-8 4 spaces


```

1 import java.util.Scanner;
2 public class Table {
3     //chair class having property - metal/wooden and methods - stress test and fire test
4     String property;
5     public Table(String property) { this.property = property; }
6     public void StressTest(){
7         System.out.println("Enter the load applied ");
8         Scanner scanner=new Scanner(System.in);
9         int load=scanner.nextInt();
10        if(property.equals("wooden")){
11            if(load<200)
12                System.out.println("Stress test passed ");
13            else
14                System.out.println("Stress test failed"); }
15        if(property.equals("metal")){
16            if(load<400)
17                System.out.println("Stress test passed ");
18            else
19                System.out.println("Stress test failed"); } }
20    public void FireTest(){
21        System.out.println("Enter the time for which furniture is burned in minutes");
22        Scanner scanner=new Scanner(System.in);
23        int time=scanner.nextInt();
24        if(property.equals("wooden")){
25            if(time<15)
26                System.out.println("Stress test passed ");
27            else
28                System.out.println("Stress test failed"); }
29        if(property.equals("metal")){
30            if(time<30)
31                System.out.println("Stress test passed ");
32            else
33                System.out.println("Stress test failed"); } } }
34
35
36

```

```

Run: Furniture
/usr/lib/jvm/java-8-oracle/bin/java ...
Enter the type of furniture - metal/wooden
metal
Enter the furniture type and test you want to perform -
for chair - stress test - 1
for chair - fire test - 2
for table - stress test - 3
for table - fire test - 4
2
Enter the time for which furniture is burned in minutes
10
Stress test passed
Process finished with exit code 0

```

10. Design classes having attributes and method(only skeleton) for a coffee shop. There are three different actors in our scenario and i have listed the different actions they do also below

* Customer

- Pays the cash to the cashier and places his order, get a token number back
- Waits for the intimation that order for his token is ready
- Upon intimation/notification he collects the coffee and enjoys his drink

(Assumption: Customer waits till the coffee is done, he wont timeout and cancel the order. Customer always likes the drink served. Exceptions like he not liking his coffee, he getting wrong coffee are not considered to keep the design simple.)

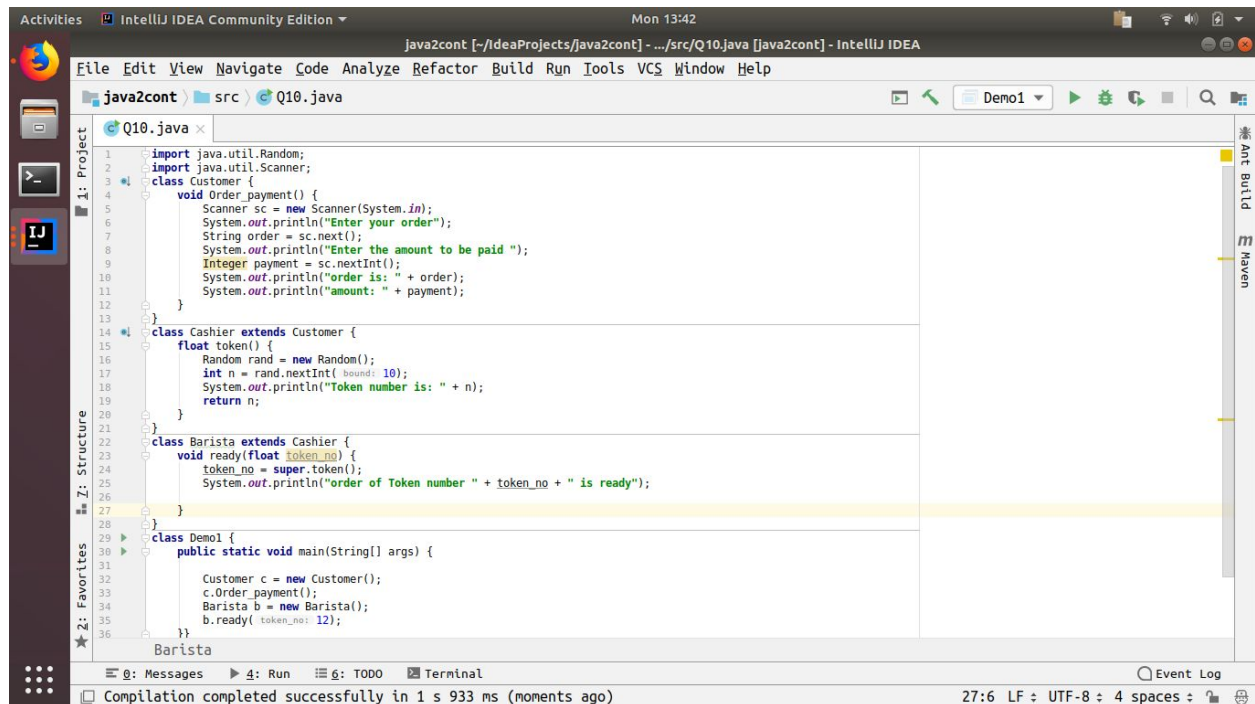
* Cashier

- Takes an order and payment from the customer
- Upon payment, creates an order and places it into the order queue
- Intimates the customer that he has to wait for his token and gives him his token

