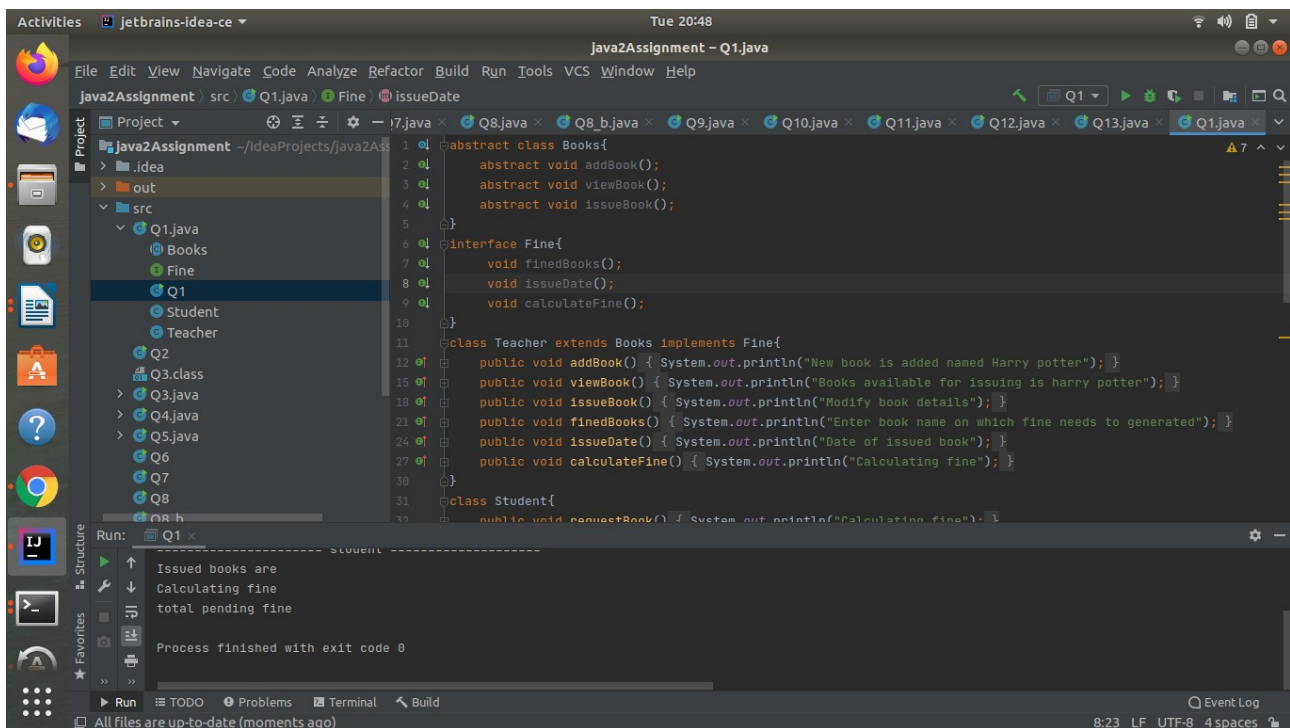


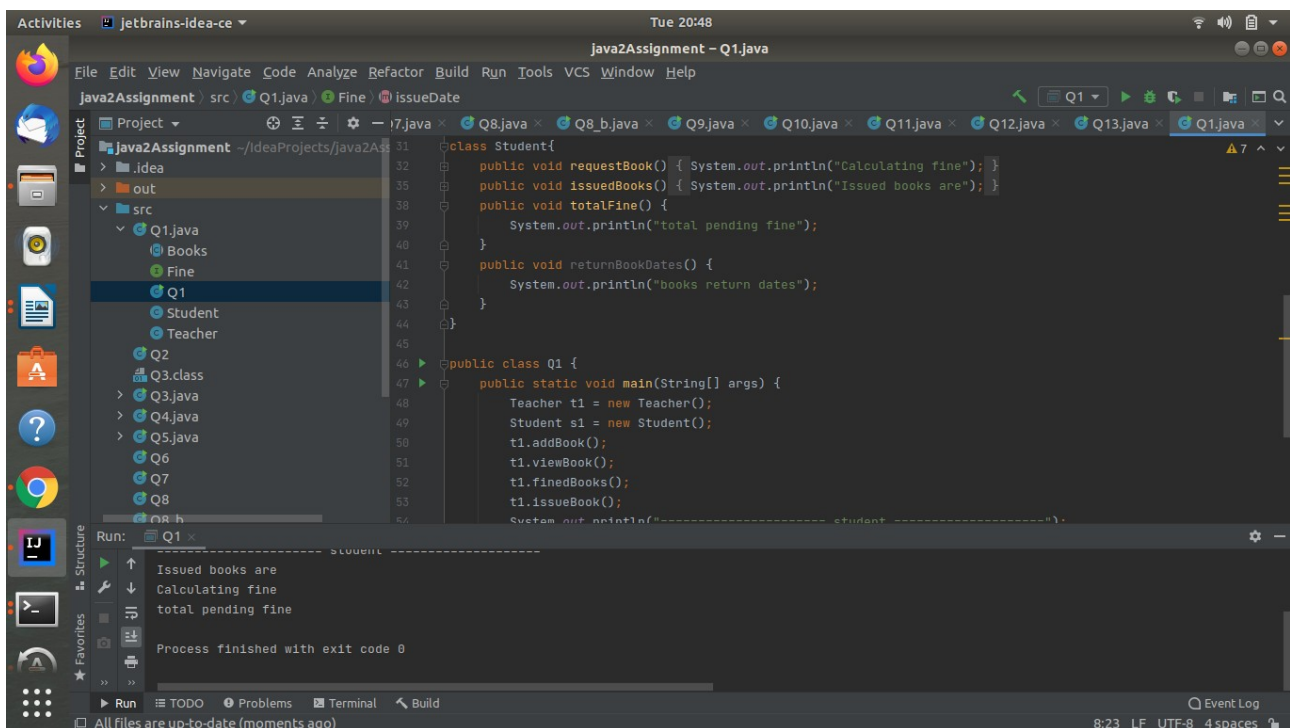
Q1: Create Java classes having suitable attributes for Library management system. Use OOPs concepts in your design. Also try to use interfaces and abstract classes.



```
1 abstract class Books {
2     abstract void addBook();
3     abstract void viewBook();
4     abstract void issueBook();
5 }
6 interface Fine {
7     void finedBooks();
8     void issueDate();
9     void calculateFine();
10 }
11 class Teacher extends Books implements Fine {
12     public void addBook() { System.out.println("New book is added named Harry potter"); }
13     public void viewBook() { System.out.println("Books available for issuing is harry potter"); }
14     public void issueBook() { System.out.println("Modify book details"); }
15     public void finedBooks() { System.out.println("Enter book name on which fine needs to generated"); }
16     public void issueDate() { System.out.println("Date of issued book"); }
17     public void calculateFine() { System.out.println("Calculating fine"); }
18 }
19 class Student {
20     public void requestBook() { System.out.println("Calculating fine"); }
21 }
```

Run: Q1 x

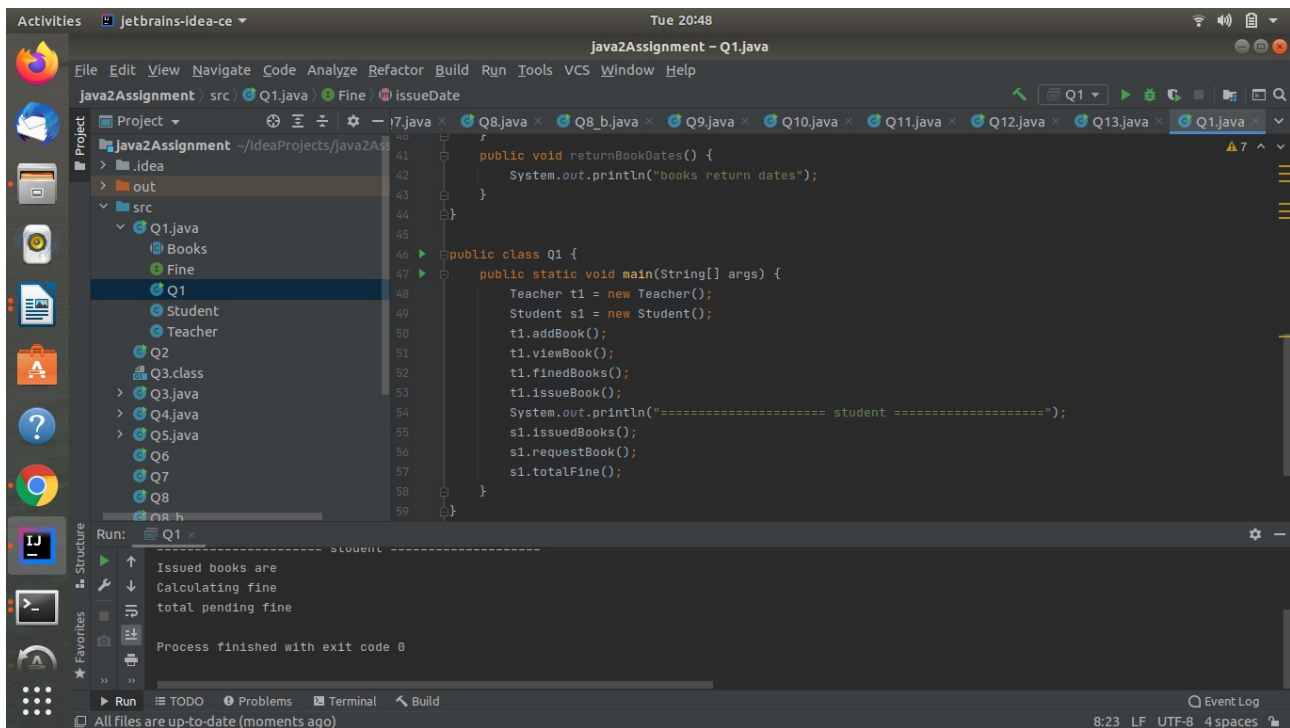
```
----- student -----
Issued books are
Calculating fine
total pending fine
Process finished with exit code 0
```



