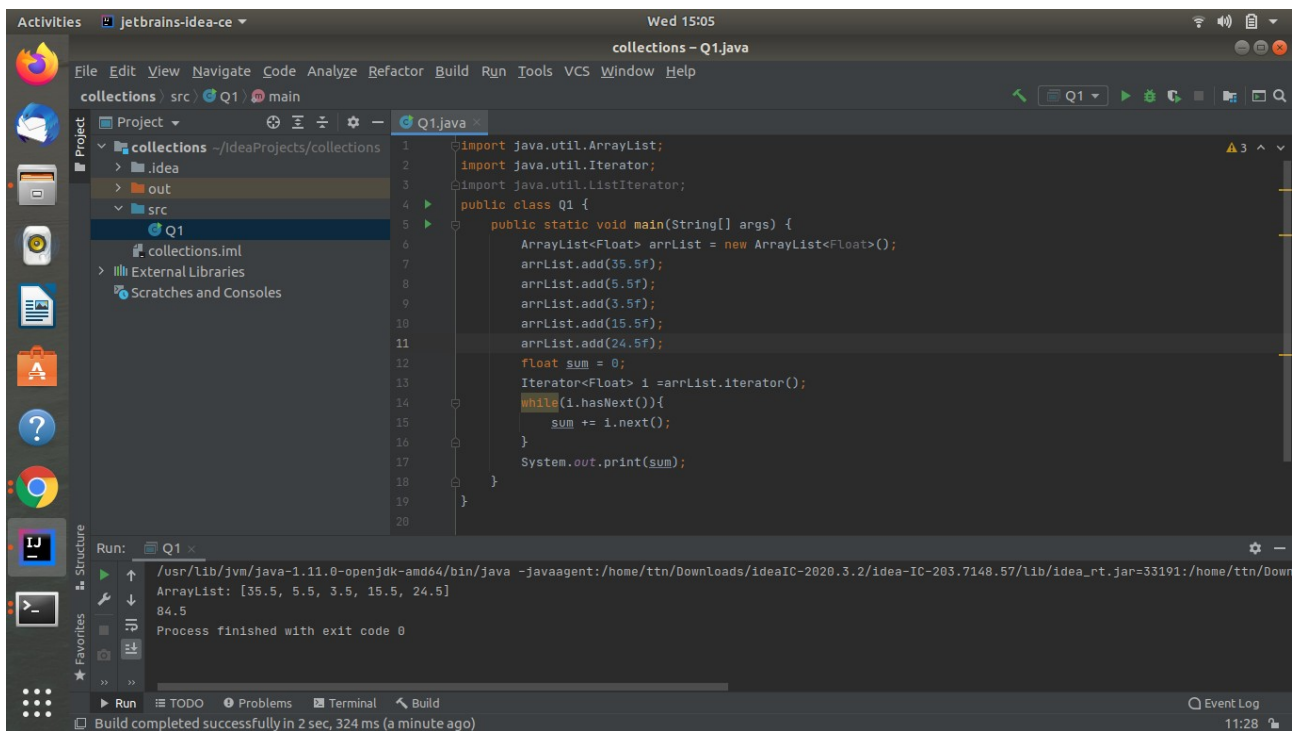


1. Write Java code to define List . Insert 5 floating point numbers in List, and using an iterator, find the sum of the numbers in List.

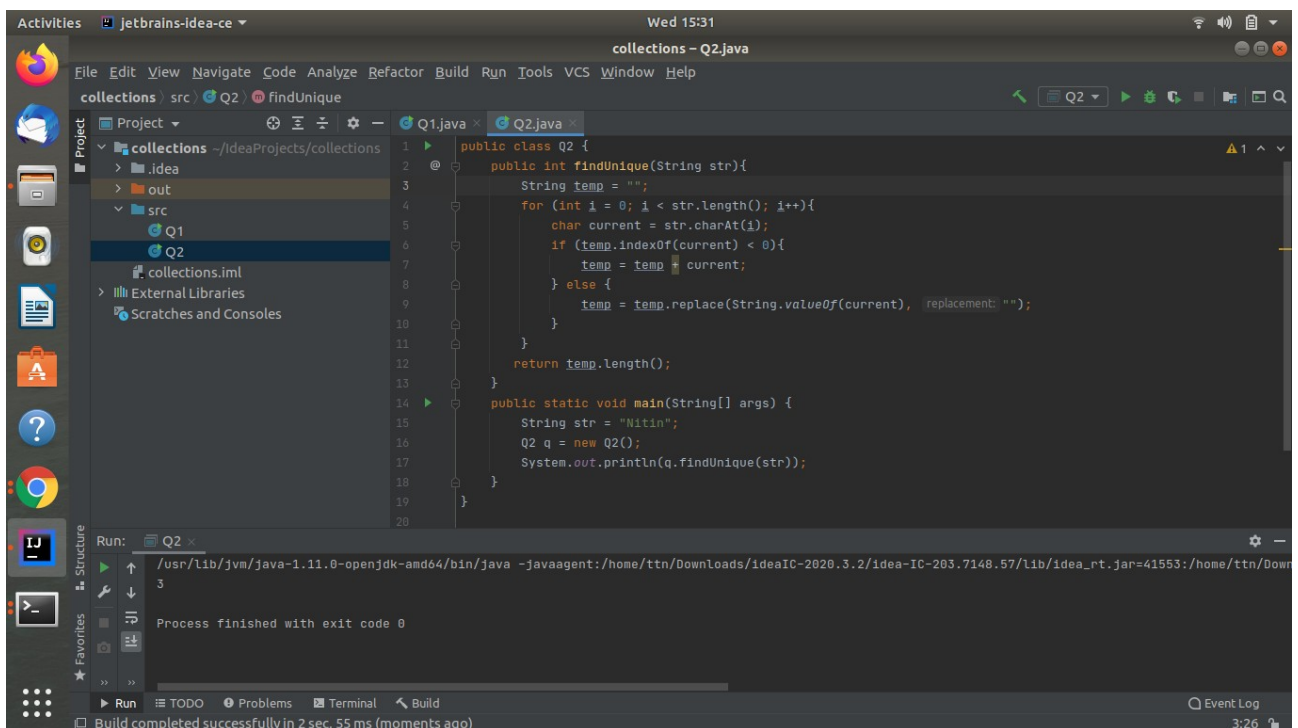


The screenshot shows the IntelliJ IDEA IDE with a project named 'collections'. The file 'Q1.java' is open, containing the following code:

```
1 import java.util.ArrayList;
2 import java.util.Iterator;
3 import java.util.ListIterator;
4 public class Q1 {
5     public static void main(String[] args) {
6         ArrayList<Float> arrList = new ArrayList<Float>();
7         arrList.add(35.5f);
8         arrList.add(5.5f);
9         arrList.add(3.5f);
10        arrList.add(15.5f);
11        arrList.add(24.5f);
12        float sum = 0;
13        Iterator<Float> i = arrList.iterator();
14        while(i.hasNext()){
15            sum += i.next();
16        }
17        System.out.print(sum);
18    }
19 }
```

The Run window at the bottom shows the output: 'ArrayList: [35.5, 5.5, 3.5, 15.5, 24.5]' and '84.5'. The process finished with exit code 0.

2Write a method that takes a string and returns the number of unique characters in the string.