(Assumption: Token returned to the customer is the order id. Order queue is unlimited. With a simple modification, we can design for a limited queue size)

* Barista

- Gets the next order from the queue
- Prepares the coffee
- Places the coffee in the completed order queue
- Places a notification that order for token is ready



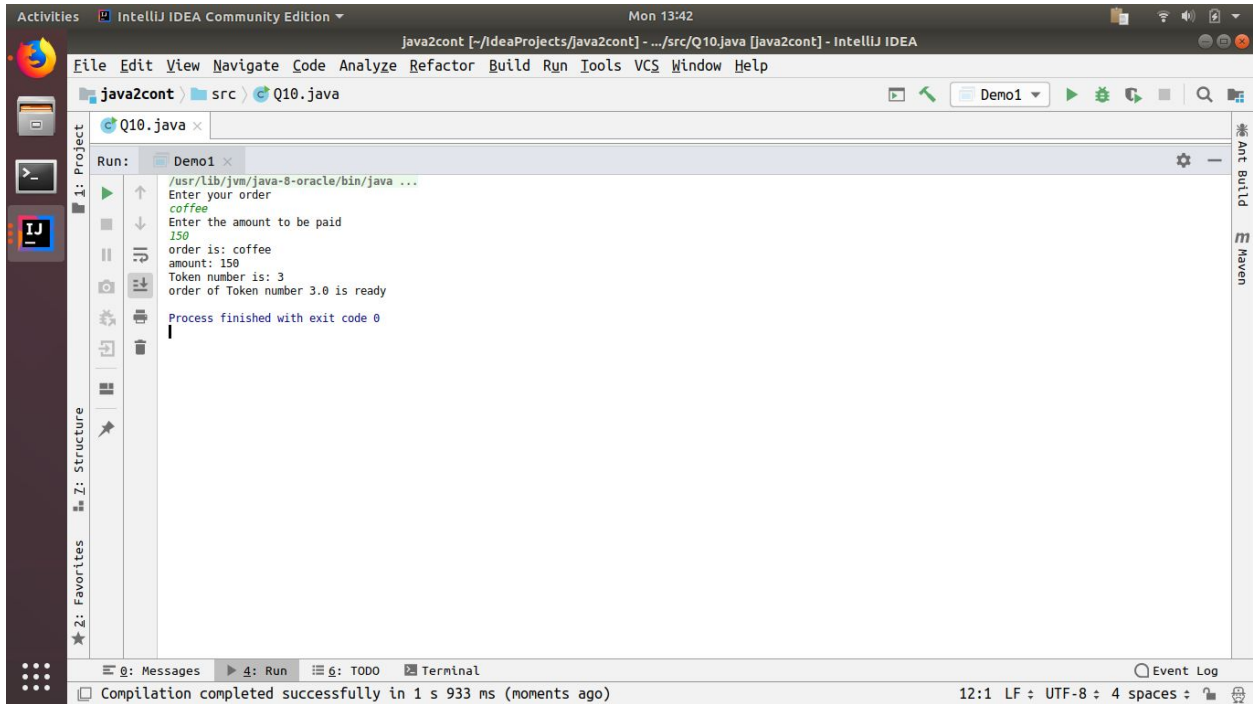
The screenshot shows the IntelliJ IDEA Community Edition interface. The main editor window displays the file `Q10.java` with the following Java code:

```
1 import java.util.Random;
2 import java.util.Scanner;
3 class Customer {
4     void Order_payment() {
5         Scanner sc = new Scanner(System.in);
6         System.out.println("Enter your order");
7         String order = sc.next();
8         System.out.println("Enter the amount to be paid ");
9         Integer payment = sc.nextInt();
10        System.out.println("order is: " + order);
11        System.out.println("amount: " + payment);
12    }
13 }
14 class Cashier extends Customer {
15     float token() {
16         Random rand = new Random();
17         int n = rand.nextInt( bound: 10);
18         System.out.println("Token number is: " + n);
19         return n;
20     }
21 }
22 class Barista extends Cashier {
23     void ready(float token_no) {
24         token_no = super.token();
25         System.out.println("order of Token number " + token_no + " is ready");
26     }
27 }
28 }
29 class Demo1 {
30     public static void main(String[] args) {
31
32         Customer c = new Customer();
33         c.Order_payment();
34         Barista b = new Barista();
35         b.ready( token_no: 12);
36     }
37 }
```