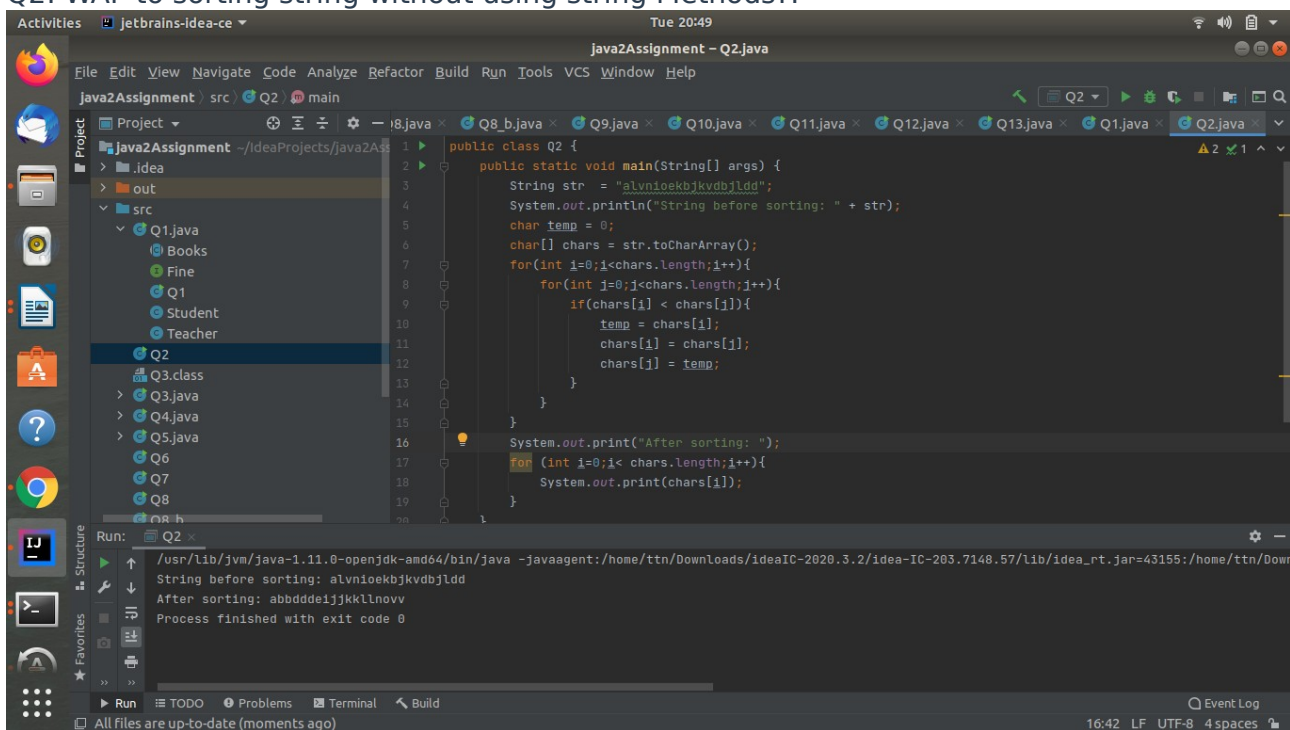
```
31 class Student {
32     public void requestBook() { System.out.println("Calculating fine"); }
33     public void issuedBooks() { System.out.println("Issued books are"); }
34     public void totalFine() {
35         System.out.println("total pending fine");
36     }
37     public void returnBookDates() {
38         System.out.println("books return dates");
39     }
40 }
41 public class Q1 {
42     public static void main(String[] args) {
43         Teacher t1 = new Teacher();
44         Student s1 = new Student();
45         t1.addBook();
46         t1.viewBook();
47         t1.finedBooks();
48         t1.issueBook();
49         System.out.println("----- student -----");
50     }
51 }
```

Run: Q1 x

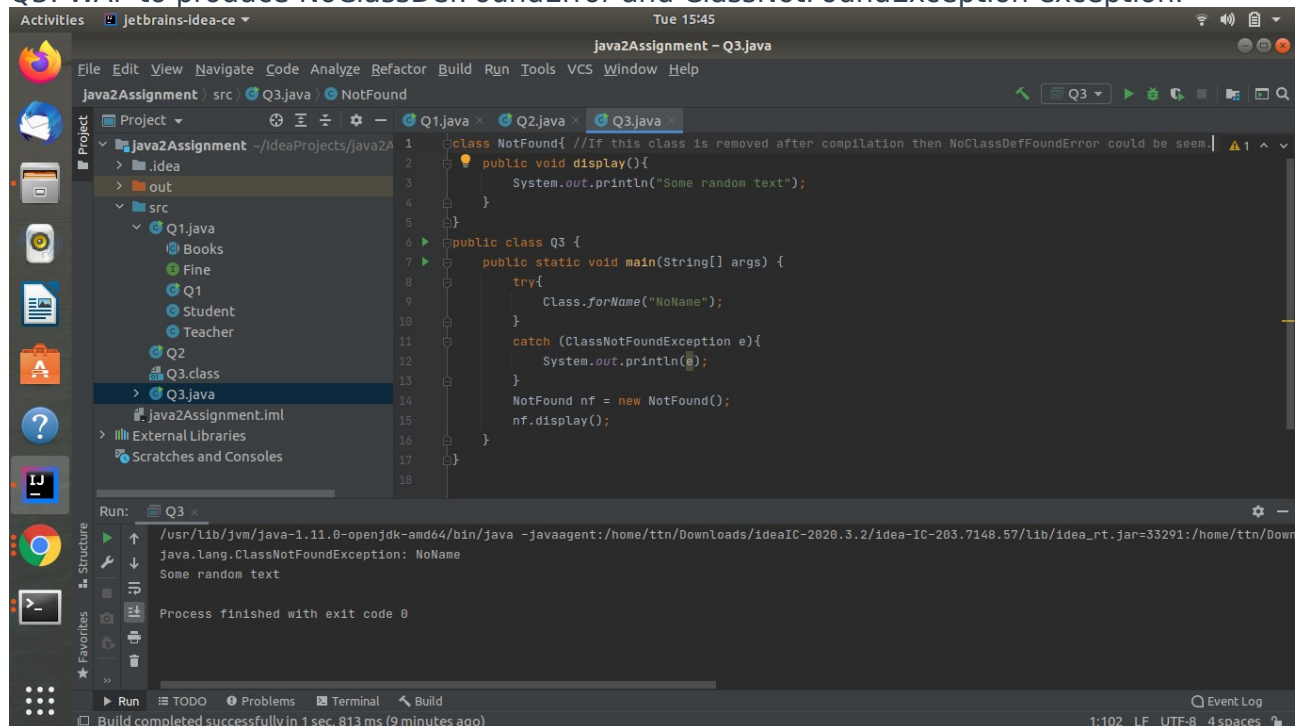
```
----- student -----
Issued books are
Calculating fine
total pending fine
Process finished with exit code 0
```



Q2: WAP to sorting string without using string Methods?.



Q3: WAP to produce NoClassDefFoundError and ClassNotFoundException exception.



The screenshot shows the IntelliJ IDEA IDE with the following content:

Project Structure:

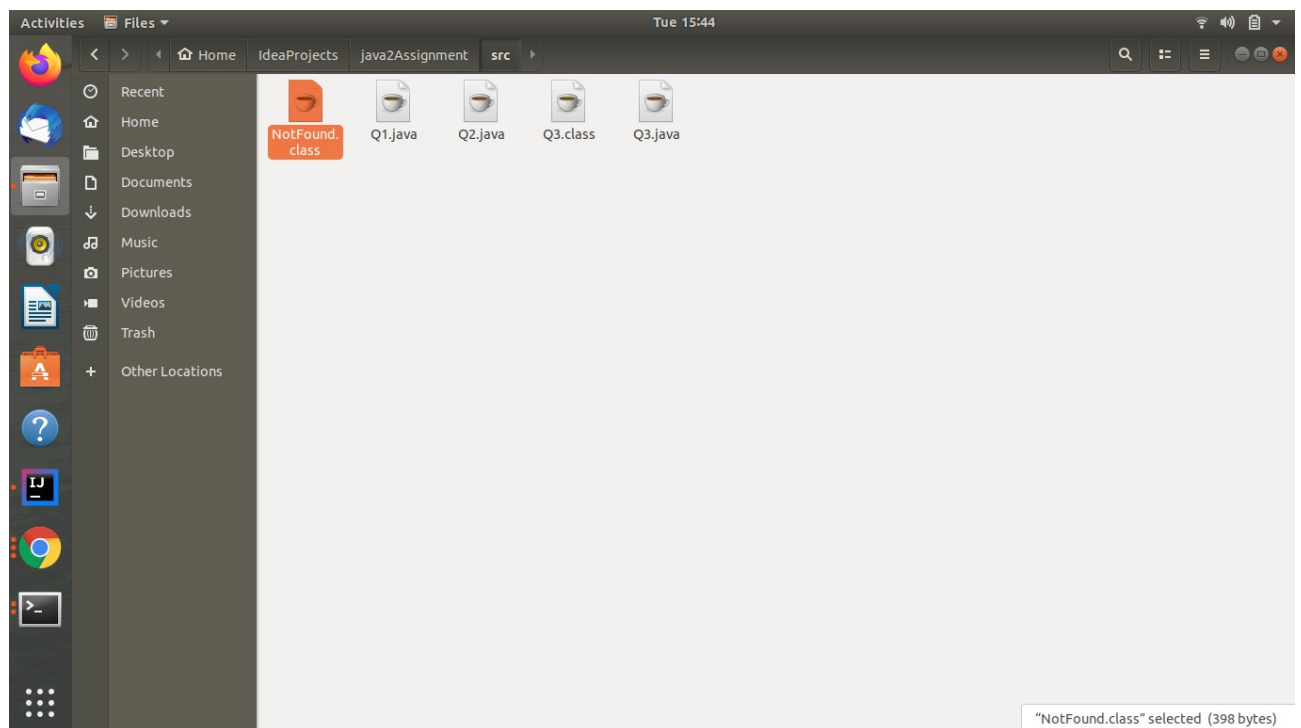
- java2Assignment / src
 - Q1.java
 - Books
 - Fine
 - Q1
 - Student
 - Teacher
 - Q2
 - Q3.class
 - Q3.java
- java2Assignment.iml
- External Libraries
- Scratches and Consoles

