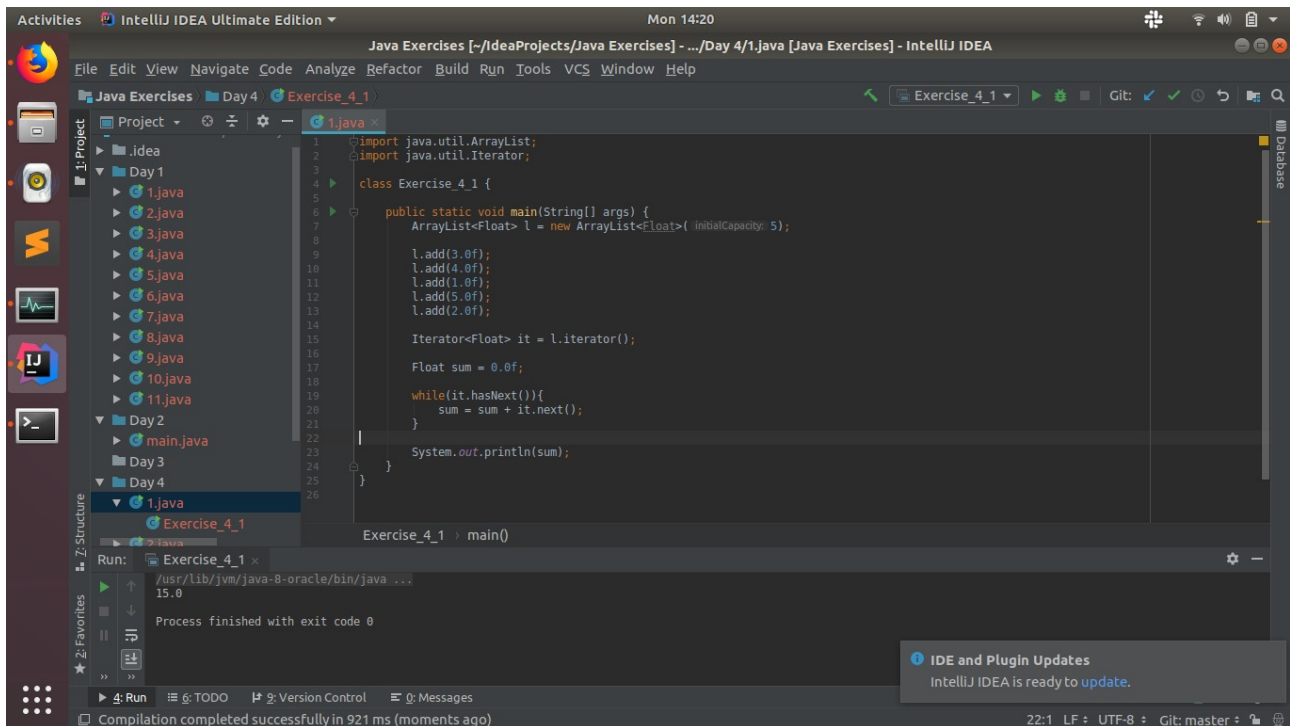


Q1. Write Java code to define List . Insert 5 floating point numbers in List, and using an iterator, find the sum of the numbers in List.

A1.



The screenshot shows the IntelliJ IDEA interface with a Java project named 'Java Exercises'. The file explorer on the left shows a directory structure with 'Day 4' containing 'Exercise_4_1'. The main editor displays the code for 'Exercise_4_1.java'. The code imports 'ArrayList' and 'Iterator' from 'java.util'. It defines a class 'Exercise_4_1' with a 'main' method. In the 'main' method, an 'ArrayList' of 'Float' is created with an initial capacity of 5. Five floating-point numbers are added to the list: 3.0f, 4.0f, 1.0f, 5.0f, and 2.0f. An 'Iterator' is obtained from the list, and a 'while' loop is used to iterate through the list, summing the elements. The final sum is printed to the console. The 'Run' tab at the bottom shows the output '15.0' and a message 'Process finished with exit code 0'. A notification at the bottom right indicates 'IDE and Plugin Updates'.

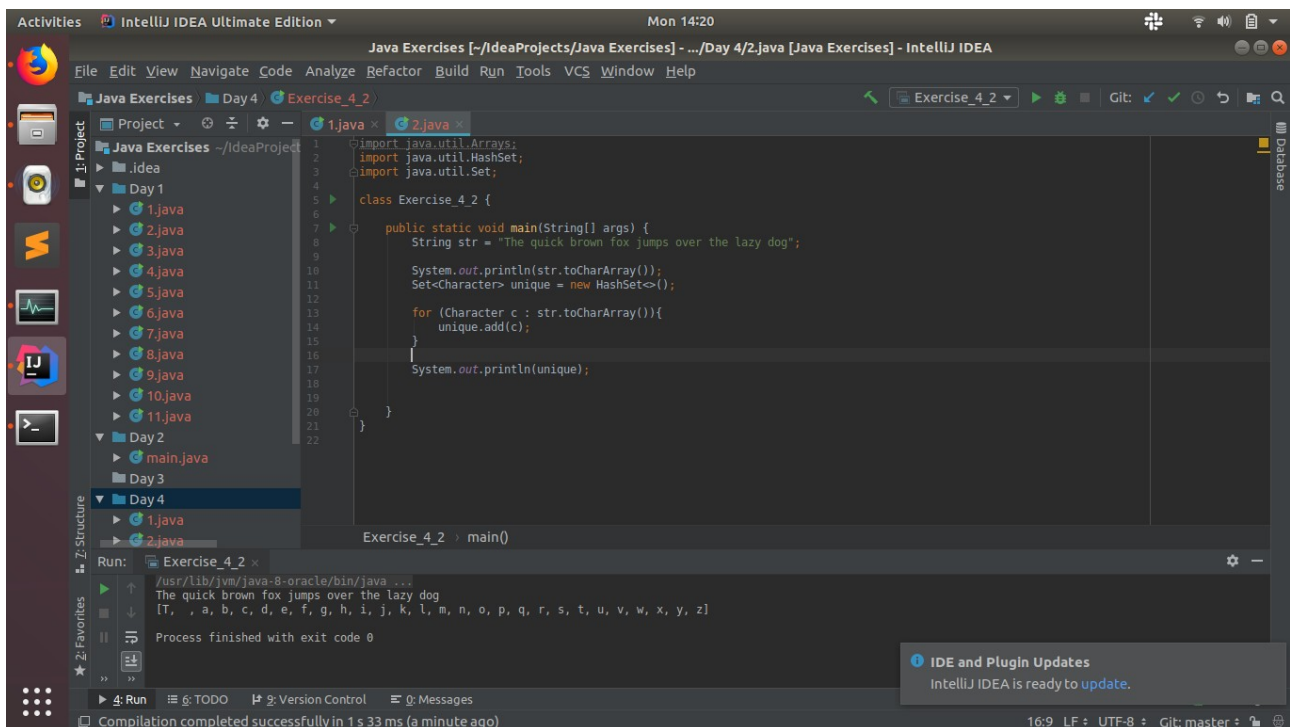
```
1 import java.util.ArrayList;
2 import java.util.Iterator;
3
4 class Exercise_4_1 {
5
6     public static void main(String[] args) {
7         ArrayList<Float> l = new ArrayList<Float> (initialCapacity: 5);
8
9         l.add(3.0f);
10        l.add(4.0f);
11        l.add(1.0f);
12        l.add(5.0f);
13        l.add(2.0f);
14
15        Iterator<Float> it = l.iterator();
16
17        Float sum = 0.0f;
18
19        while(it.hasNext()){
20            sum = sum + it.next();
21        }
22
23        System.out.println(sum);
24    }
25 }
26
```