The left sidebar shows the Project view with the file structure:

- java2cont
- src
- Q10.java

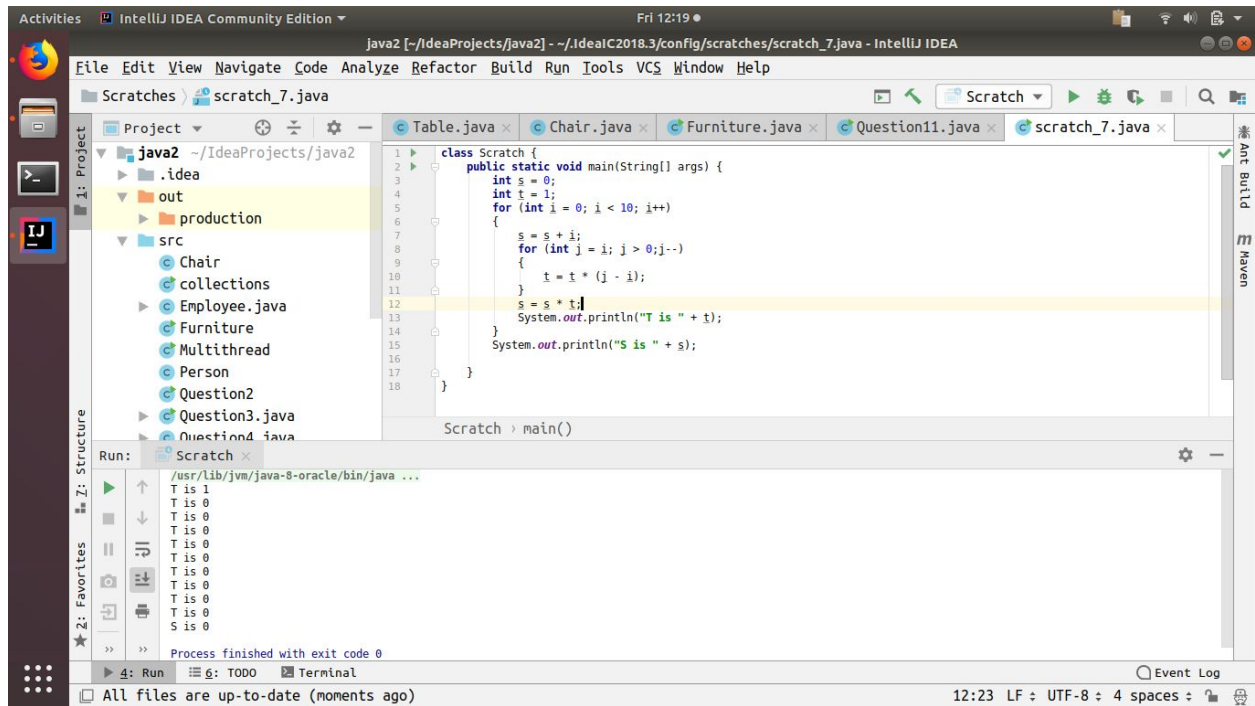
The bottom status bar indicates: "Compilation completed successfully in 1 s 933 ms (moments ago)". The bottom right corner shows: "27:6 LF : UTF-8 : 4 spaces :".