Q3.java Code:

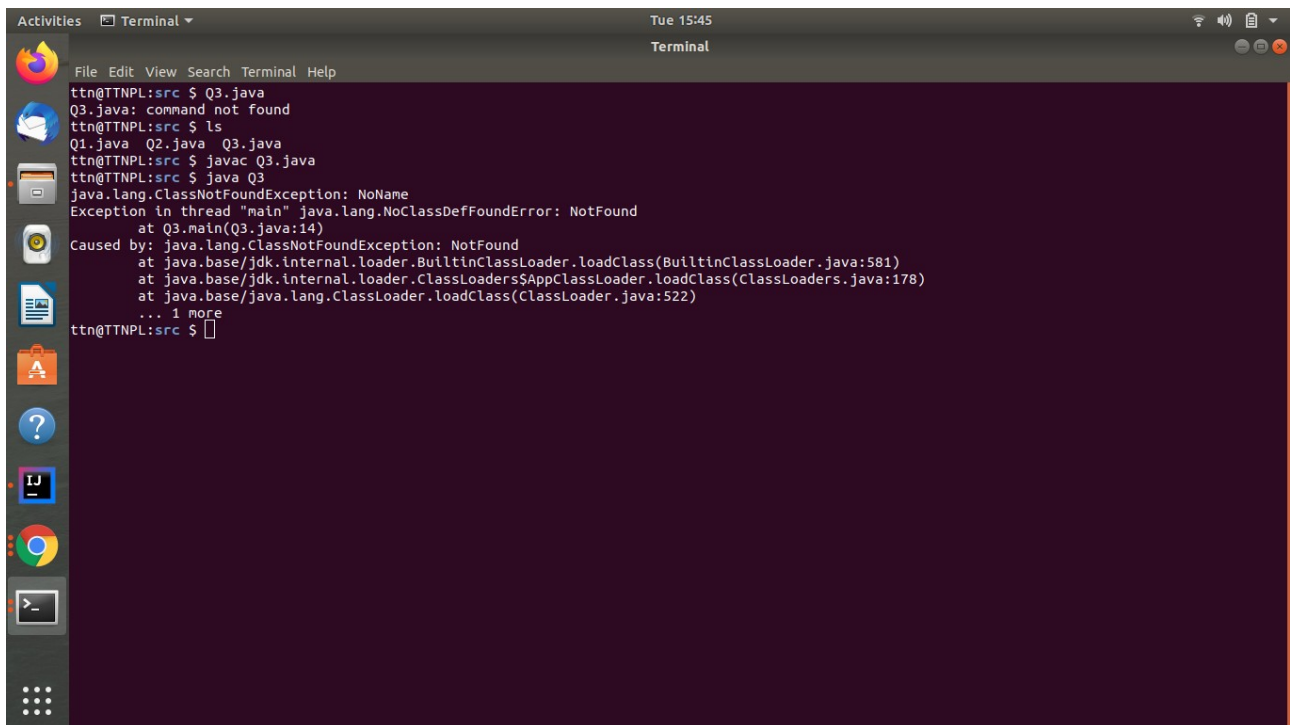
```
1 class NotFound{ //If this class is removed after compilation then NoClassDefFoundError could be seen
2     public void display(){
3         System.out.println("Some random text");
4     }
5 }
6
7 public class Q3 {
8     public static void main(String[] args) {
9         try{
10             Class.forName("NoName");
11         }
12         catch (ClassNotFoundException e){
13             System.out.println(e);
14         }
15         NotFound nf = new NotFound();
16         nf.display();
17     }
18 }
```

Run Output:

```
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/home/ttn/Downloads/ideaIC-2020.3.2/idea-IC-203.7148.57/lib/idea_rt.jar=33291:/home/ttn/Down
java.lang.ClassNotFoundException: NoName
Some random text
Process finished with exit code 0
```



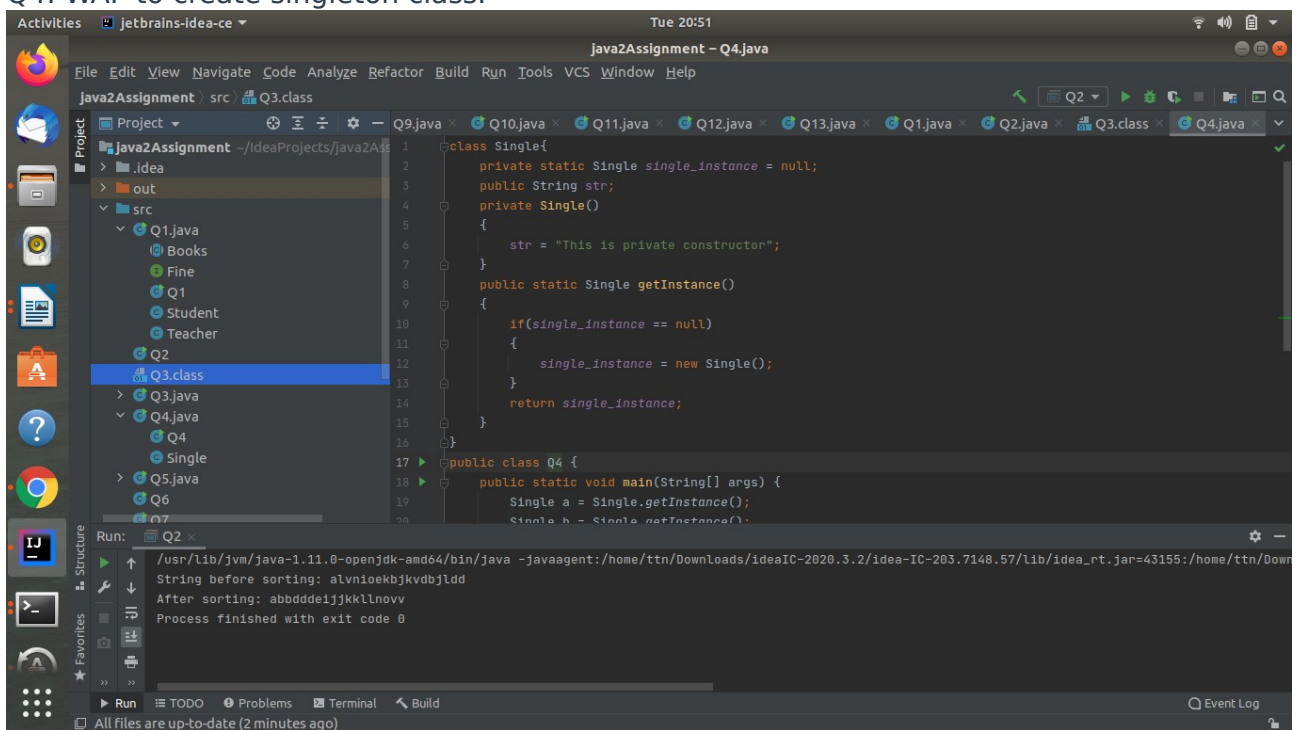
"NotFound.class" selected (398 bytes)



A terminal window titled "Terminal" showing a series of commands and their outputs. The user is in a directory named "src". The commands and outputs are as follows:

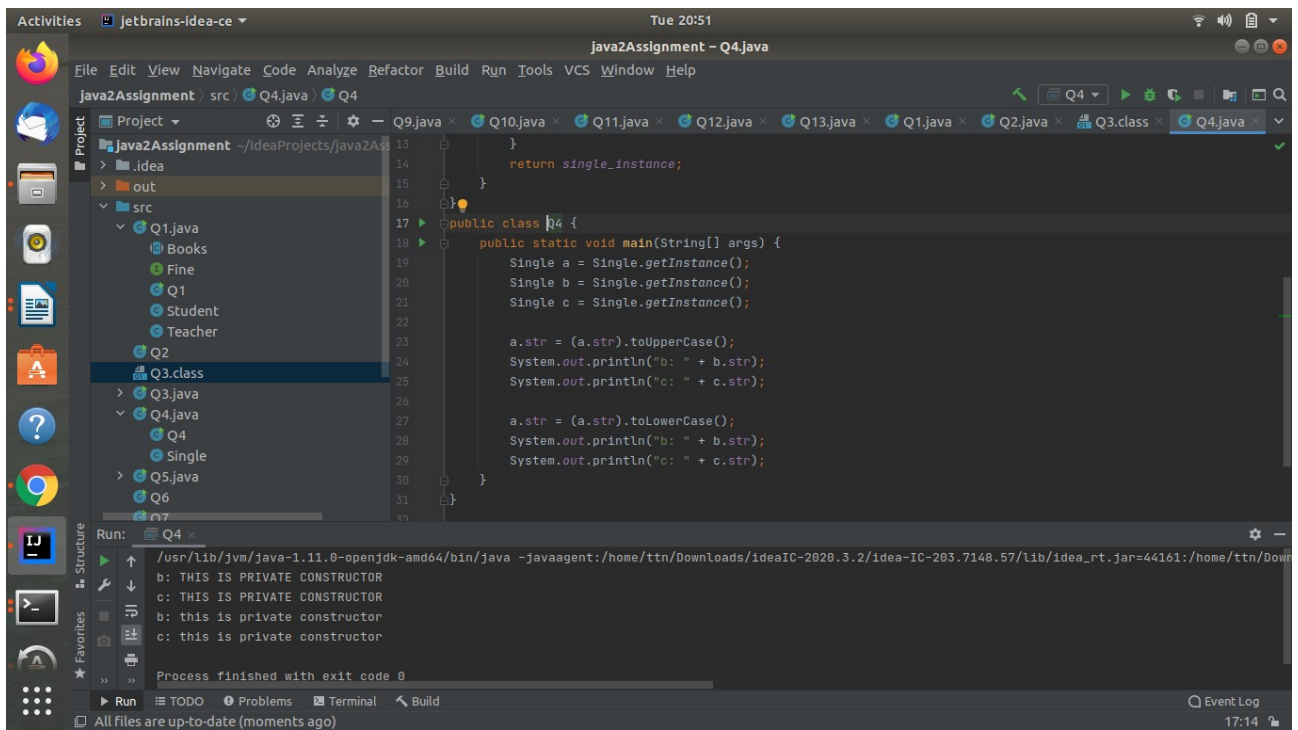
```
ttn@TTNPL:src $ Q3.java
Q3.java: command not found
ttn@TTNPL:src $ ls
Q1.java Q2.java Q3.java
ttn@TTNPL:src $ javac Q3.java
ttn@TTNPL:src $ java Q3
java.lang.ClassNotFoundException: NoName
Exception in thread "main" java.lang.NoClassDefFoundError: NotFound
    at Q3.main(Q3.java:14)
Caused by: java.lang.ClassNotFoundException: NotFound
    at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:581)
    at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(ClassLoaders.java:178)
    at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:522)
    ... 1 more
ttn@TTNPL:src $
```