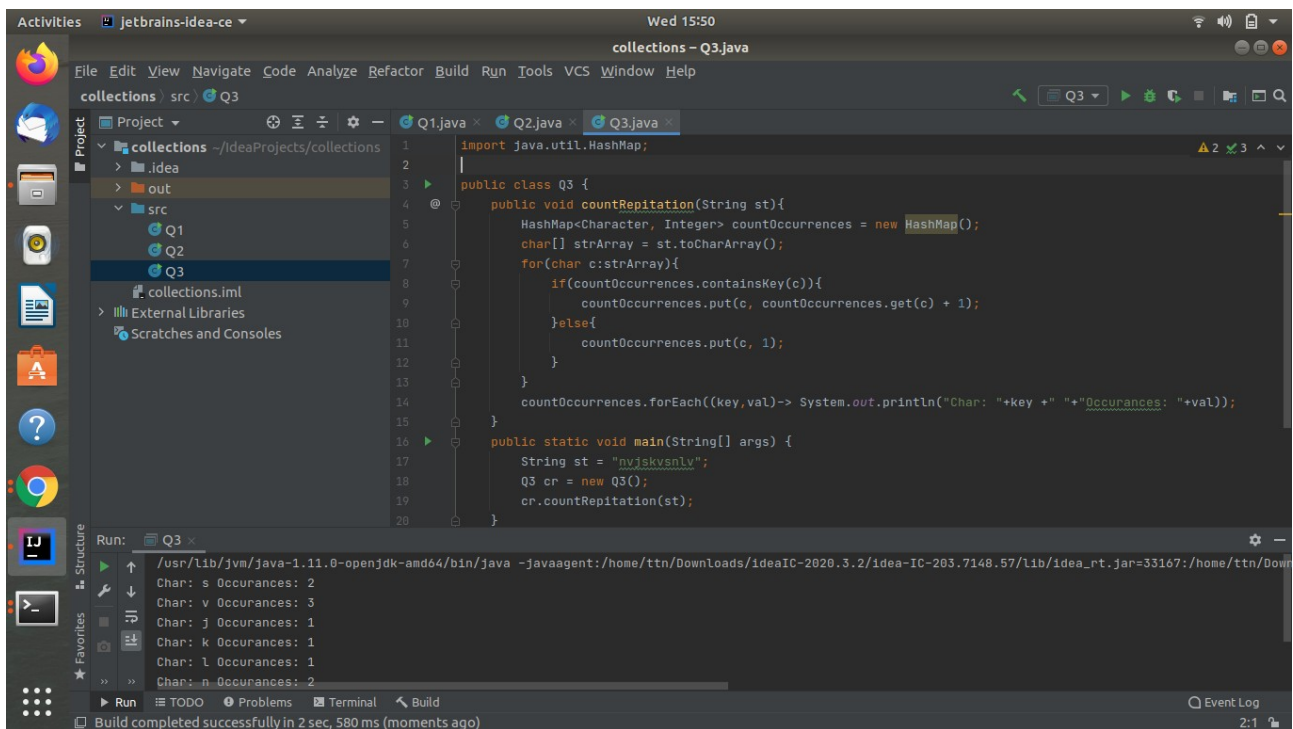


The screenshot shows the IntelliJ IDEA IDE with a project named 'collections'. The file 'Q2.java' is open, containing the following code:

```
1 public class Q2 {
2     public int findUnique(String str){
3         String temp = "";
4         for (int i = 0; i < str.length(); i++){
5             char current = str.charAt(i);
6             if (temp.indexOf(current) < 0){
7                 temp = temp + current;
8             } else {
9                 temp = temp.replace(String.valueOf(current), "");
10            }
11        }
12        return temp.length();
13    }
14    public static void main(String[] args) {
15        String str = "Nitin";
16        Q2 q = new Q2();
17        System.out.println(q.findUnique(str));
18    }
19 }
```

The Run window at the bottom shows the output: '3'. The process finished with exit code 0.

3. Write a method that takes a string and print the number of occurrence of each character in the string.



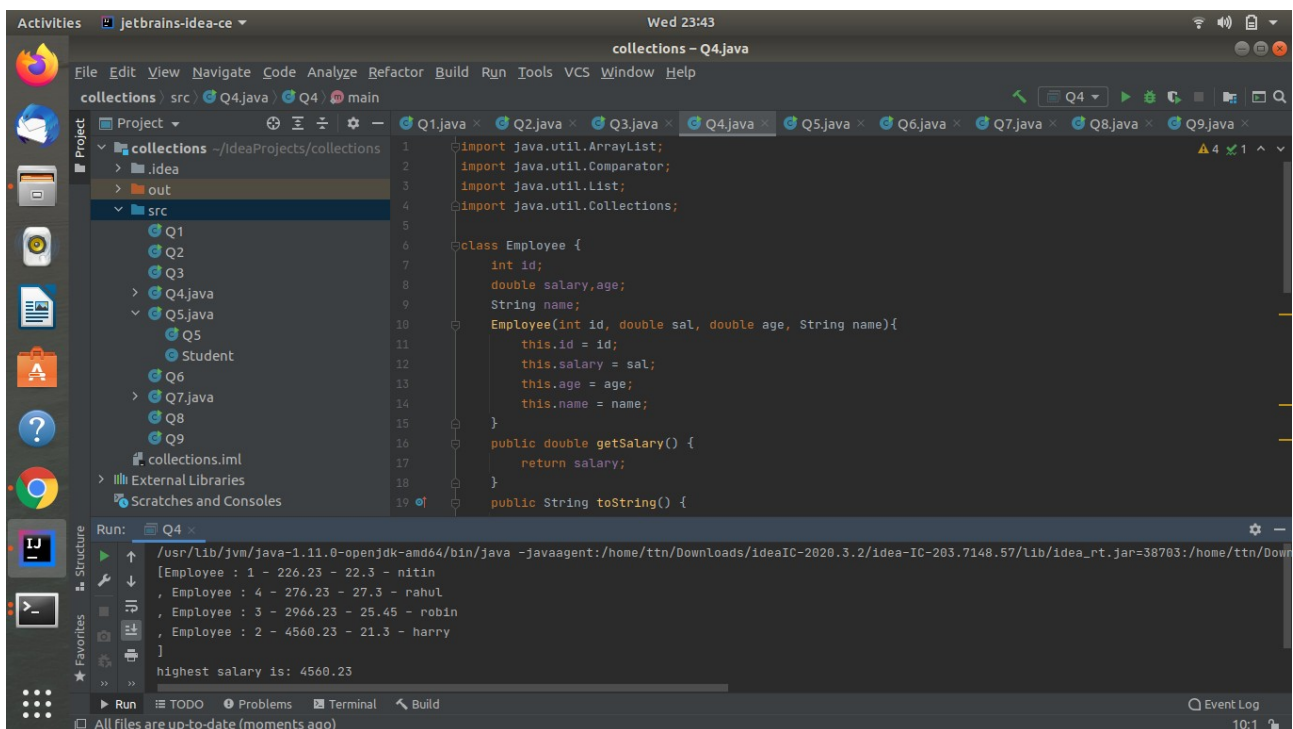
The screenshot shows the IntelliJ IDEA IDE with a project named 'collections'. The 'src' directory contains files Q1, Q2, and Q3. The 'Q3.java' file is open, showing a Java class with a method to count character occurrences. The code uses a HashMap to store the frequency of each character in a given string. The main method tests the class with the string 'nviskvsnlv'.

```
1 import java.util.HashMap;
2
3 public class Q3 {
4     public void countRepitition(String st){
5         HashMap<Character, Integer> countOccurrences = new HashMap();
6         char[] strArray = st.toCharArray();
7         for(char c:strArray){
8             if(countOccurrences.containsKey(c)){
9                 countOccurrences.put(c, countOccurrences.get(c) + 1);
10            }else{
11                countOccurrences.put(c, 1);
12            }
13        }
14        countOccurrences.forEach((key, val)-> System.out.println("Char: "+key +" "+"Occurrences: "+val));
15    }
16    public static void main(String[] args) {
17        String st = "nviskvsnlv";
18        Q3 cr = new Q3();
19        cr.countRepitition(st);
20    }
21 }
```