Run: Exercise_4_1 x
/usr/lib/jvm/java-8-oracle/bin/java ...
15.0
Process finished with exit code 0

IDE and Plugin Updates
IntelliJ IDEA is ready to update.

Q2. Write a method that takes a string and returns the number of unique characters in the string.

A2.



The screenshot shows the IntelliJ IDEA interface with a Java project named 'Java Exercises'. The file explorer on the left shows a directory structure with 'Day 4' containing 'Exercise_4_2'. The main editor displays the code for 'Exercise_4_2.java'. The code imports 'Arrays', 'HashSet', and 'Set' from 'java.util'. It defines a class 'Exercise_4_2' with a 'main' method. In the 'main' method, a string 'The quick brown fox jumps over the lazy dog' is defined. The string is converted to a character array, and a 'HashSet' is created to store unique characters. A 'for' loop iterates through the character array, adding each character to the 'HashSet'. The final size of the 'HashSet' is printed to the console. The 'Run' tab at the bottom shows the output 'The quick brown fox jumps over the lazy dog' and '[T, , a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z]' and a message 'Process finished with exit code 0'. A notification at the bottom right indicates 'IDE and Plugin Updates'.

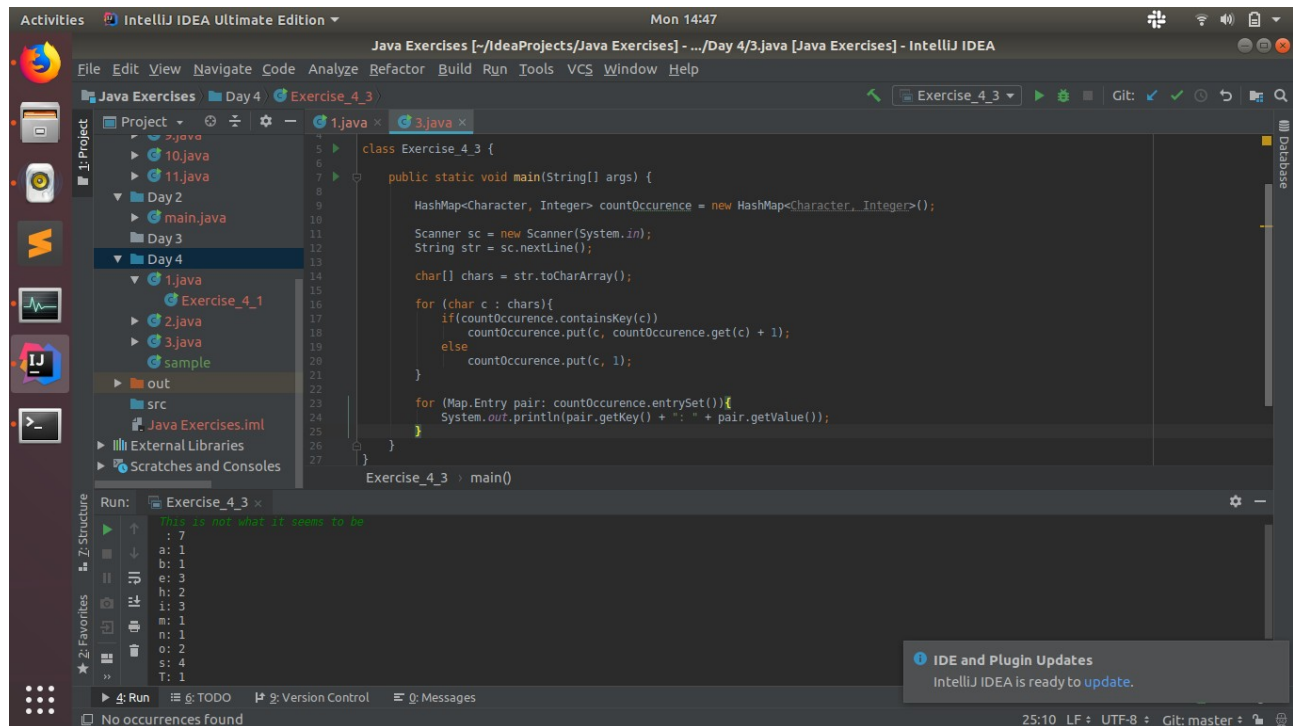
```
1 import java.util.Arrays;
2 import java.util.HashSet;
3 import java.util.Set;
4
5 class Exercise_4_2 {
6
7     public static void main(String[] args) {
8         String str = "The quick brown fox jumps over the lazy dog";
9
10        System.out.println(str.toCharArray());
11        Set<Character> unique = new HashSet<>();
12
13        for (Character c : str.toCharArray()){
14            unique.add(c);
15        }
16
17        System.out.println(unique);
18    }
19 }
20
21 }
22
```

Run: Exercise_4_2 x
/usr/lib/jvm/java-8-oracle/bin/java ...
The quick brown fox jumps over the lazy dog
[T, , a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z]
Process finished with exit code 0

IDE and Plugin Updates
IntelliJ IDEA is ready to update.