Q4: WAP to create singleton class.

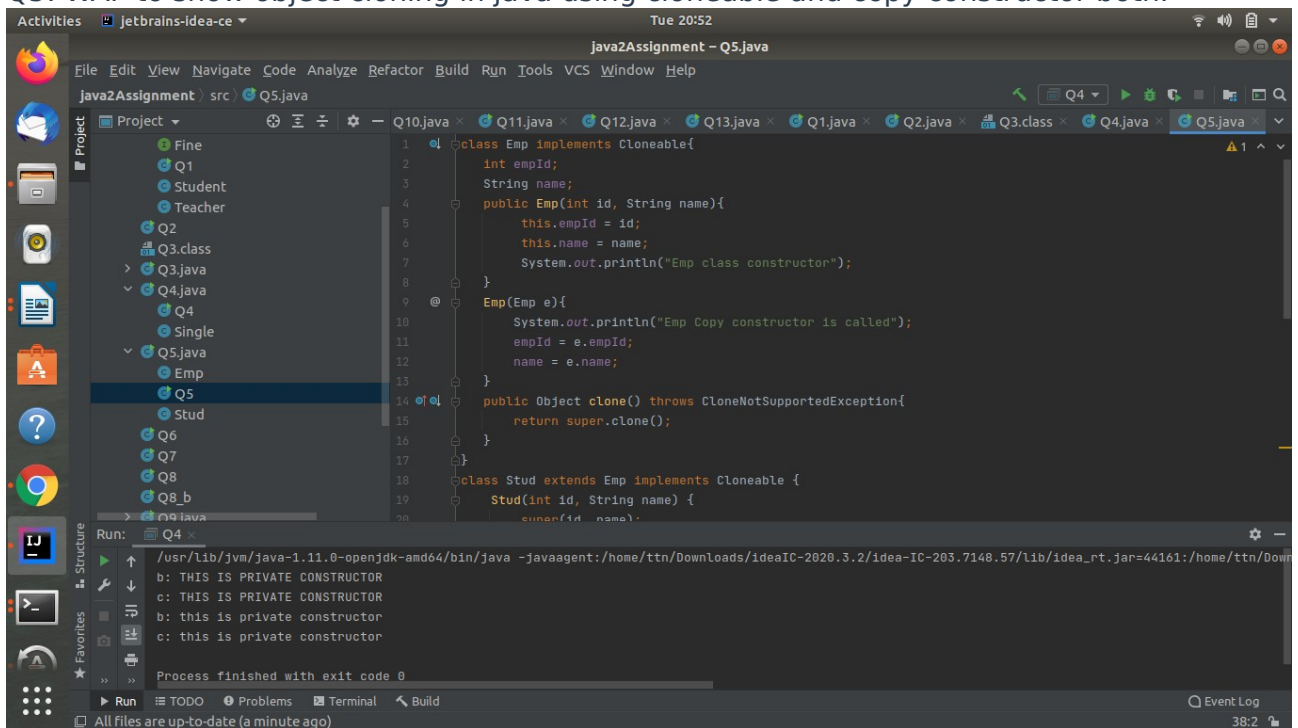


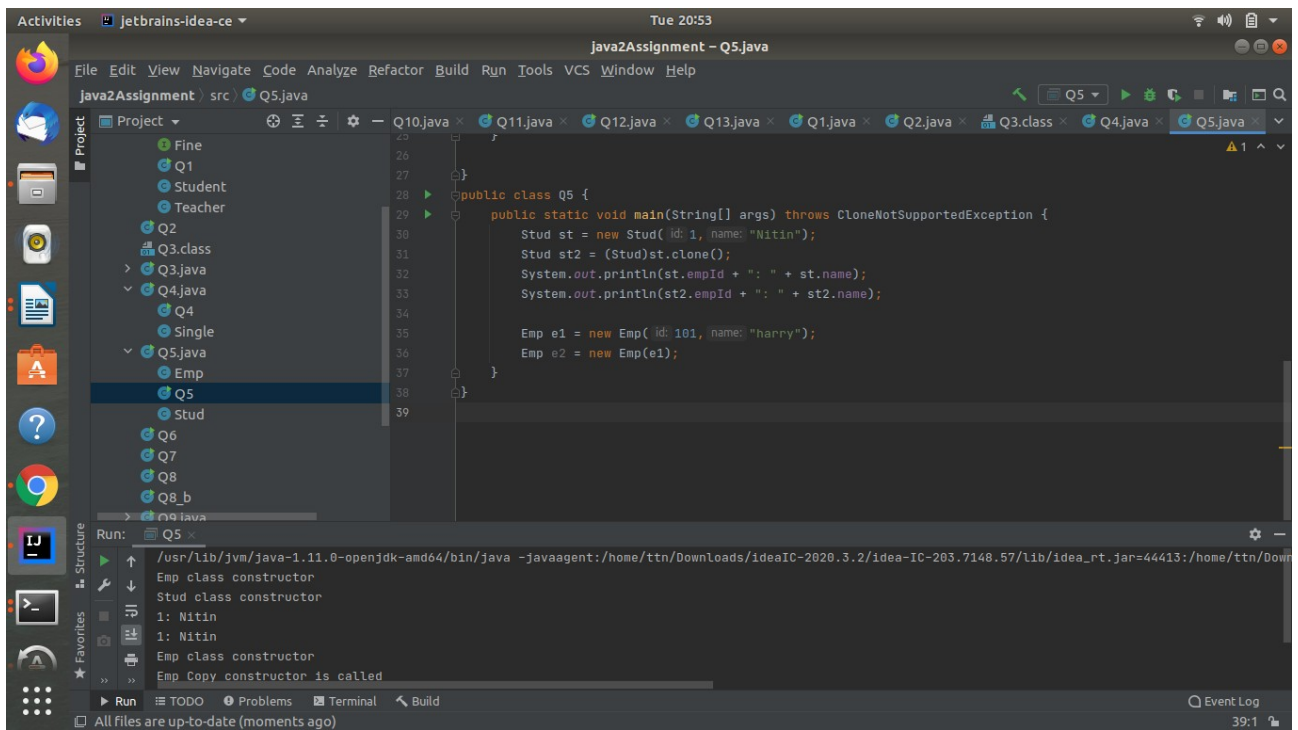
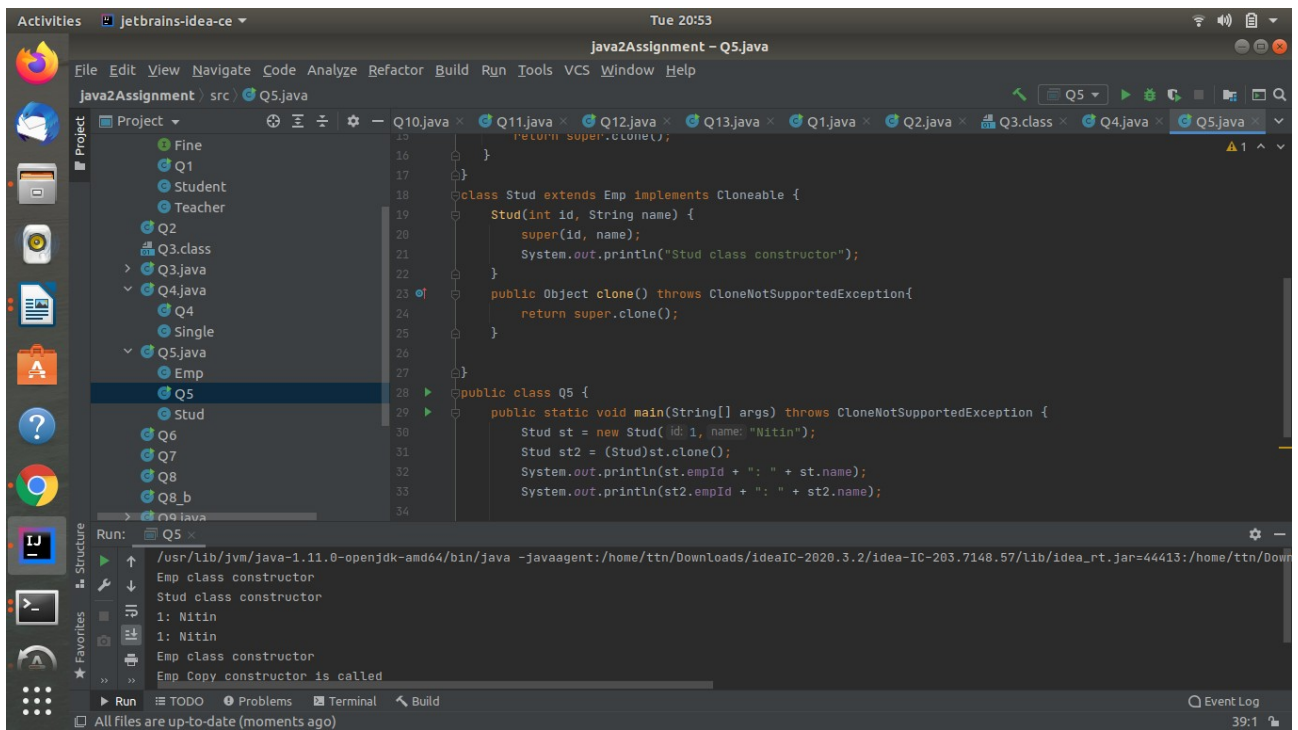
A screenshot of the JetBrains IDE (IntelliJ IDEA) showing a project named "Java2Assignment". The project structure on the left includes a "src" directory with files "Q1.java", "Q2", "Q3.class", "Q3.java", "Q4", "Q4.java", "Q5.java", "Q6", and "Q7". The main editor displays the code for "Q4.java", which defines a Singleton class and a test class Q4.

```
1 class Single{
2     private static Single single_instance = null;
3     public String str;
4     private Single()
5     {
6         str = "This is private constructor";
7     }
8     public static Single getInstance()
9     {
10        if(single_instance == null)
11        {
12            single_instance = new Single();
13        }
14        return single_instance;
15    }
16 }
17 public class Q4 {
18     public static void main(String[] args) {
19         Single a = Single.getInstance();
20         Single b = Single.getInstance();
```