The Run console shows the output of the program:

```
Char: s Occurances: 2
Char: v Occurances: 3
Char: j Occurances: 1
Char: k Occurances: 1
Char: l Occurances: 1
Char: n Occurances: 2
```

4. Write a program to sort Employee objects based on highest salary using Comparator. Employee class{ Double Age; Double Salary; String Name

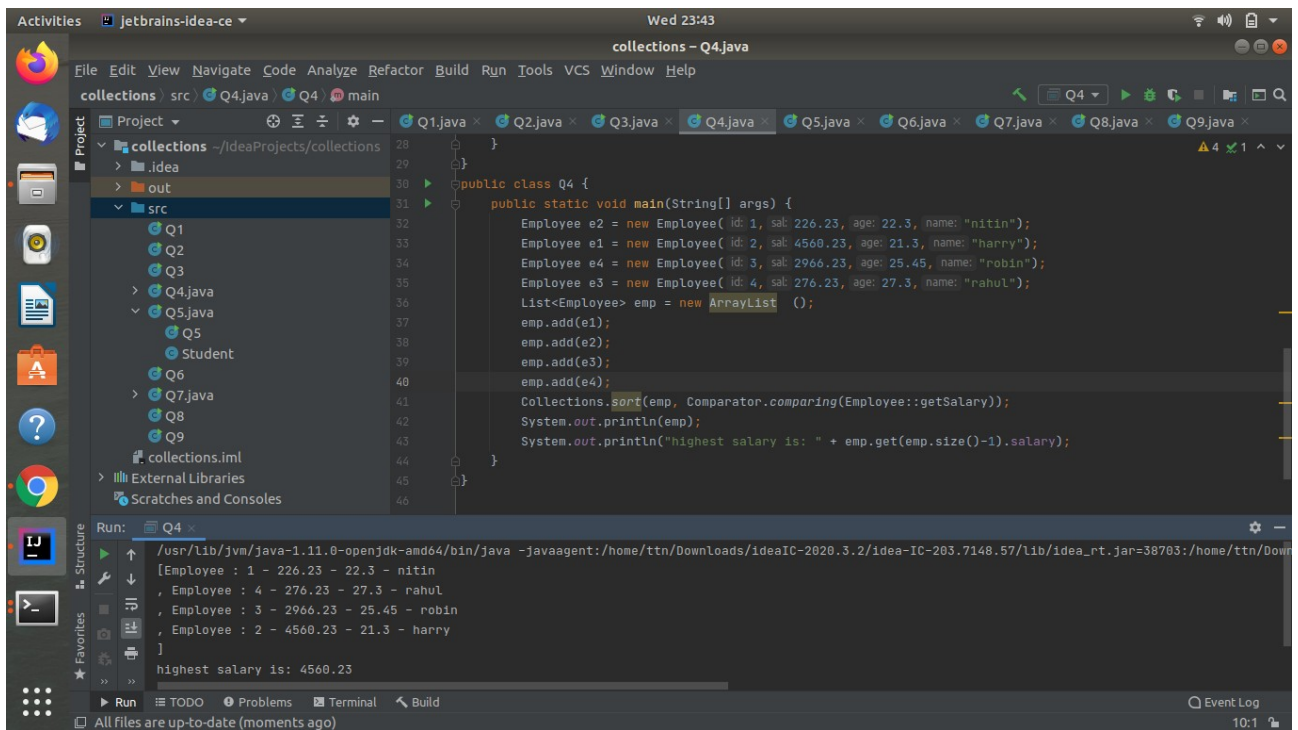
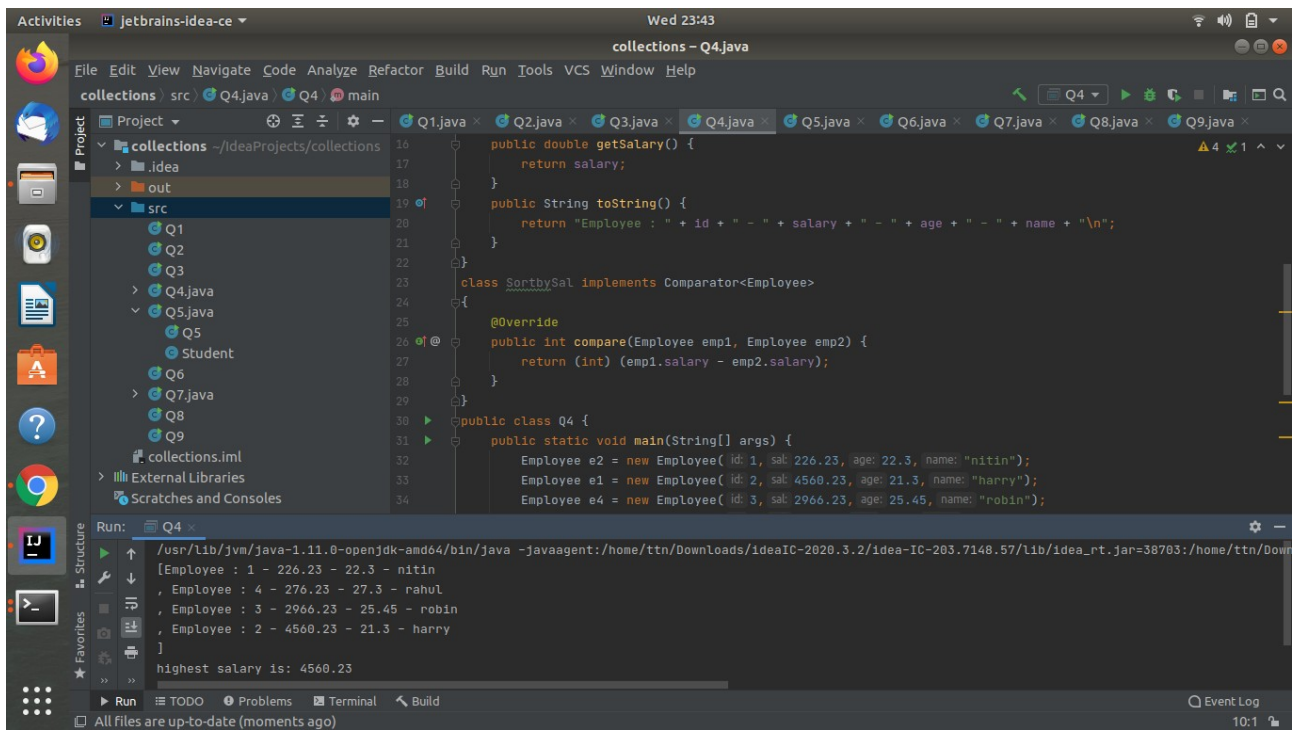