Q3. Write a method that takes a string and print the number of occurrence of each character characters in the string.

A3.



The screenshot shows the IntelliJ IDEA interface with a Java file named `Exercise_4_3.java` open. The code defines a class `Exercise_4_3` with a `main` method. It uses a `HashMap` to store the frequency of each character in a string entered by the user. The output shows the frequency of each character in the input string "a b c d e f g h i j k l m n o p q r s t u v w x y z".

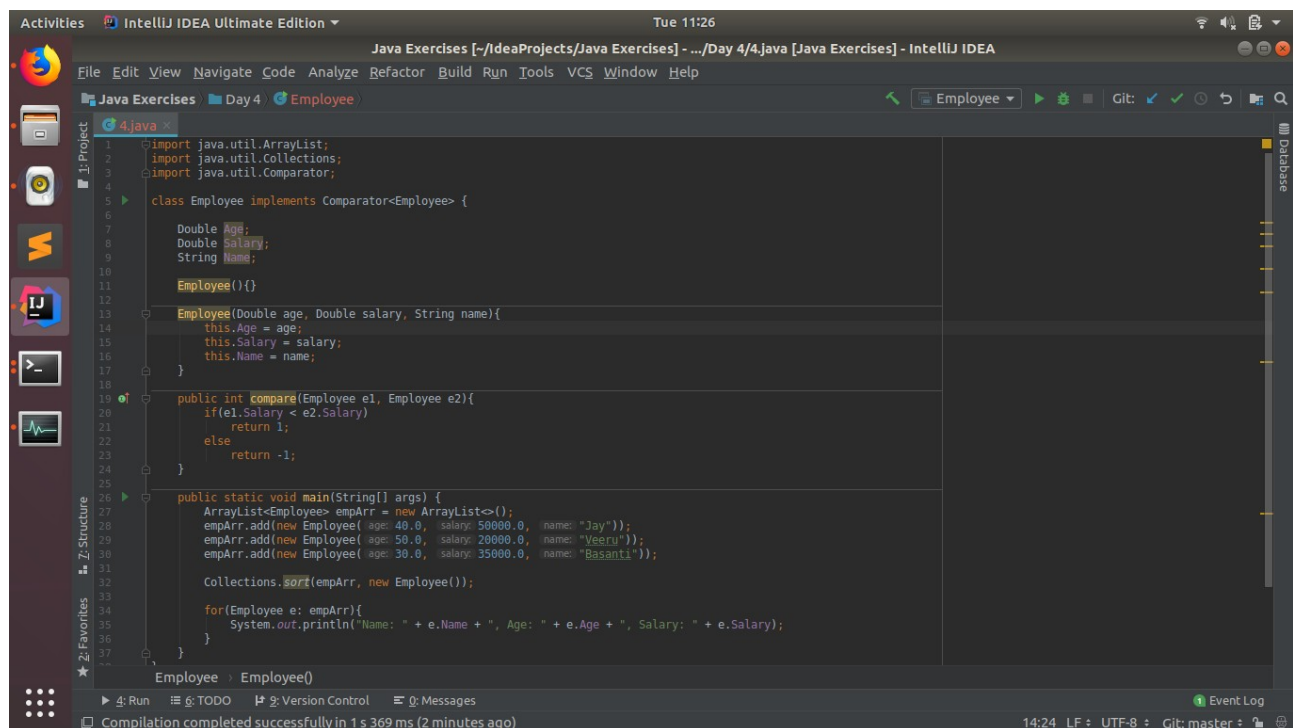
```
class Exercise_4_3 {
    public static void main(String[] args) {
        HashMap<Character, Integer> countOccurrence = new HashMap<Character, Integer>();
        Scanner sc = new Scanner(System.in);
        String str = sc.nextLine();
        char[] chars = str.toCharArray();
        for (char c : chars) {
            if (countOccurrence.containsKey(c)) {
                countOccurrence.put(c, countOccurrence.get(c) + 1);
            } else {
                countOccurrence.put(c, 1);
            }
        }
        for (Map.Entry pair : countOccurrence.entrySet()) {
            System.out.println(pair.getKey() + " : " + pair.getValue());
        }
    }
}
```

The Run window shows the output of the program:

```
This is not what it seems to be
: 7
a: 1
b: 1
c: 3
d: 2
e: 3
f: 1
g: 1
h: 1
i: 3
j: 1
k: 1
l: 1
m: 1
n: 1
o: 2
p: 2
q: 4
r: 1
s: 1
t: 1
```

Q4. Write a program to sort Employee objects based on highest salary using Comparator. Employee class{ Double Age; Double Salary; String Name

A4.



The screenshot shows the IntelliJ IDEA interface with a Java file named `Employee.java` open. The code defines an `Employee` class that implements the `Comparable` interface. It uses a `Comparator` to sort an array of `Employee` objects based on their salary. The output shows the sorted list of employees by salary.