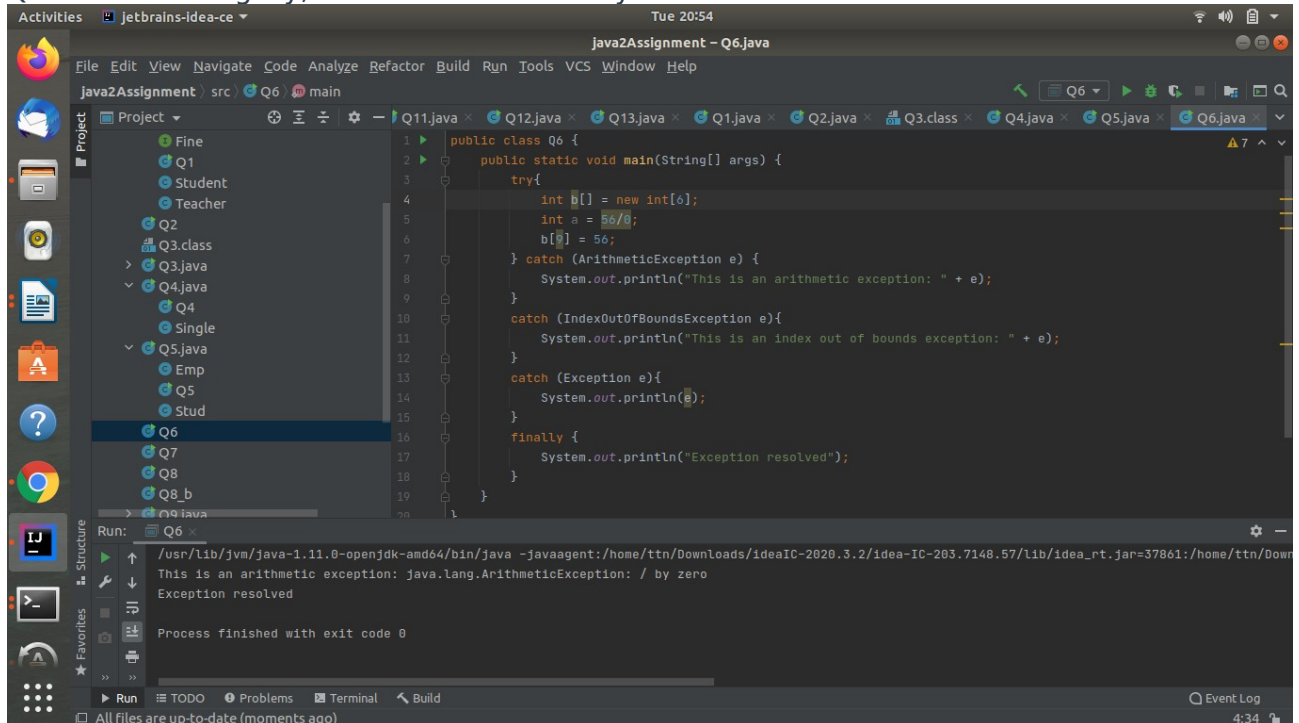



Q5: WAP to show object cloning in java using cloneable and copy constructor both.





Q6: WAP showing try, multi-catch and finally blocks.

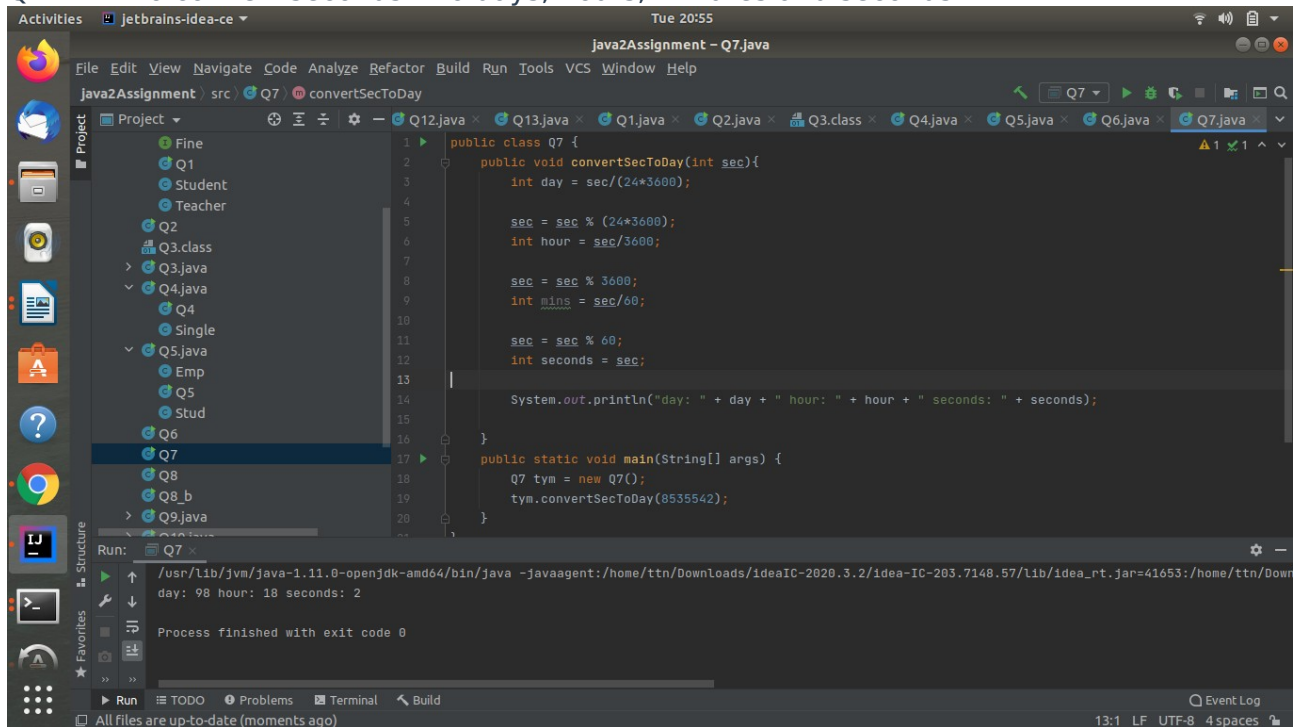


The screenshot shows the IntelliJ IDEA interface with the file `Q6.java` open. The code implements a try-catch-finally block to handle exceptions. The `try` block initializes an array `b` and attempts to divide `a` by `b[0]`. The `catch` blocks handle `ArithmeticException` and `IndexOutOfBoundsException`. The `finally` block prints a message indicating the exception is resolved. The Run window shows the output: "This is an arithmetic exception: java.lang.ArithmeticException: / by zero" followed by "Exception resolved".

```
1 public class Q6 {
2     public static void main(String[] args) {
3         try{
4             int b[] = new int[6];
5             int a = 56/0;
6             b[0] = 56;
7         } catch (ArithmeticException e) {
8             System.out.println("This is an arithmetic exception: " + e);
9         }
10        catch (IndexOutOfBoundsException e){
11            System.out.println("This is an index out of bounds exception: " + e);
12        }
13        catch (Exception e){
14            System.out.println(e);
15        }
16        finally {
17            System.out.println("Exception resolved");
18        }
19    }
20 }
```

Run: Q6 x
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/home/ttn/Downloads/ideaIC-2020.3.2/idea-IC-203.7148.57/lib/idea_rt.jar=37861:/home/ttn/Down
This is an arithmetic exception: java.lang.ArithmeticException: / by zero
Exception resolved
Process finished with exit code 0

Q7: WAP to convert seconds into days, hours, minutes and seconds.