The screenshot shows the IntelliJ IDEA IDE with a project named 'collections'. The 'src' directory contains files Q1 through Q9. The 'Q4.java' file is open, showing a Java class with an Employee class and a main method. The Employee class has attributes id, salary, age, and name. The main method creates a list of Employee objects and sorts them based on salary using a Comparator. The output shows the sorted list of employees and the highest salary.

```
1 import java.util.ArrayList;
2 import java.util.Comparator;
3 import java.util.List;
4 import java.util.Collections;
5
6 class Employee {
7     int id;
8     double salary, age;
9     String name;
10    Employee(int id, double sal, double age, String name){
11        this.id = id;
12        this.salary = sal;
13        this.age = age;
14        this.name = name;
15    }
16    public double getSalary() {
17        return salary;
18    }
19    public String toString() {
```

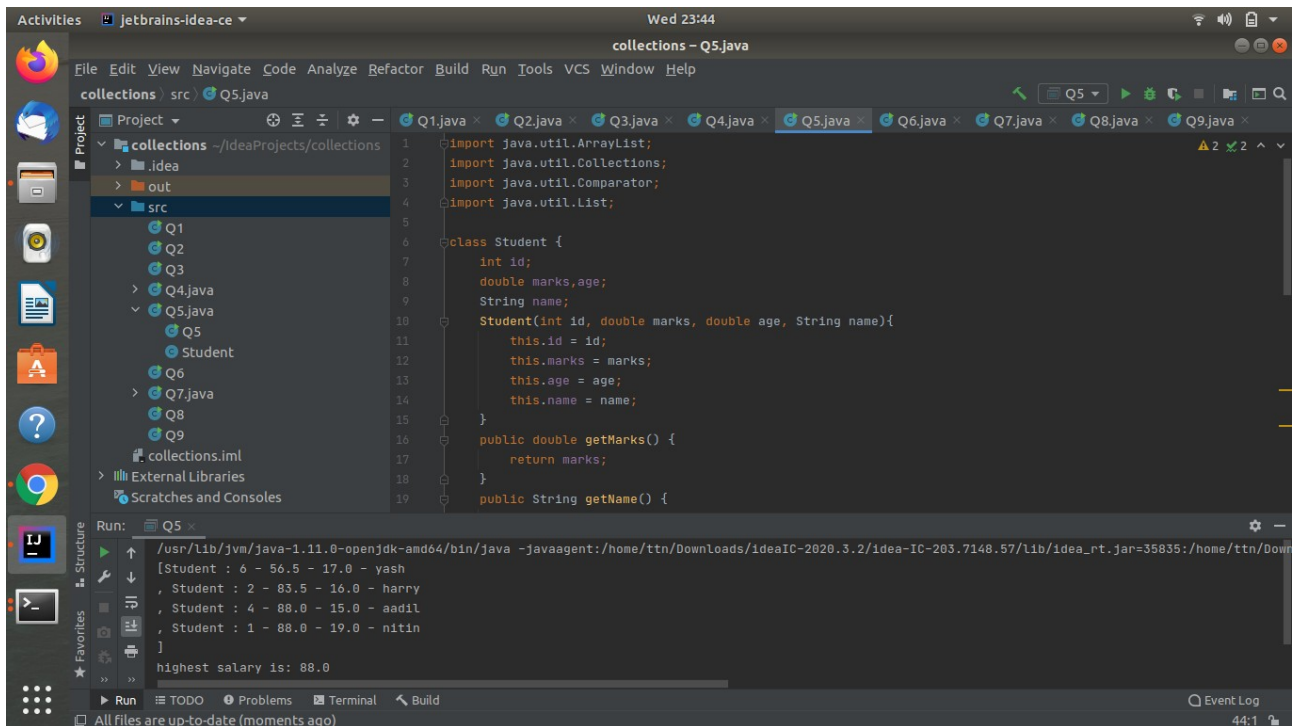
The Run console shows the output of the program:

```
[Employee : 1 - 226.23 - 22.3 - nitin
, Employee : 4 - 276.23 - 27.3 - rahul
, Employee : 3 - 2966.23 - 25.45 - robin
, Employee : 2 - 4560.23 - 21.3 - harry
]
highest salary is: 4560.23
```





5. Write a program to sort the Student objects based on Score, if the score are same then sort on First Name. Class Student{ String Name; Double Score; Double Age

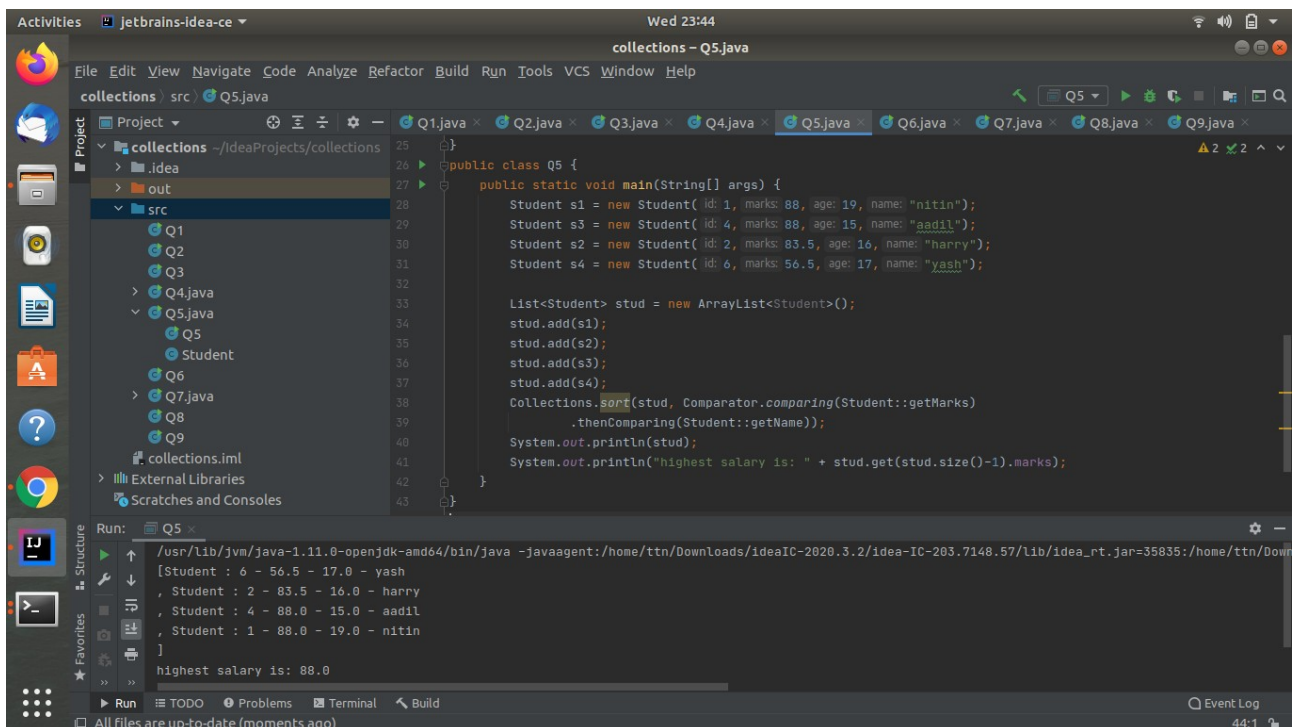


The screenshot shows the IntelliJ IDEA interface with the 'collections' project open. The 'Q5.java' file is selected, displaying the following code:

```
1 import java.util.ArrayList;
2 import java.util.Collections;
3 import java.util.Comparator;
4 import java.util.List;
5
6 class Student {
7     int id;
8     double marks, age;
9     String name;
10    Student(int id, double marks, double age, String name){
11        this.id = id;
12        this.marks = marks;
13        this.age = age;
14        this.name = name;
15    }
16    public double getMarks() {
17        return marks;
18    }
19    public String getName() {
```