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;

class Employee implements Comparable<Employee> {
    Double Age;
    Double Salary;
    String Name;

    Employee() {}

    Employee(Double age, Double salary, String name) {
        this.Age = age;
        this.Salary = salary;
        this.Name = name;
    }

    public int compare(Employee e1, Employee e2) {
        if (e1.Salary < e2.Salary) {
            return 1;
        } else {
            return -1;
        }
    }

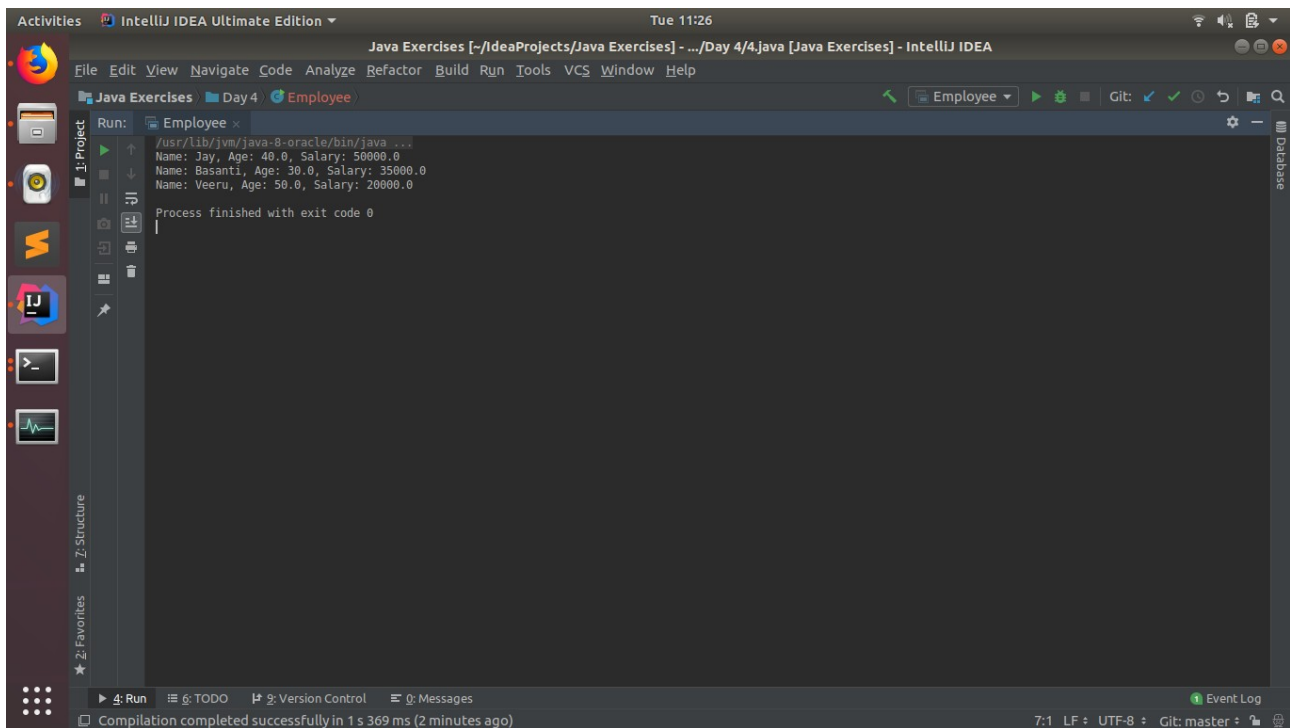
    public static void main(String[] args) {
        ArrayList<Employee> empArr = new ArrayList<>();
        empArr.add(new Employee(40.0, salary: 50000.0, name: "Jay"));
        empArr.add(new Employee(50.0, salary: 20000.0, name: "Veeru"));
        empArr.add(new Employee(30.0, salary: 35000.0, name: "Basanti"));

        Collections.sort(empArr, new Employee());