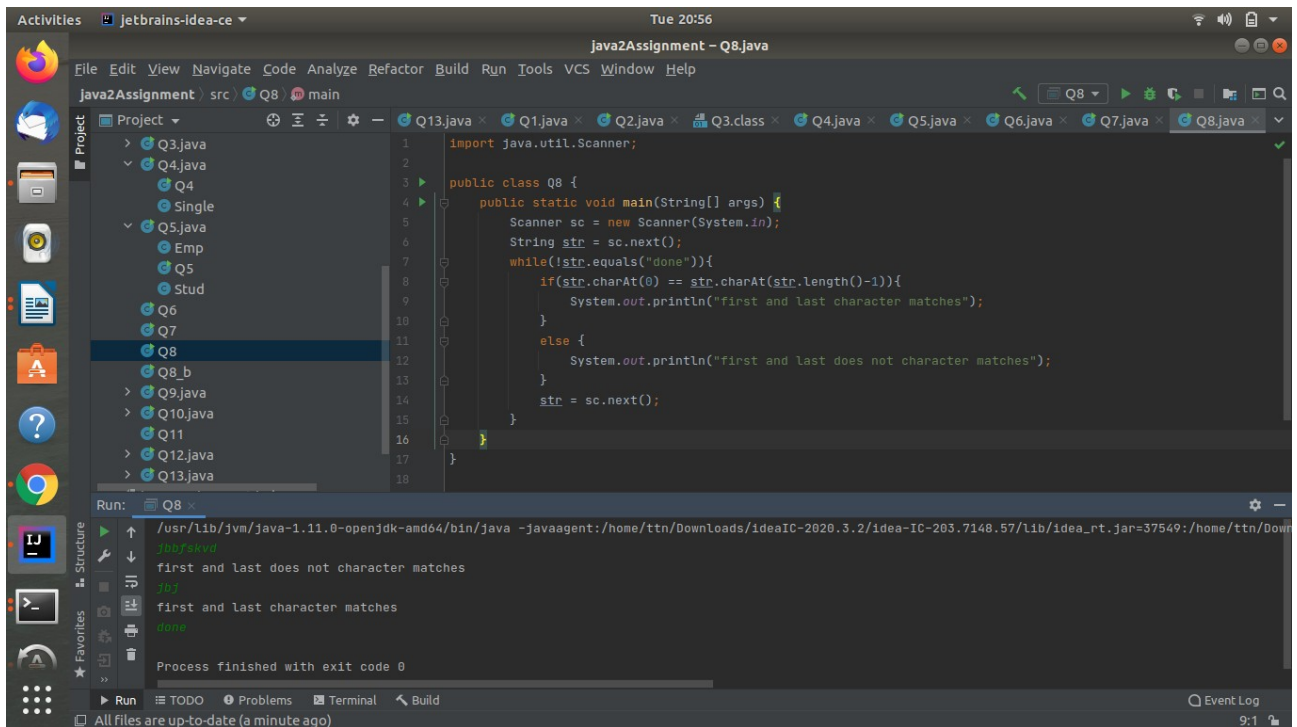


The screenshot shows the IntelliJ IDEA interface with the file `Q7.java` open. The code implements a `convertSecToDay` method that calculates days, hours, minutes, and seconds from a given number of seconds. The `main` method creates an instance of `Q7` and calls `convertSecToDay` with the value 8535542. The Run window shows the output: "day: 98 hour: 18 seconds: 2".

```
1 public class Q7 {
2     public void convertSecToDay(int sec){
3         int day = sec/(24*3600);
4
5         sec = sec % (24*3600);
6         int hour = sec/3600;
7
8         sec = sec % 3600;
9         int mins = sec/60;
10
11        sec = sec % 60;
12        int seconds = sec;
13
14        System.out.println("day: " + day + " hour: " + hour + " seconds: " + seconds);
15    }
16
17    public static void main(String[] args) {
18        Q7 tym = new Q7();
19        tym.convertSecToDay(8535542);
20    }
21 }
```

Run: Q7 x
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/home/ttn/Downloads/ideaIC-2020.3.2/idea-IC-203.7148.57/lib/idea_rt.jar=41653:/home/ttn/Down
day: 98 hour: 18 seconds: 2
Process finished with exit code 0

Q8: 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



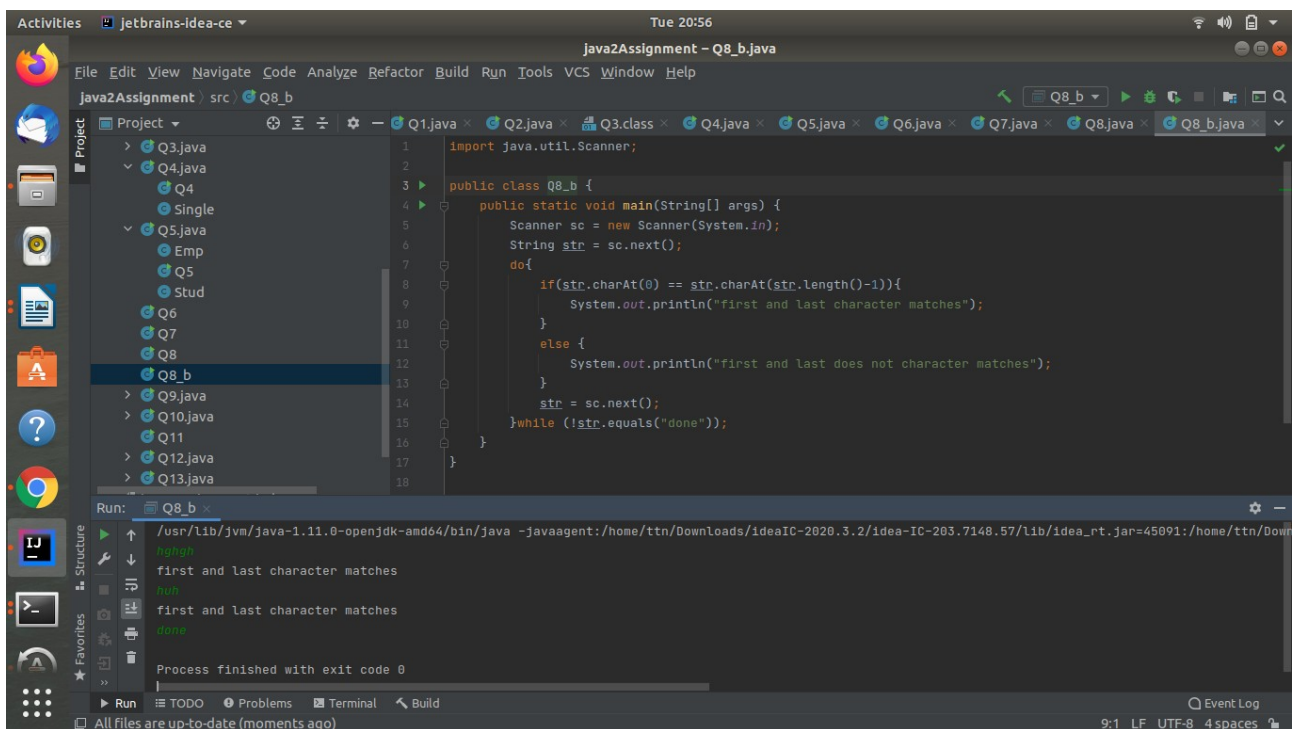
The screenshot shows the IntelliJ IDEA interface with the file 'Q8.java' open. The code implements a while loop to read words from the keyboard until 'done' is entered. For each word, it checks if the first character is equal to the last character and prints the result.

```
import java.util.Scanner;

public class Q8 {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str = sc.next();
        while(!str.equals("done")){
            if(str.charAt(0) == str.charAt(str.length()-1)){
                System.out.println("first and last character matches");
            }
            else {
                System.out.println("first and last does not character matches");
            }
            str = sc.next();
        }
    }
}
```

The Run window shows the output of the program:

```
first and last does not character matches
first and last character matches
done
Process finished with exit code 0
```



The screenshot shows the IntelliJ IDEA interface with the file 'Q8_b.java' open. The code implements a do-while loop to read words from the keyboard until 'done' is entered. For each word, it checks if the first character is equal to the last character and prints the result.

```
import java.util.Scanner;

public class Q8_b {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String str = sc.next();
        do{
            if(str.charAt(0) == str.charAt(str.length()-1)){
                System.out.println("first and last character matches");
            }
            else {
                System.out.println("first and last does not character matches");
            }
            str = sc.next();
        }while (!str.equals("done"));
    }
}
```

The Run window shows the output of the program:

```
first and last character matches
first and last character matches
done
Process finished with exit code 0
```


Q9: 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.

The screenshot shows the IntelliJ IDEA IDE with the 'java2Assignment - Q9.java' project. The Project view on the left shows the package structure: Q5.java, Emp, Q5, Stud, Q6, Q7, Q8, Q8_b, Q9.java, Chair, Furniture, MetalChair, MetalTable, Q9, Table, WoodenChair, and WoodenTable. The main editor displays the code for Q9.java, which defines the Furniture interface and its implementations, Chair and Table.

```
1 interface Furniture{
2     public void stressTest();
3     public void fireTest();
4 }
5 abstract class Chair implements Furniture{
6     abstract String chairType();
7 }
8 abstract class Table implements Furniture{
9     abstract String tableType();
10 }
11 class MetalChair extends Chair{
12     public void stressTest(){
13         System.out.println("Test passed");
14     }
15     public void fireTest(){
16         System.out.println("Test passed");
17     }
18     public String chairType() { return "MetalChair"; }
19 }
20
```

The Run view at the bottom shows the execution of the Q9 class, with the following output:

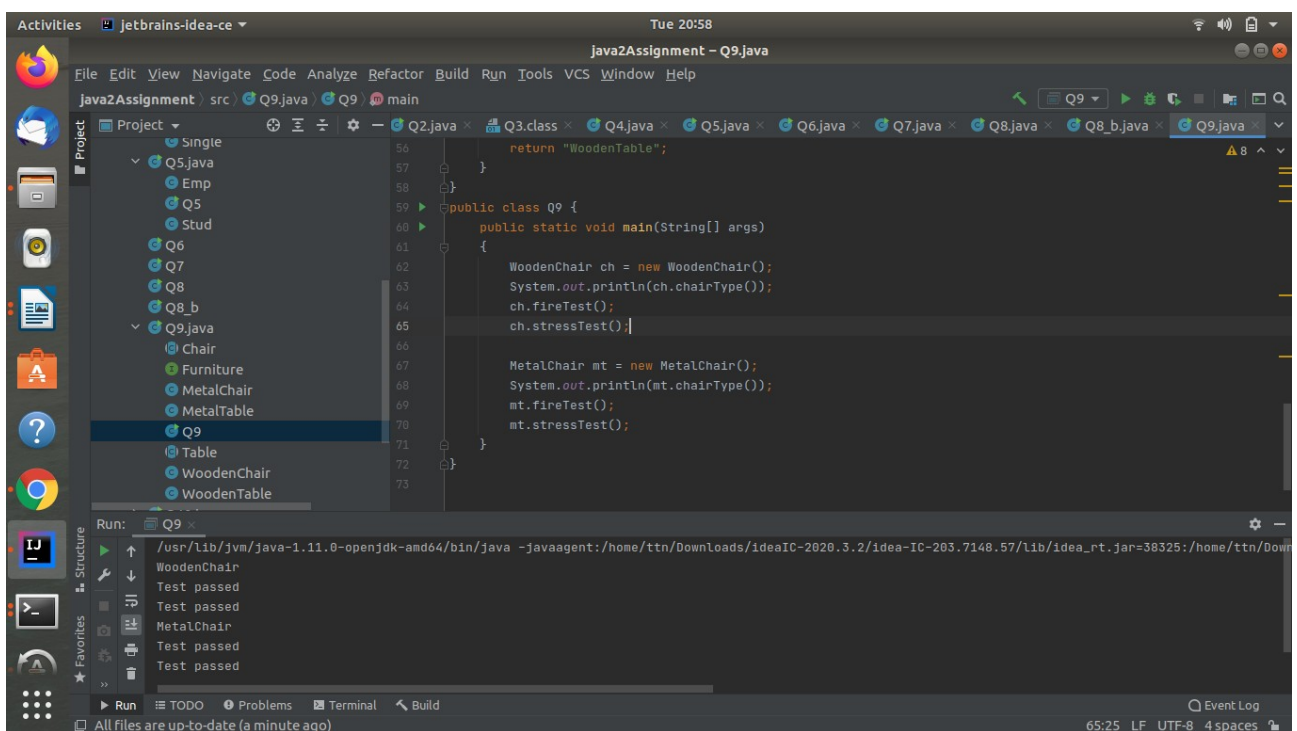
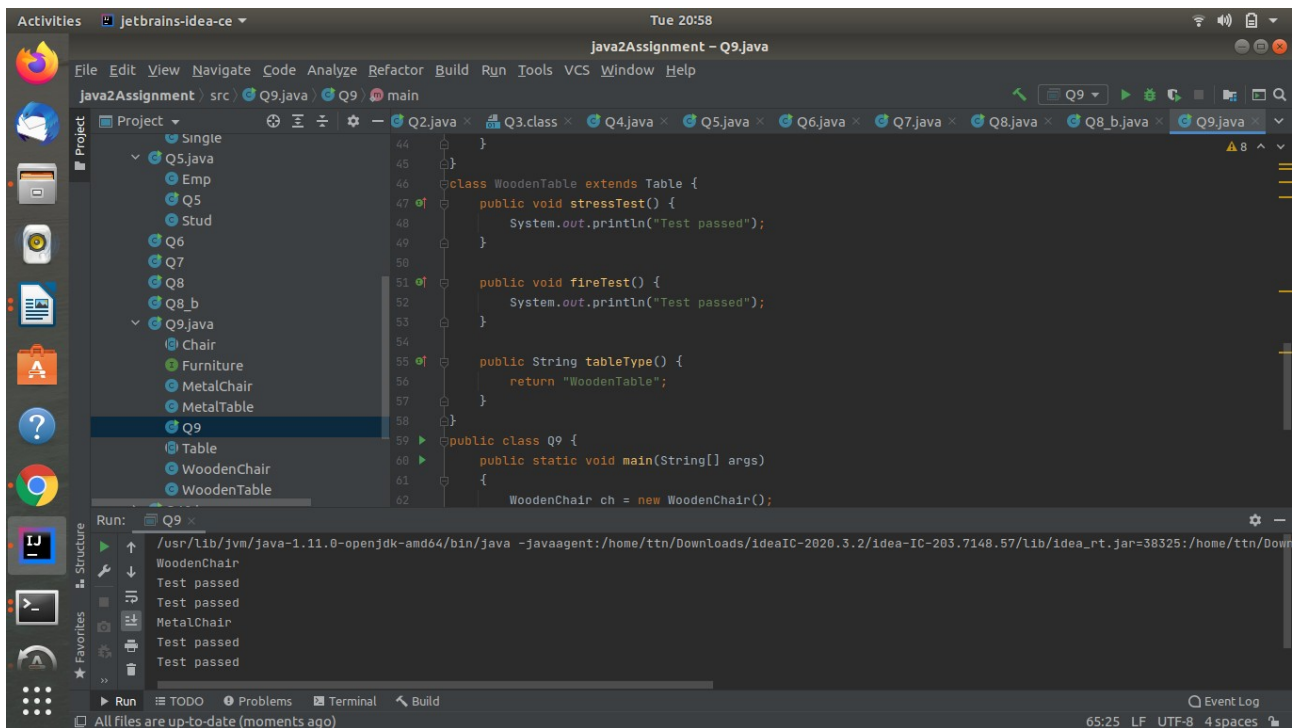
```
Run: Q9
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/home/ttn/Downloads/ideaIC-2020.3.2/idea-IC-203.7148.57/lib/idea_rt.jar=38325:/home/ttn/Down
WoodenChair
Test passed
Test passed
MetalChair
Test passed
Test passed
```

The screenshot shows the IntelliJ IDEA IDE with the 'java2Assignment - Q9.java' project. The Project view on the left shows the package structure: Q5.java, Emp, Q5, Stud, Q6, Q7, Q8, Q8_b, Q9.java, Chair, Furniture, MetalChair, MetalTable, Q9, Table, WoodenChair, and WoodenTable. The main editor displays the code for Q9.java, which defines the Furniture interface and its implementations, Chair and Table.

```
21 }
22 class WoodenChair extends Chair{
23     public void stressTest(){
24         System.out.println("Test passed");
25     }
26     public void fireTest(){
27         System.out.println("Test passed");
28     }
29     public String chairType() { return "WoodenChair"; }
30 }
31
32 class MetalTable extends Table {
33     public void stressTest() {
34         System.out.println("Test passed");
35     }
36     public void fireTest() {
37         System.out.println("Test passed");
38     }
39 }
40
41
```

The Run view at the bottom shows the execution of the Q9 class, with the following output:

```
Run: Q9
/usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/home/ttn/Downloads/ideaIC-2020.3.2/idea-IC-203.7148.57/lib/idea_rt.jar=38325:/home/ttn/Down
WoodenChair
Test passed
Test passed
MetalChair
Test passed
Test passed
```



Q10: 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.

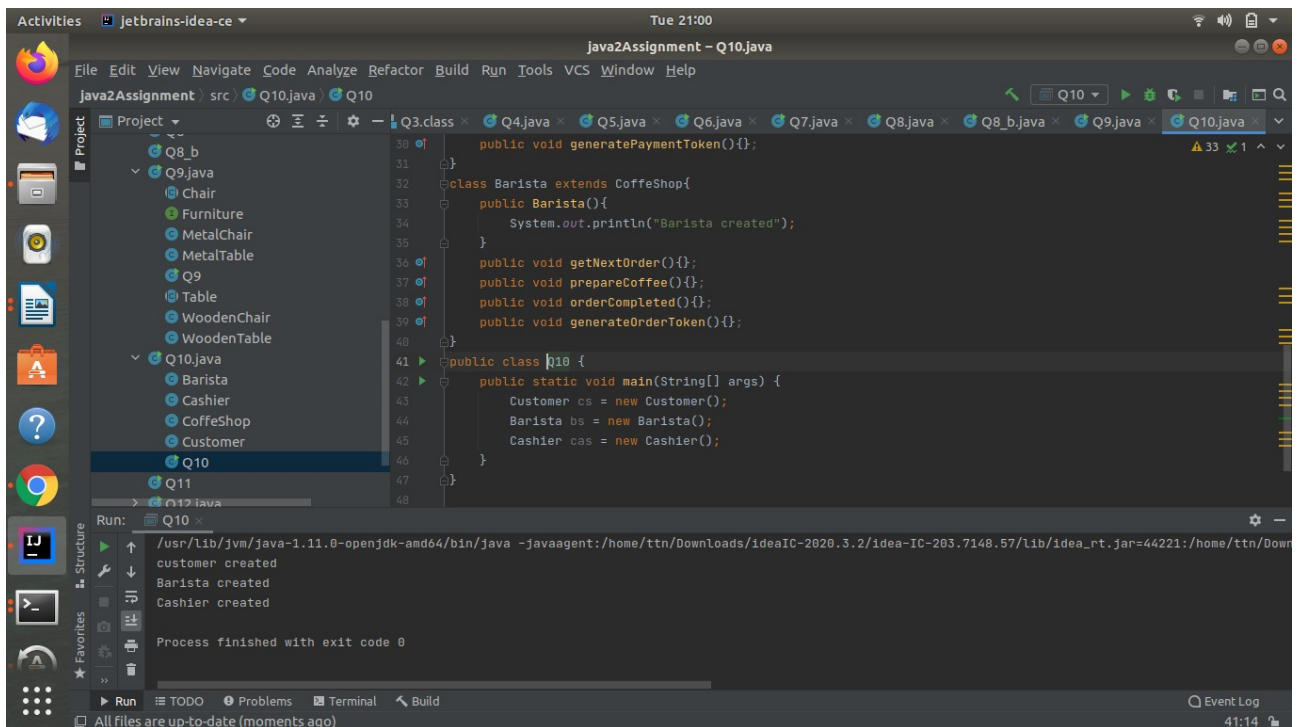
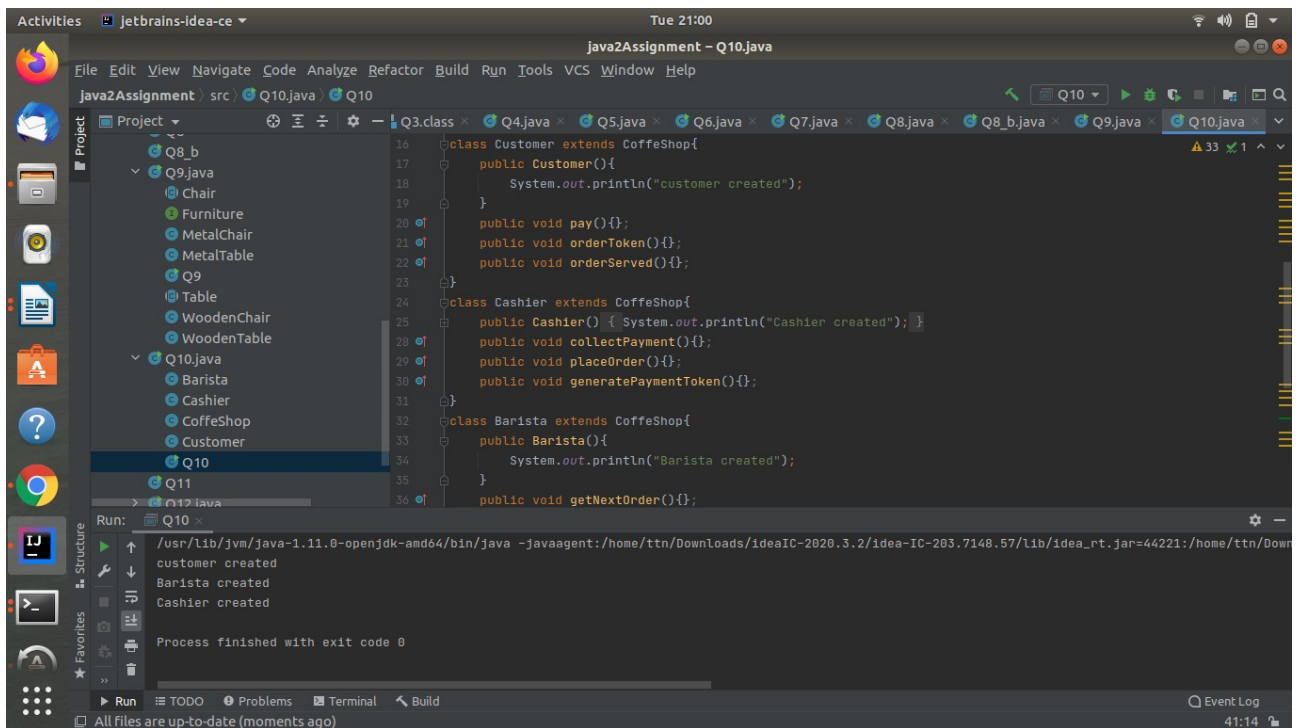
With a simple modification, we can design for a limited queue size)