The Run window at the bottom shows the execution 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=35835:/home/ttn/Down
[Student : 6 - 56.5 - 17.0 - yash
Student : 2 - 83.5 - 16.0 - harry
Student : 4 - 88.0 - 15.0 - aadil
Student : 1 - 88.0 - 19.0 - nitin
]
highest salary is: 88.0
```



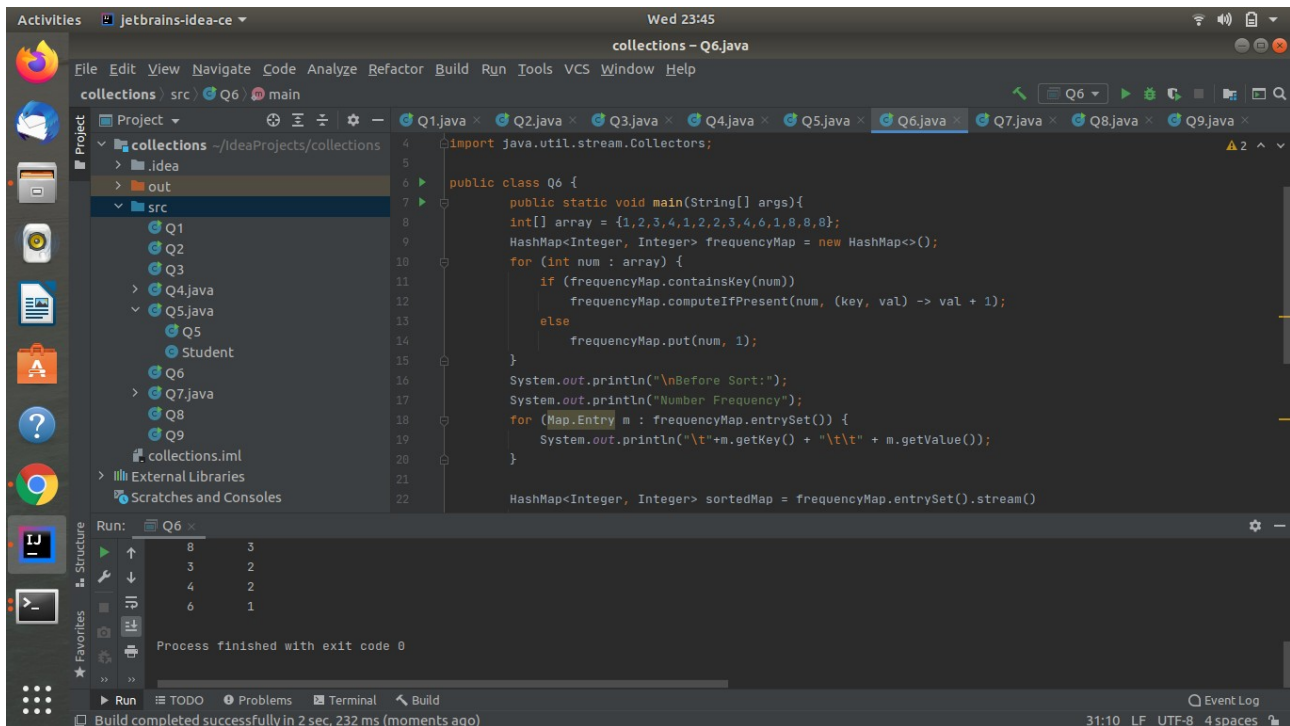
The screenshot shows the IntelliJ IDEA interface with the 'collections' project open. The 'Q5.java' file is selected, displaying the following code:

```
25 }
26 public class Q5 {
27     public static void main(String[] args) {
28         Student s1 = new Student(id: 1, marks: 88, age: 19, name: "nitin");
29         Student s3 = new Student(id: 4, marks: 88, age: 15, name: "aadil");
30         Student s2 = new Student(id: 2, marks: 83.5, age: 16, name: "harry");
31         Student s4 = new Student(id: 6, marks: 56.5, age: 17, name: "yash");
32
33         List<Student> stud = new ArrayList<Student>();
34         stud.add(s1);
35         stud.add(s2);
36         stud.add(s3);
37         stud.add(s4);
38         Collections.sort(stud, Comparator.comparing(Student::getMarks)
39             .thenComparing(Student::getName));
40         System.out.println(stud);
41         System.out.println("highest salary is: " + stud.get(stud.size()-1).marks);
42     }
43 }
```

The Run window at the bottom shows the execution 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=35835:/home/ttn/Down
[Student : 6 - 56.5 - 17.0 - yash
Student : 2 - 83.5 - 16.0 - harry
Student : 4 - 88.0 - 15.0 - aadil
Student : 1 - 88.0 - 19.0 - nitin
]
highest salary is: 88.0
```