        for (Employee e : empArr) {
            System.out.println("Name: " + e.Name + ", Age: " + e.Age + ", Salary: " + e.Salary);
        }
    }
}
```

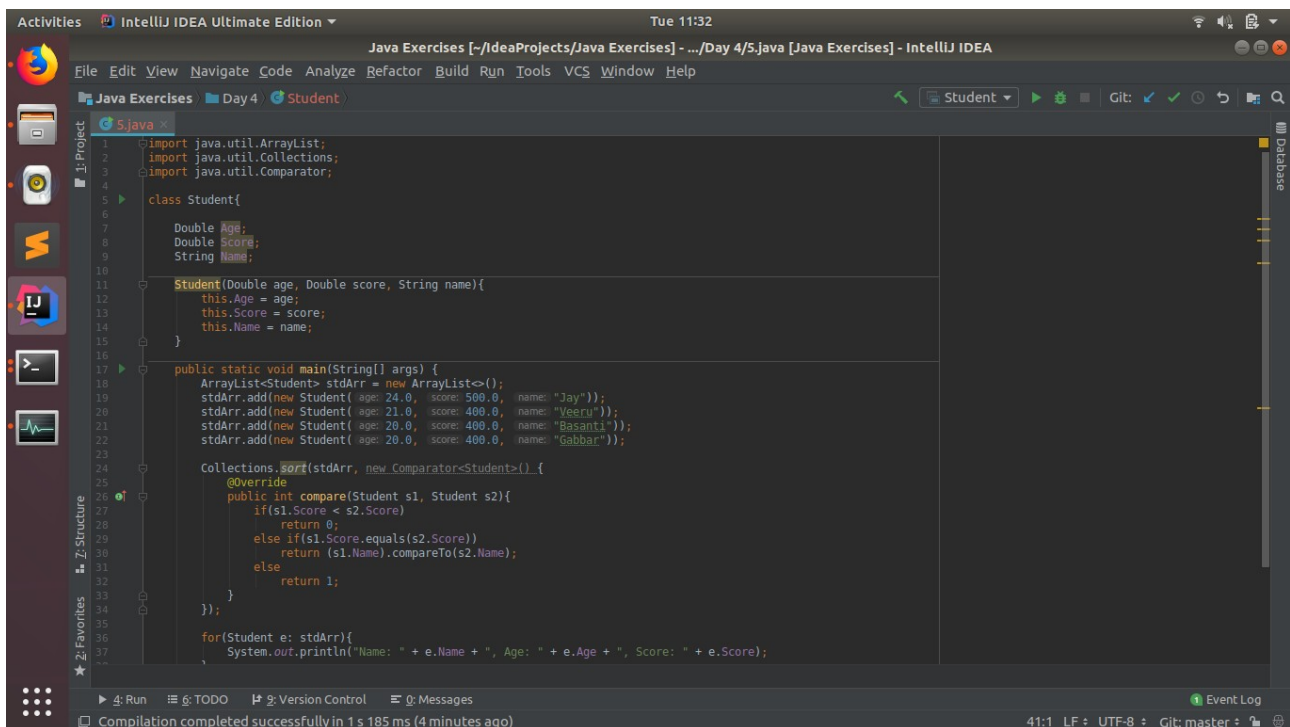
The Run window shows the output of the program:

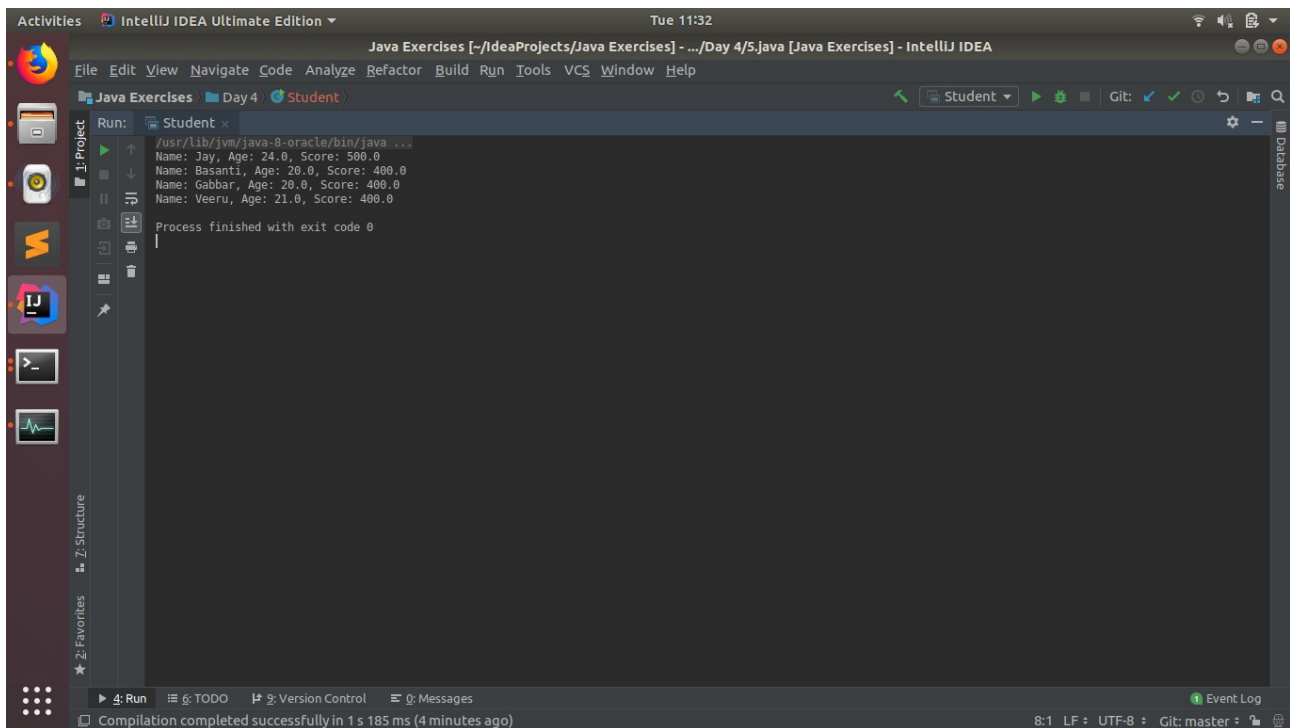
```
Employee > Employee()
Compilation completed successfully in 1 s 369 ms (2 minutes ago)