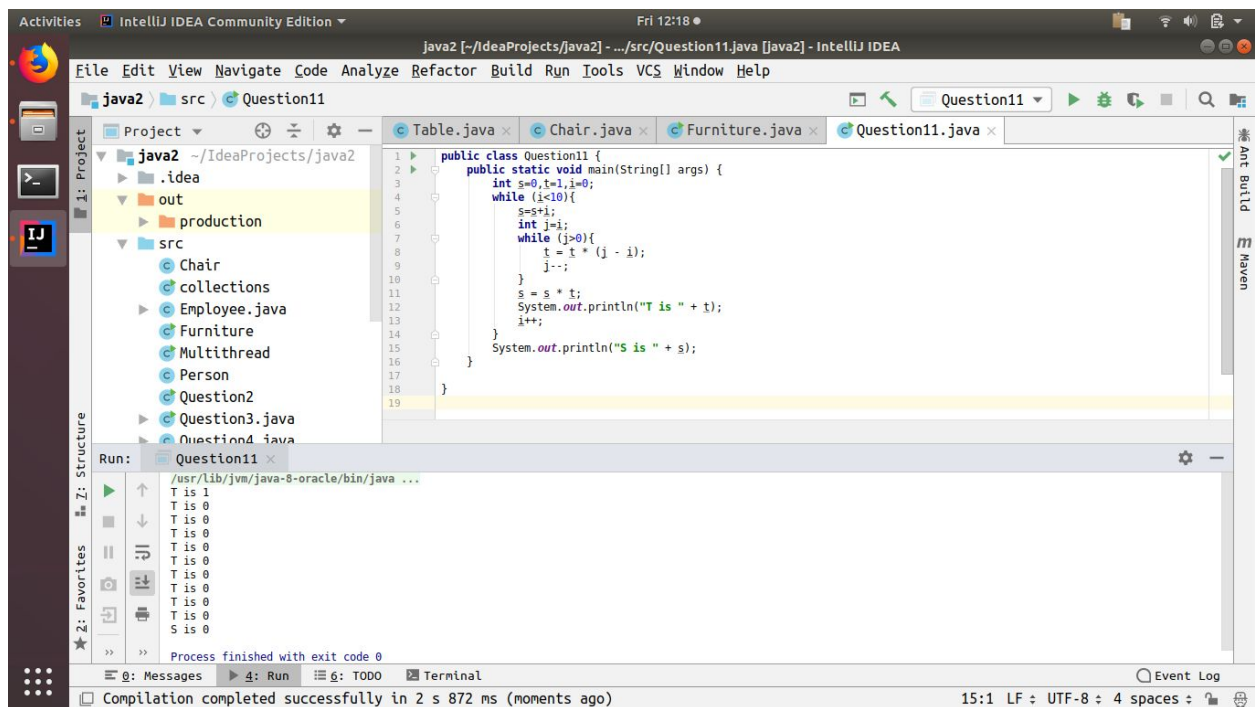
11. Convert the following code so that it uses nested while statements instead of for statements:

```
int s = 0;
int t = 1;
for (int i = 0; i < 10; i++)
{
    s = s + i;
    for (int j = i; j > 0; j--)
    {
        t = t * (j - i);
    }
    s = s * t;
    System.out.println("T is " + t);
}
System.out.println("S is " + s);
```

Output for above for loop :



Using while loop :



12.What will be the output on new Child(); ?