6. Print the elements of an array in the decreasing frequency if 2 numbers have same frequency then print the one which came first.



```
import java.util.stream.Collectors;

public class Q6 {
    public static void main(String[] args){
        int[] array = {1,2,3,4,1,2,2,3,4,6,1,8,8,8};
        HashMap<Integer, Integer> frequencyMap = new HashMap<>();
        for (int num : array) {
            if (frequencyMap.containsKey(num))
                frequencyMap.computeIfPresent(num, (key, val) -> val + 1);
            else
                frequencyMap.put(num, 1);
        }
        System.out.println("\nBefore Sort:");
        System.out.println("Number Frequency");
        for (Map.Entry m : frequencyMap.entrySet()) {
            System.out.println("\t"+m.getKey() + "\t\t" + m.getValue());
        }

        HashMap<Integer, Integer> sortedMap = frequencyMap.entrySet().stream()

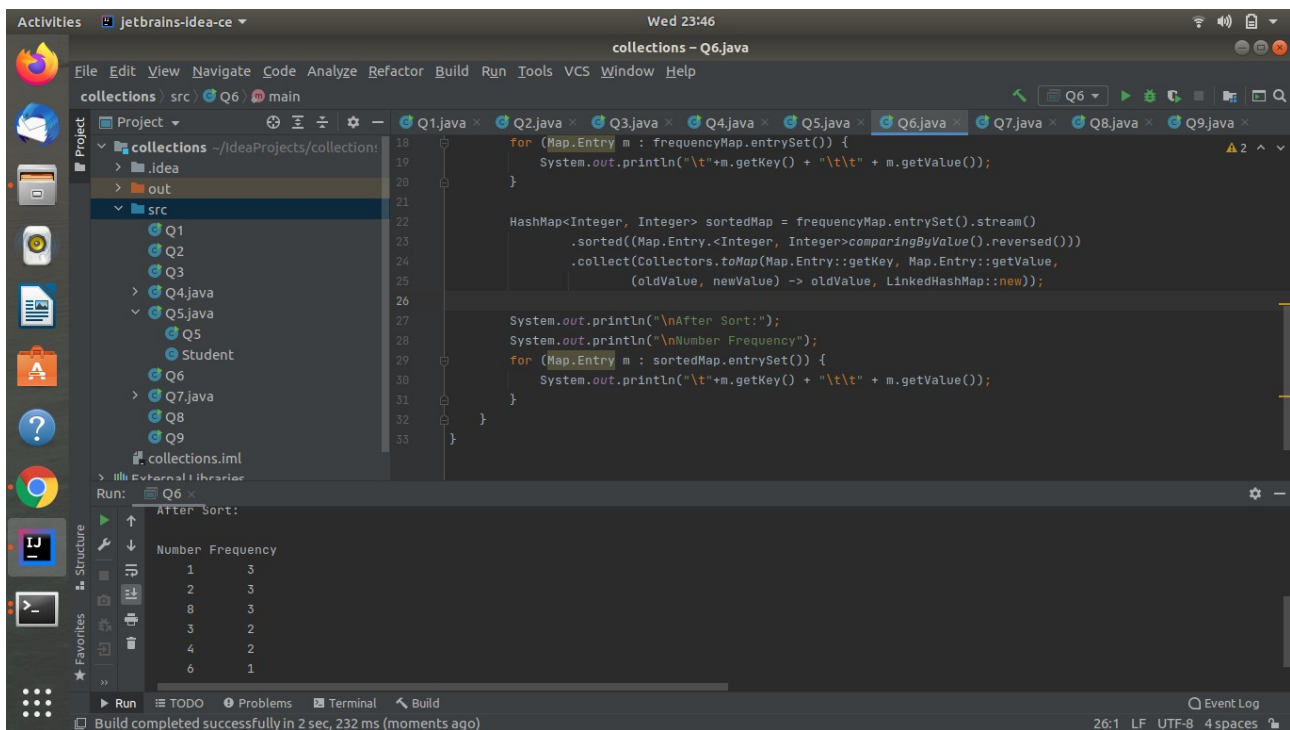
```

Run: Q6 x

Number	Frequency
8	3
3	2
4	2
6	1

Process finished with exit code 0

Build completed successfully in 2 sec, 232 ms (moments ago)



```
        HashMap<Integer, Integer> sortedMap = frequencyMap.entrySet().stream()
            .sorted((Map.Entry.<Integer, Integer>comparingByValue().reversed()))
            .collect(Collectors.toMap(Map.Entry::getKey, Map.Entry::getValue,
                (oldValue, newValue) -> oldValue, LinkedHashMap::new));

        System.out.println("\nAfter Sort:");
        System.out.println("Number Frequency");
        for (Map.Entry m : sortedMap.entrySet()) {
            System.out.println("\t"+m.getKey() + "\t\t" + m.getValue());
        }
    }
}
```

Run: Q6 x

After Sort:

Number	Frequency
1	3
2	3
8	3
3	2
4	2
6	1

Build completed successfully in 2 sec, 232 ms (moments ago)

7.Design a Data Structure SpecialStack that supports all the stack operations like push(), pop(), isEmpty(), isFull() and an additional operation getMin() which should return minimum element from the SpecialStack. (Expected complexity  $O(1)$ )

The screenshot shows the IntelliJ IDEA interface with the `SpecialStack` class implemented. The class uses two stacks: `s` for the main elements and `min` to keep track of the minimum element. The `push` method is currently visible in the editor.

```

import java.util.Stack;

class SpecialStack {
    Stack<Integer> min;
    Stack<Integer> s;
    Integer minEl;
    int currSize = -1;

    SpecialStack(int size){
        s = new Stack<Integer>();
        min = new Stack<Integer>();
    }

    void getMin(){
        minEl = min.pop();
        min.push(minEl);
        if(s.isEmpty()){
            System.out.println("stack is empty");
        }
    }
}
  
```

The Run console shows the following output for the `push` operation:

```

element pushed5
element pushed3
Stack is not full
Minimum element is3
Removed element3
Minimum element is5
Process finished with exit code 0
  
```

The screenshot shows the IntelliJ IDEA interface with the `SpecialStack` class implemented. The `pop` method is currently visible in the editor. It checks if the stack is empty and updates the `min` stack accordingly.

```

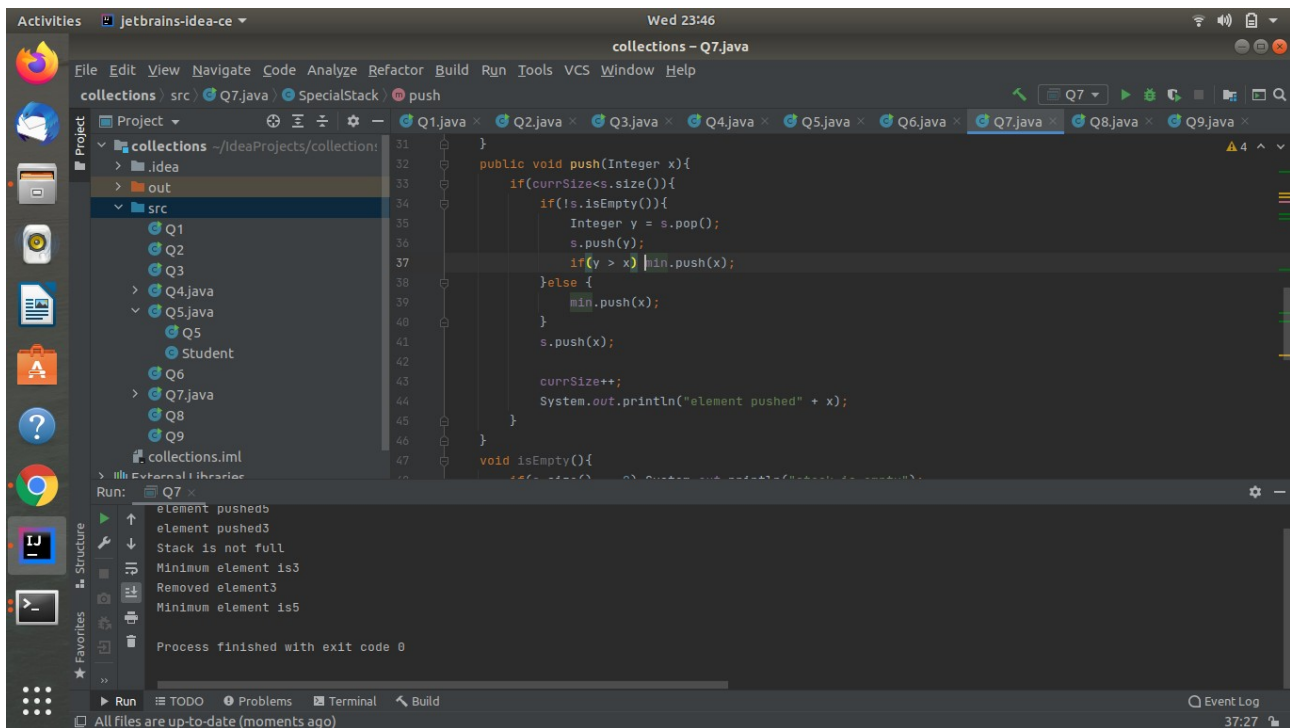
    void pop(){
        if(s.isEmpty()){
            System.out.println("stack is empty cannot remove element");
        }else {
            Integer t = s.pop();
            min.pop();
            currSize--;
            System.out.println("Removed element" + t);
        }
    }

    public void push(Integer x){
        if(currSize<s.size()){
            if(!s.isEmpty()){
                Integer y = s.pop();
                s.push(y);
                if(y > x) min.push(x);
            }else {
                min.push(x);
            }
        }
    }
  