```



Q5. 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

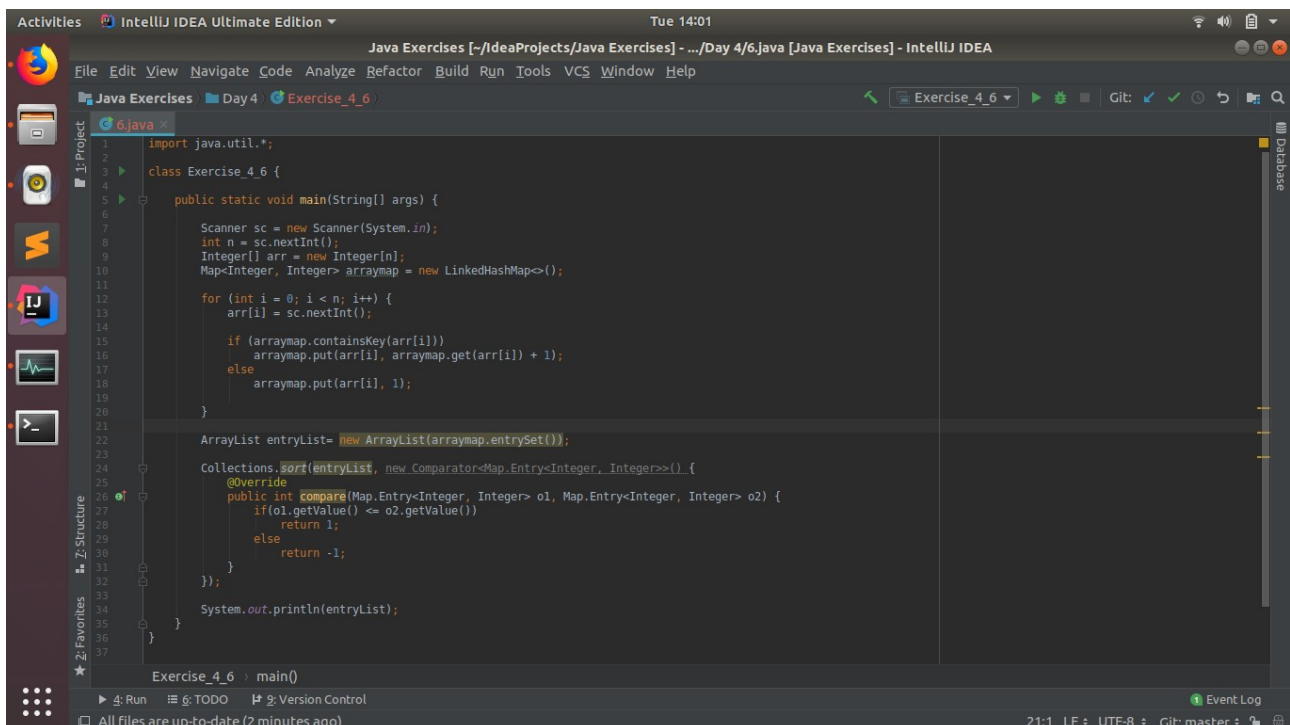
A5.

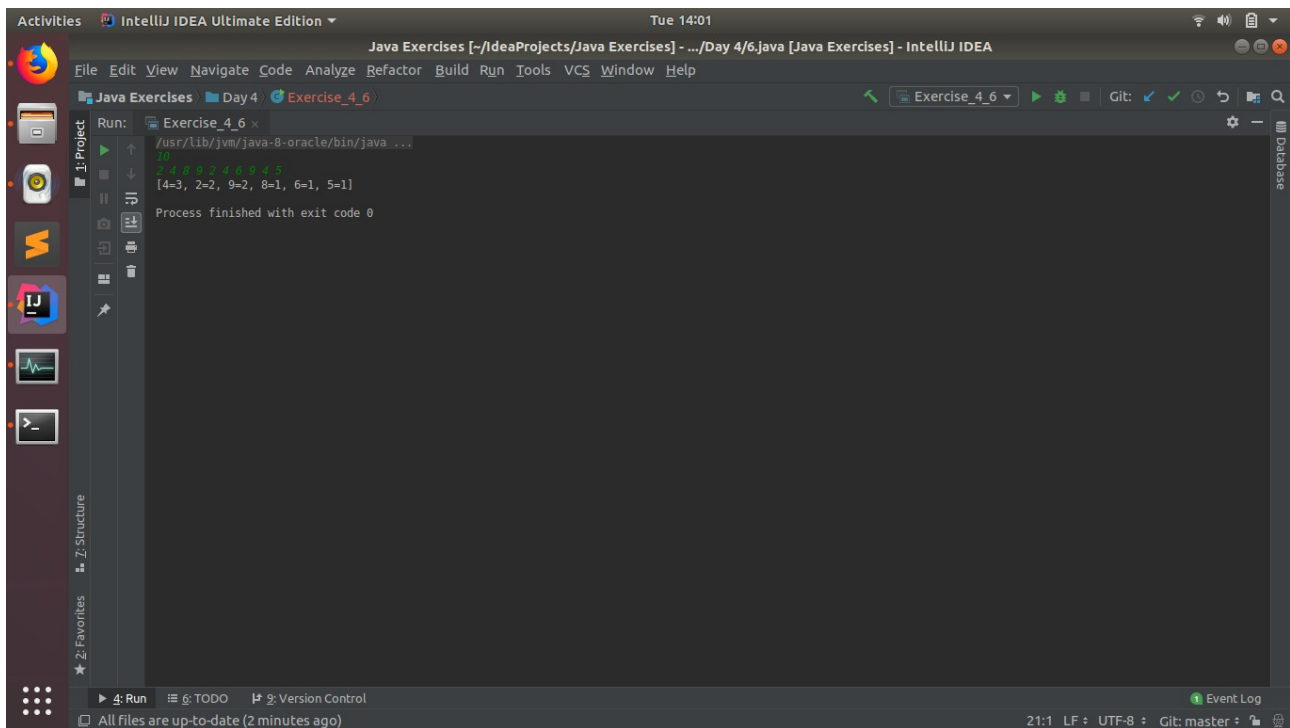




Q6. Print the elements of an array in the decreasing frequency if 2 numbers have same frequency then print the one which came first.

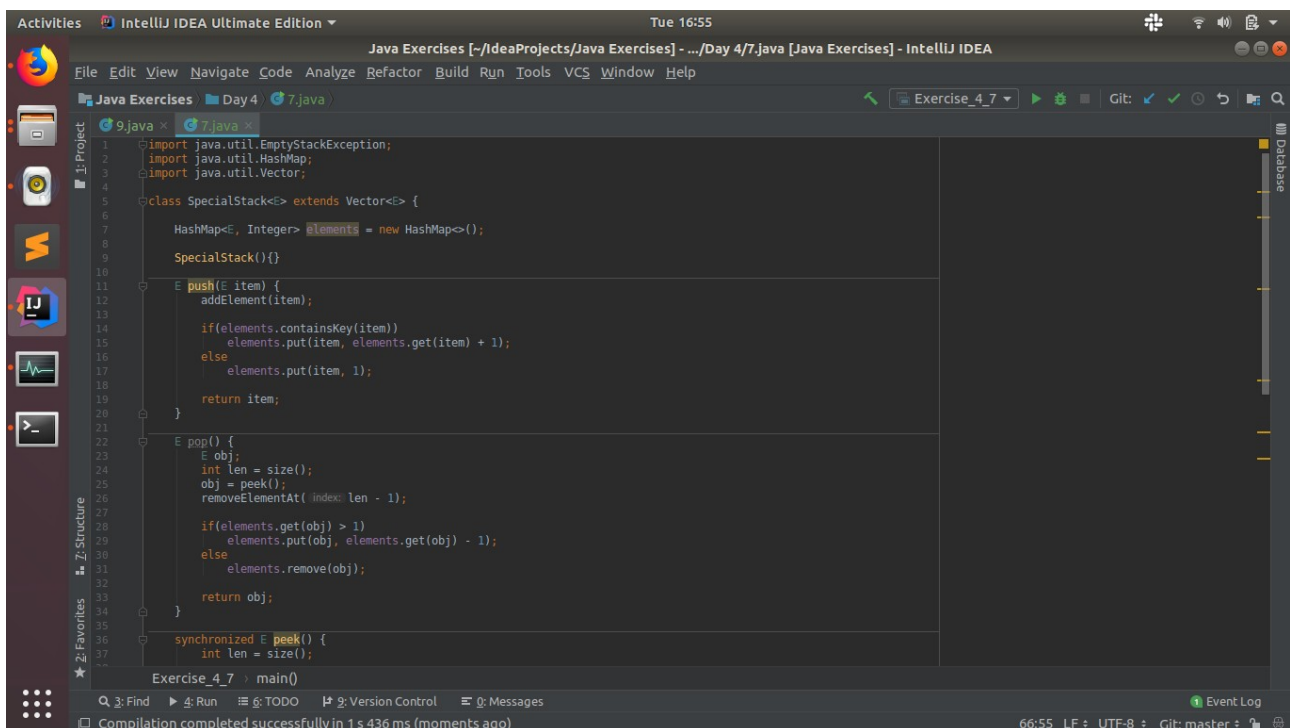