class Parent extends Grandparent {

```

{
    System.out.println("instance - parent");
}

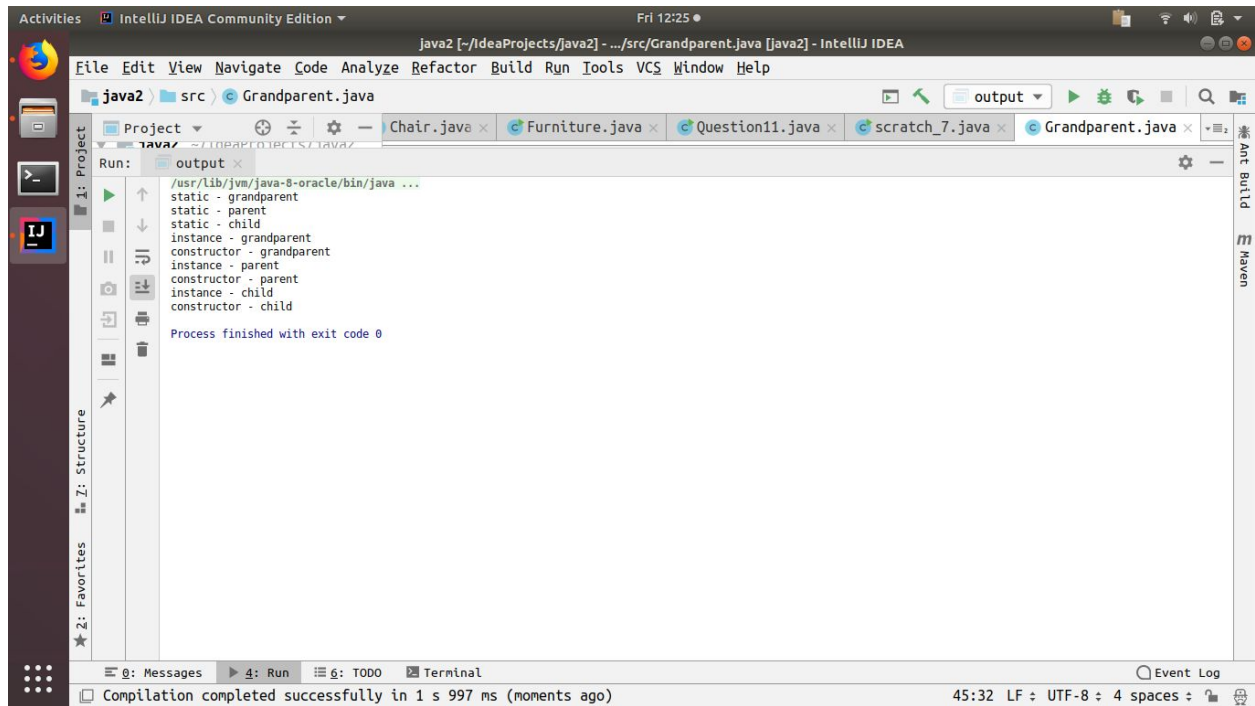
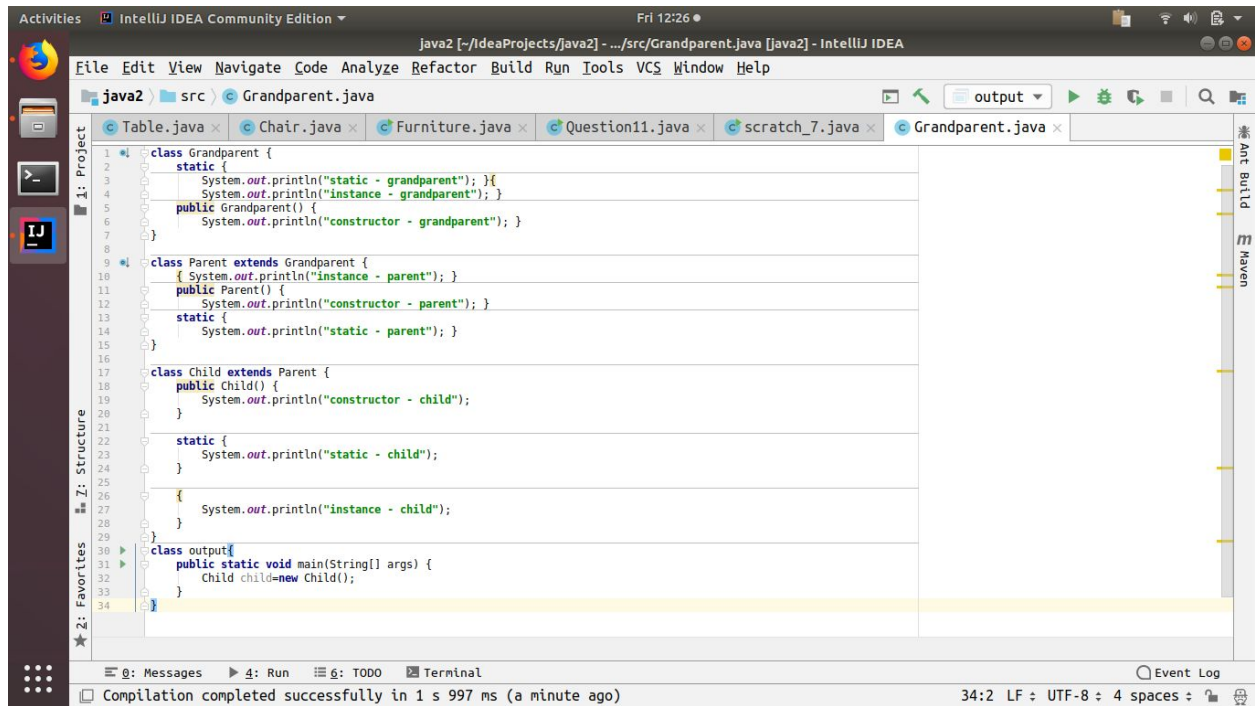
```

```

    }
    public Parent() {
        System.out.println("constructor - parent");
    }
    static {
        System.out.println("static - parent");
    }
}
class Grandparent {

    static {
        System.out.println("static - grandparent");
    }
    {
        System.out.println("instance - grandparent");
    }
    public Grandparent() {
        System.out.println("constructor - grandparent");
    }
}
class Child extends Parent {
    public Child() {
        System.out.println("constructor - child");
    }
    static {
        System.out.println("static - child");
    }
    {
        System.out.println("instance - child");
    }
}

```



Q13. Create a custom exception that do not have any stack trace.