```

The Run console shows the following output for the `pop` operation:

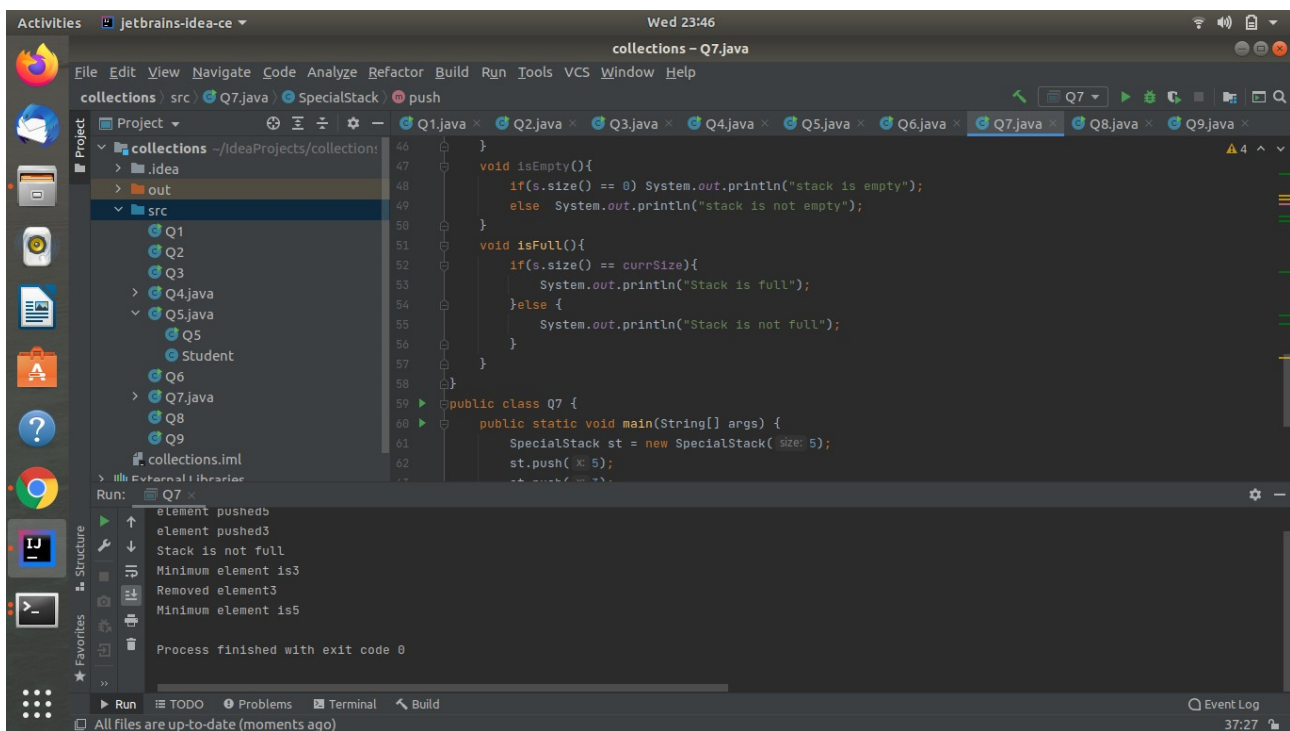
```

element pushed5
element pushed3
Stack is not full
Minimum element is3
Removed element3
Minimum element is5
Process finished with exit code 0
  