A6.

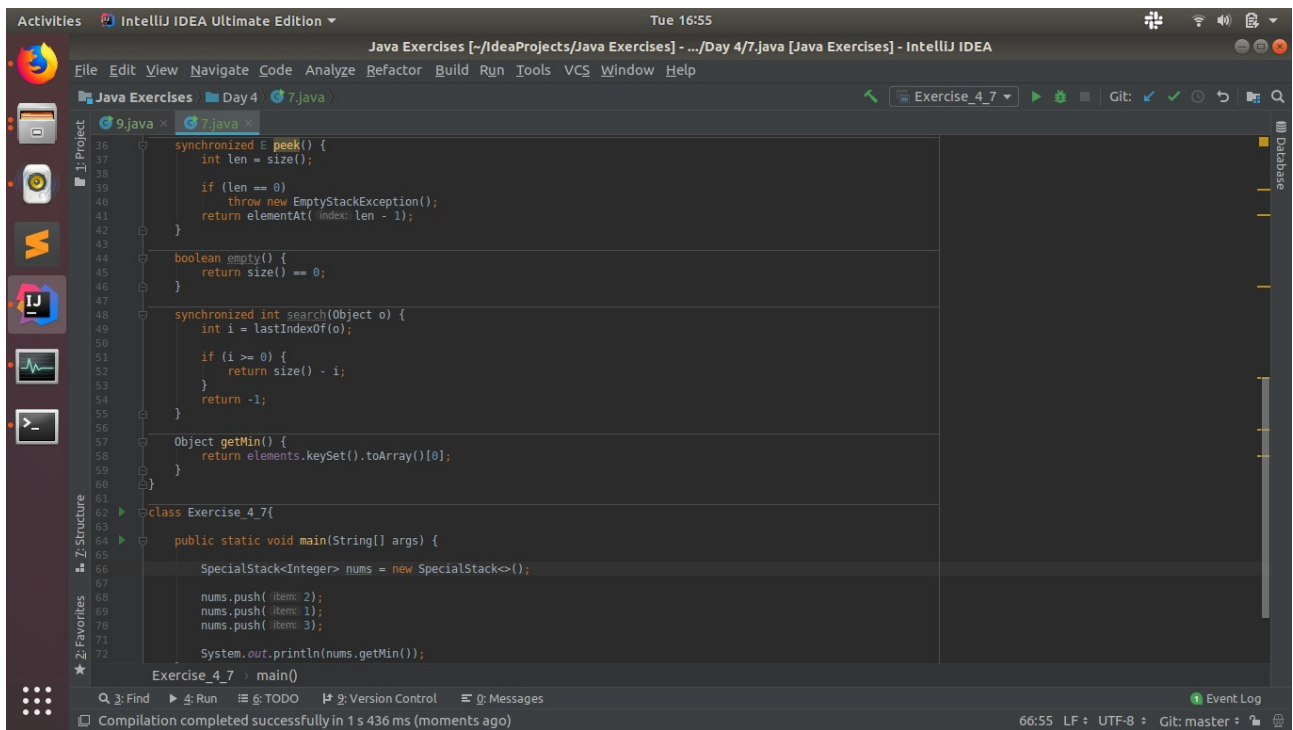




Q7. Design a Data Structure SpecialStack that supports all the stack operations like push(), pop(), isEmpty(), isFull() and an additional operation getMin() which should return minimum element from the SpecialStack. (Expected complexity $O(1)$)

A7.

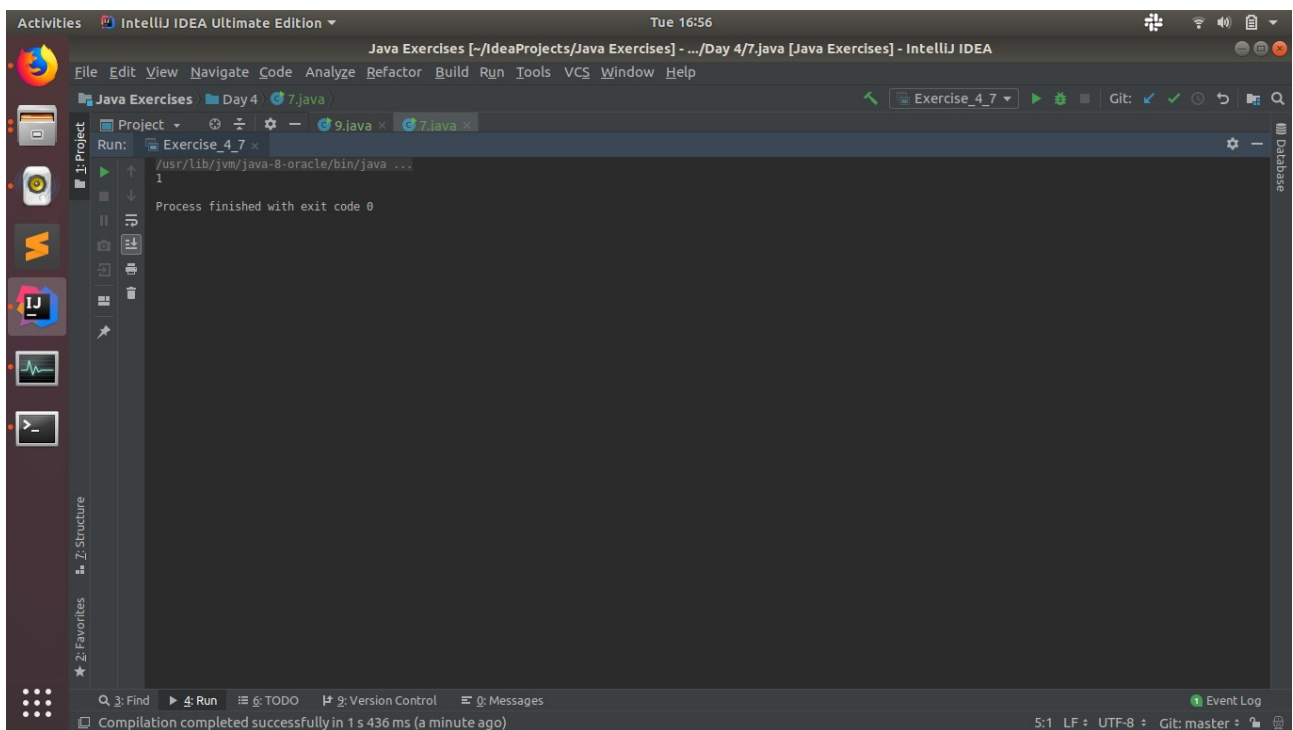