* 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

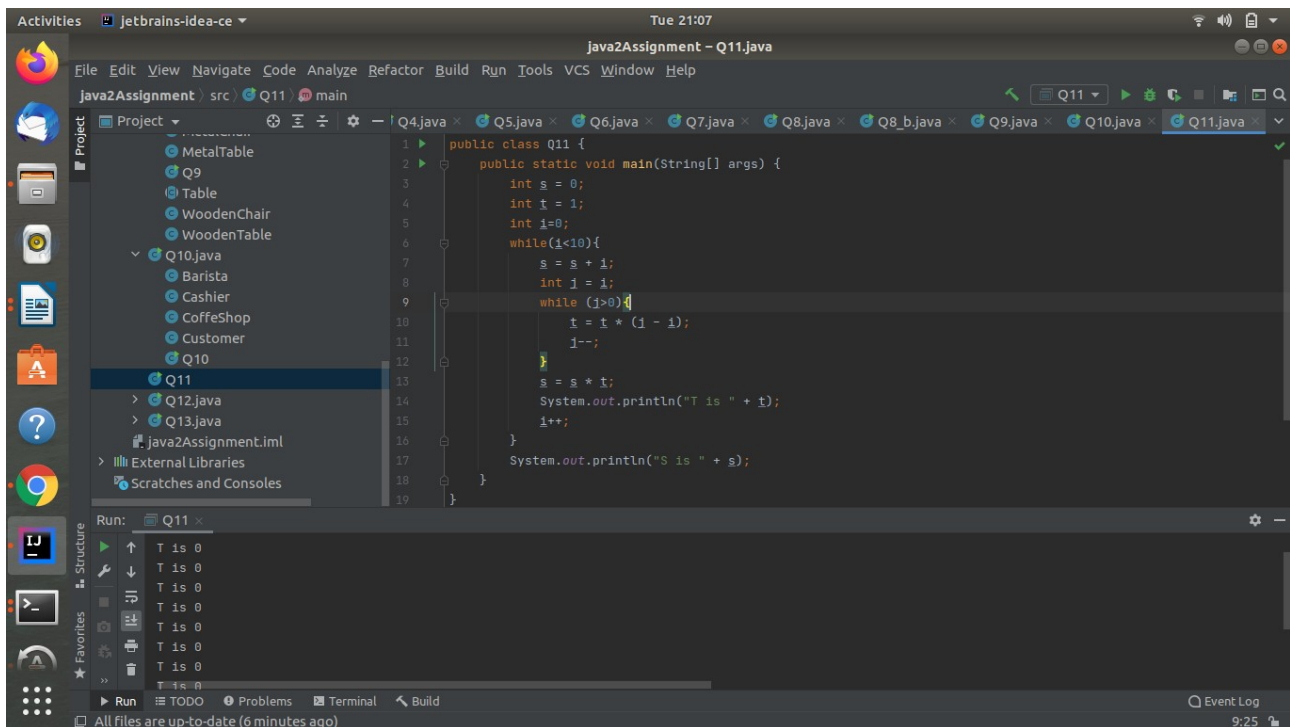
```
1 class CoffeShop {
2     public void pay(){};
3     public void orderToken(){};
4     public void orderServed(){};
5
6     public void collectPayment(){};
7     public void placeOrder(){};
8     public void generatePaymentToken(){};
9
10    public void getNextOrder(){};
11    public void prepareCoffee(){};
12    public void orderCompleted(){};
13    public void generateOrderToken(){};
14
15 }
16 class Customer extends CoffeShop{
17     public Customer(){
18         System.out.println("customer created");
19     }
20 }
```

Run: /usr/lib/jvm/java-1.11.0-openjdk-amd64/bin/java -javaagent:/home/ttn/Downloads/ideaIC-2020.3.2/idea-IC-203.7148.57/lib/idea_rt.jar=44221:/home/ttn/Down
customer created
Barista created
Cashier created
Process finished with exit code 0



Q11: 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);
```



Q12: 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");
    }

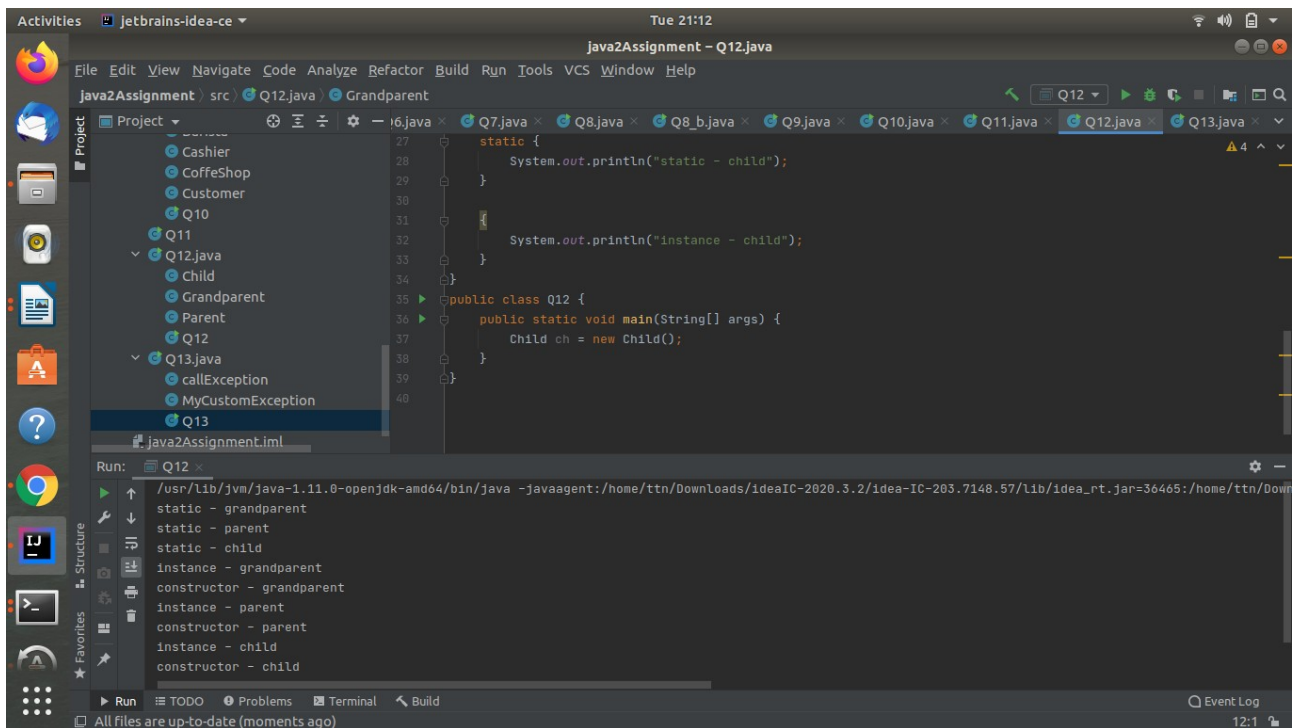
    public Grandparent() {
        System.out.println("constructor - grandparent");
    }
}

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.