```

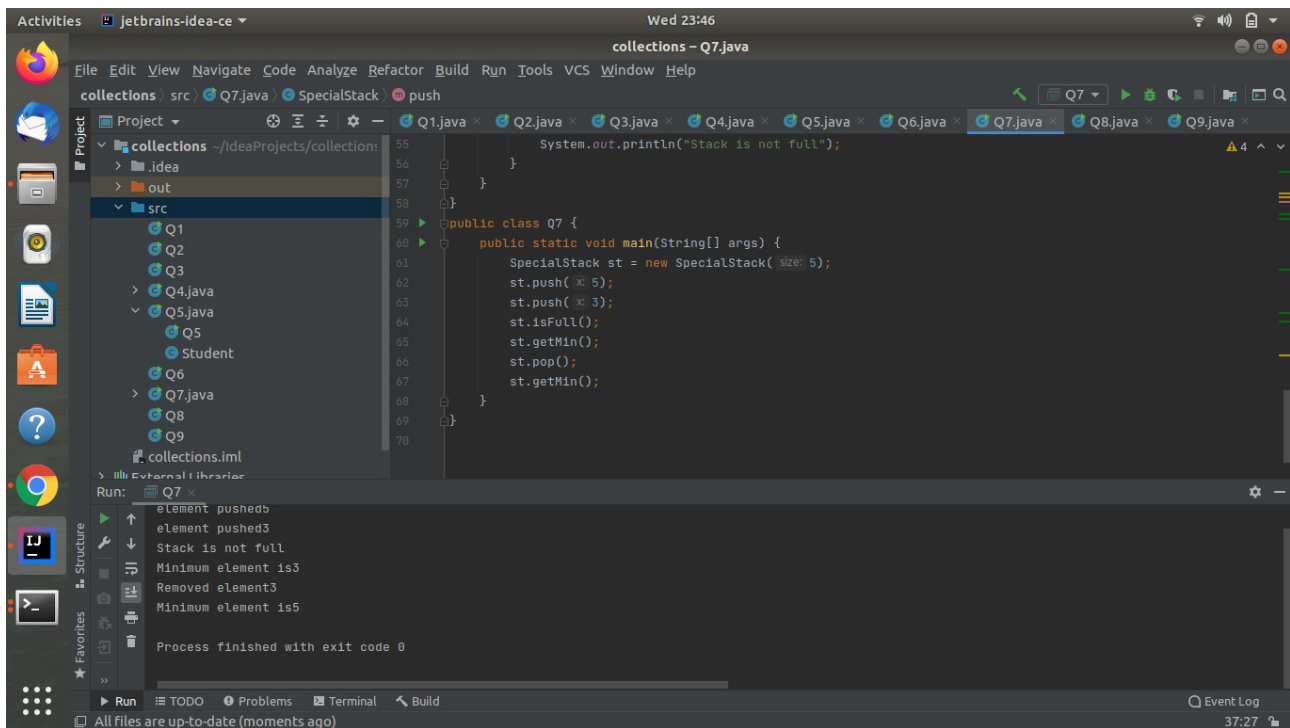


8. Write a program to format date as example "21-March-2016"

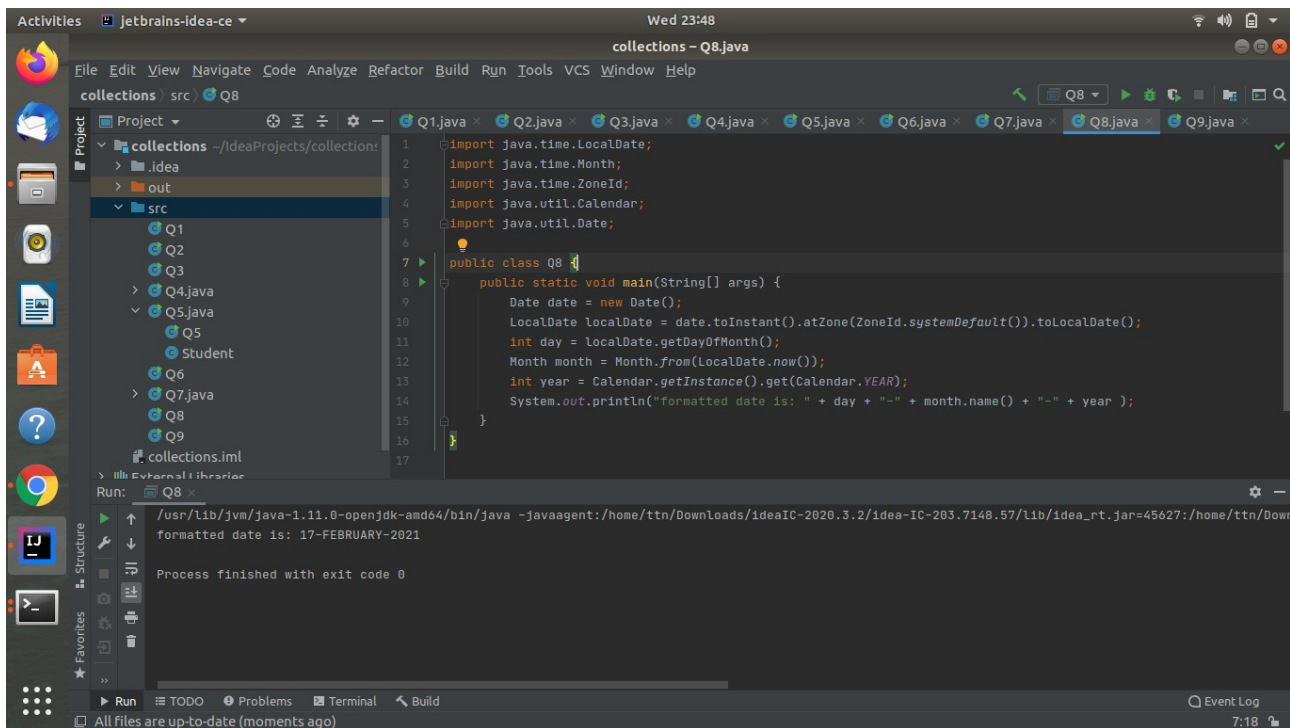
9. Write a program to display times in different country format.





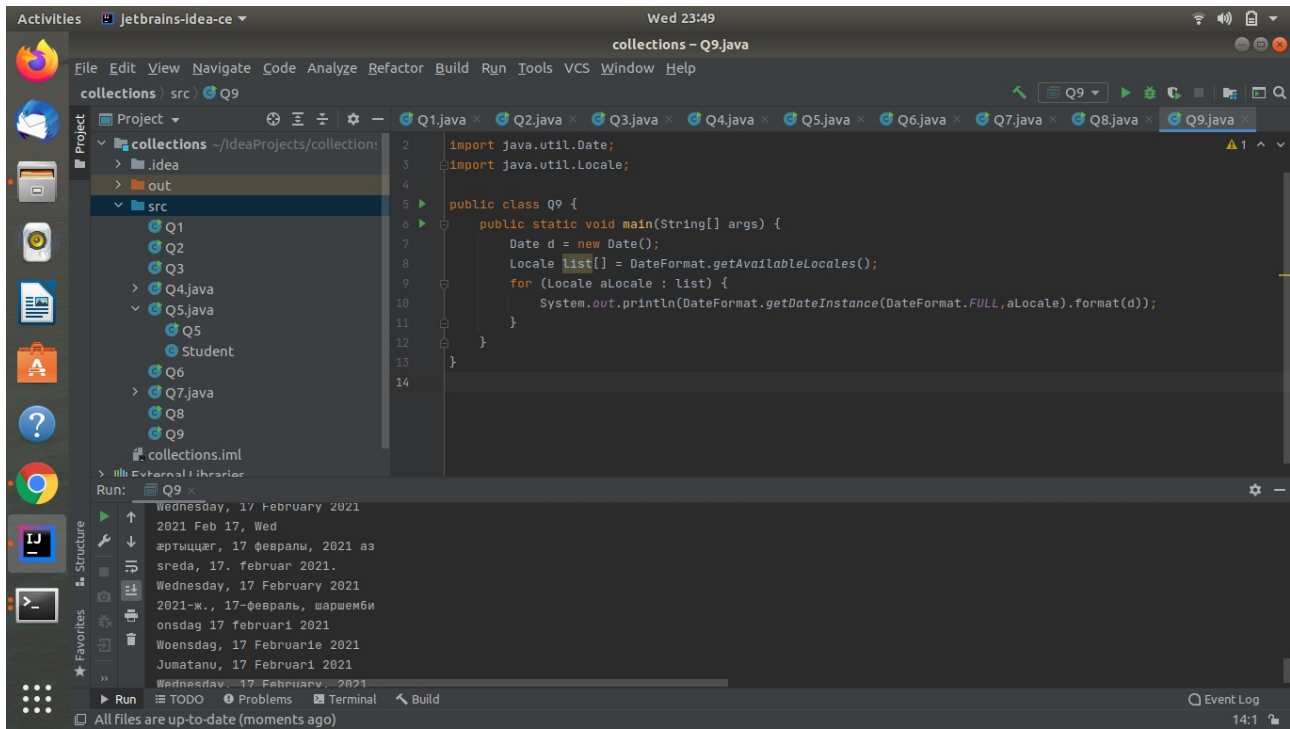


8. Write a program to format date as example "21-March-2016"





9. Write a program to display times in different country format.



The screenshot shows the IntelliJ IDEA IDE with a project named 'collections'. The file 'Q9.java' is open in the editor. The code imports `java.util.Date` and `java.util.Locale`. It defines a class `Q9` with a `main` method. Inside `main`, a `Date` object `d` is created. A list of available locales is retrieved using `DateFormat.getAvailableLocales()`. A loop iterates over this list, and for each locale, the date is formatted using `DateFormat.getInstance(DateFormat.FULL, aLocale).format(d)` and printed to the console.

```
1 import java.util.Date;
2 import java.util.Locale;
3
4
5 public class Q9 {
6     public static void main(String[] args) {
7         Date d = new Date();
8         Locale list[] = DateFormat.getAvailableLocales();
9         for (Locale aLocale : list) {
10             System.out.println(DateFormat.getInstance(DateFormat.FULL, aLocale).format(d));
11         }
12     }
13 }
14
```

The Run window shows the output of the program, displaying the date 'Wednesday, 17 February 2021' in various formats and languages, including English, Russian, and others.

Run: Q9

```
Wednesday, 17 February 2021
2021 Feb 17, Wed
среда, 17 февраля, 2021 аз
sreda, 17. februar 2021.
Wednesday, 17 February 2021
2021-ж., 17-февраль, шаршемби
onsdag 17 februari 2021
Woensdag, 17 Februarie 2021
Jumatanu, 17 Februari 2021
Wednesday, 17 February, 2021
```

Event Log

14:1

All files are up-to-date (moments ago)