```
36 synchronized E peek() {
37     int len = size();
38
39     if (len == 0)
40         throw new EmptyStackException();
41     return elementAt(index: len - 1);
42 }
43
44 boolean empty() {
45     return size() == 0;
46 }
47
48 synchronized int search(Object o) {
49     int i = lastIndexOf(o);
50
51     if (i >= 0) {
52         return size() - i;
53     }
54     return -1;
55 }
56
57 Object getMin() {
58     return elements.keySet().toArray()[0];
59 }
60
61
62 class Exercise_4_7 {
63
64     public static void main(String[] args) {
65
66         SpecialStack<Integer> nums = new SpecialStack<>();
67
68         nums.push( (Item<> 2);
69         nums.push( (Item<> 1);
70         nums.push( (Item<> 3);
71
72         System.out.println(nums.getMin());
73     }
74 }
```

Exercise_4_7 -> main()

Compilation completed successfully in 1 s 436 ms (moments ago)

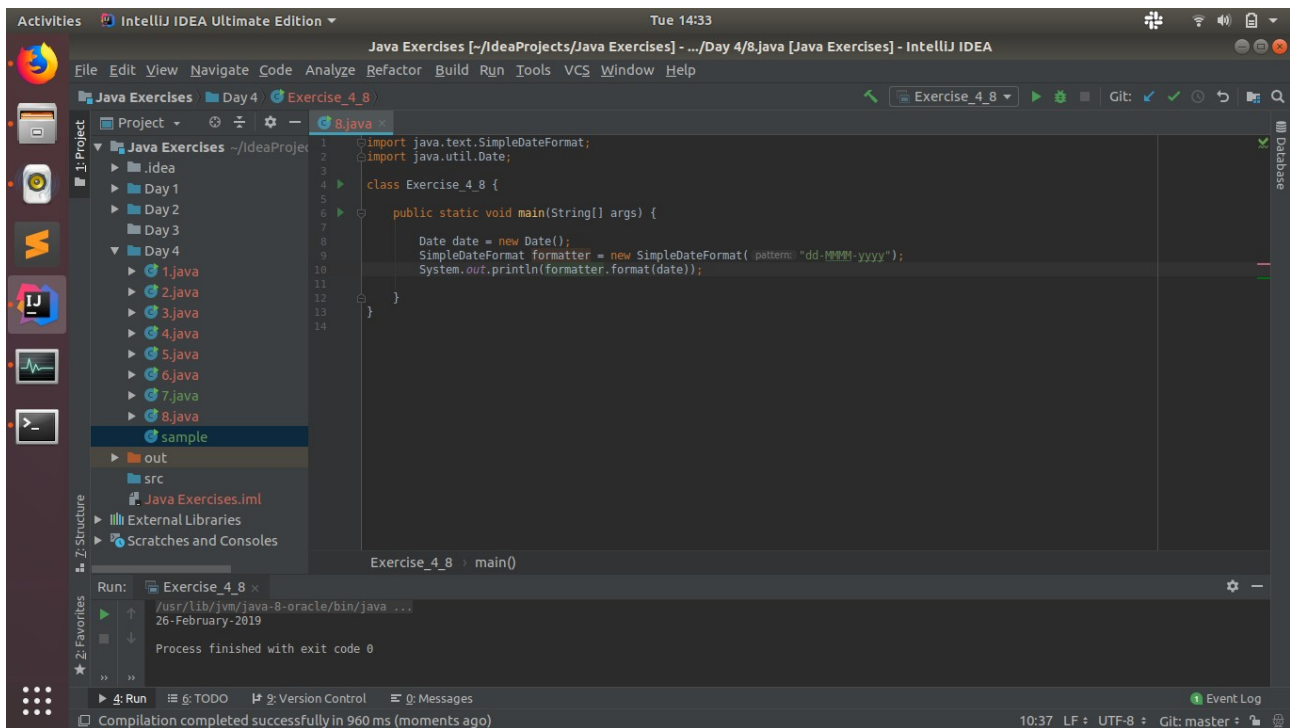


```
Run: Exercise_4_7
/usr/lib/jvm/java-8-oracle/bin/java ...
1
Process finished with exit code 0
```

Compilation completed successfully in 1 s 436 ms (a minute ago)

Q8. Write a program to format date as example "21-March-2016"

A8.



Q9. Write a program to display times in different country format.

A9.

