

ASSIGNMENT JAVA DAY12

Harshit Kushmakar | 16896

1. Create a class Employee with three data members as given below:

Create an ArrayList of the above-mentioned class and perform the below operations on the same.

a. Add Employee objects in the list.

b. Remove the object at a particular Index.

c. Print the list using the below methods:

i. Using for loop

ii. Using for-each loop

iii. Using Iterator Interface

iv. Print backward using ListIterator interface.

d. Remove all the instances from the list who have salary less than 10000.

Use Iterator interface to perform the functionality.

```
package assignment12;

public class Employee {
    int empID;
    String empName;
    double salary;
    public Employee(int empID, String empName, double salary) {
        this.empID = empID;
        this.empName = empName;
        this.salary = salary;
    }
}
```

```
package assignment12;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.ListIterator;

public class main {

    public static void main(String[] args) {
        // Create an ArrayList of Employee objects
        ArrayList<Employee> Employees = new ArrayList<>();
    }
}
```

```

Employees.add(new Employee(1, "John", 15000));
Employees.add(new Employee(2, "Jane", 20000));
Employees.add(new Employee(3, "Mike", 8000));
Employees.add(new Employee(4, "Sarah", 12000));

Employees.remove(2);

// Print the list using for loop
System.out.println("Using for loop:");
for (int i = 0; i < Employees.size(); i++) {
    Employee emp = Employees.get(i);
    System.out.println(emp.empID + " " + emp.empName + " " +
emp.salary);
}

// Print the list using for-each loop
System.out.println("Using for-each loop:");
for (Employee emp : Employees) {
    System.out.println(emp.empID + " " + emp.empName + " " +
emp.salary);
}

// Print the list using Iterator interface
System.out.println("Using Iterator interface:");
Iterator<Employee> iterator = Employees.iterator();
while (iterator.hasNext()) {
    Employee emp = iterator.next();
    System.out.println(emp.empID + " " + emp.empName + " " +
emp.salary);
}

// Print the list backward using ListIterator interface
System.out.println("Print backward using ListIterator interface:");
ListIterator<Employee> listIterator =
Employees.listIterator(Employees.size());
while (listIterator.hasPrevious()) {
    Employee emp = listIterator.previous();
    System.out.println(emp.empID + " " + emp.empName + " " +
emp.salary);
}

// Remove all the instances from the list who have salary less than
10000 using Iterator interface
Iterator<Employee> itr = Employees.iterator();
while (itr.hasNext()) {
    Employee emp = itr.next();
    if (emp.salary < 10000) {
        itr.remove();
    }
}

// Print the updated list
System.out.println("Updated list:");
for (Employee emp : Employees) {
    System.out.println(emp.empID + " " + emp.empName + " " +
emp.salary);
}
}

```

```
"C:\Program Files\Java\jdk-11\bin\java.exe" "-javaagent:C:\Program Files\Jet
Using for loop:
1 John 15000.0
2 Jane 20000.0
4 Sarah 12000.0
Using for-each loop:
1 John 15000.0
2 Jane 20000.0
4 Sarah 12000.0
Using Iterator interface:
1 John 15000.0
2 Jane 20000.0
4 Sarah 12000.0
Print backward using ListIterator interface:
4 Sarah 12000.0
2 Jane 20000.0
1 John 15000.0
Updated list:
1 John 15000.0
2 Jane 20000.0
4 Sarah 12000.0
```

2. Using the above class, what will be the output of the below mentioned code snippet? Share your findings with the help of comments. What should be done to resolve the problem if any?

The output of the code snippet will be 2, which is the size of the empList after removing the Employee object with empID 5 and empName "A" and salary 45000.

To resolve this problem, we can override the equals() and hashCode() methods in the Employee class to check for field equality.

3. Sort the above list on empId using Comparable interface.

```
package assignment12;

public class Employee implements Comparable<Employee>{
    int empID;
    String empName;
    double salary;

    public int getEmpID() {
        return empID;
    }

    public void setEmpID(int empID) {
        this.empID = empID;
    }

    public String getEmpName() {
        return empName;
    }

    public void setEmpName(String empName) {
        this.empName = empName;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {
        this.salary = salary;
    }

    public Employee(int empID, String empName, double salary) {
        this.empID = empID;
        this.empName = empName;
        this.salary = salary;
    }

    @Override
    public String toString() {
        return "Employee{" +
            "empID=" + empID +
            ", empName='" + empName + '\'' +
            ", salary=" + salary +
            '}';
    }

    @Override
    public int compareTo(Employee other) {
        return Integer.compare(this.empID, other.empID);
    }
}
```

```

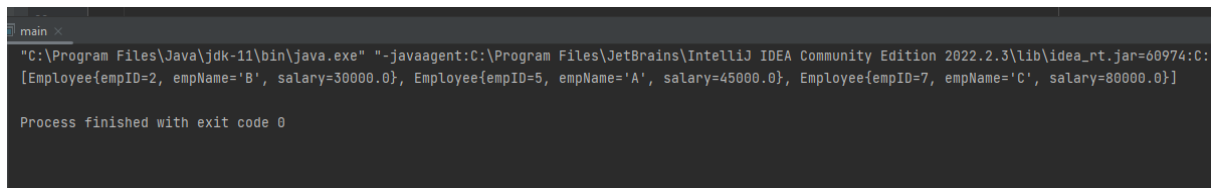
public class main{
    public static void main(String[] args) {
        List<Employee> empList = new ArrayList<Employee>();
        empList.add(new Employee(5,"A",45000));
        empList.add(new Employee(2,"B",30000));
        empList.add(new Employee(7,"C",80000));

        Collections.sort(empList);

        System.out.println(empList);

    }
}

```



```

main x
"C:\Program Files\Java\jdk-11\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.2.3\lib\idea_rt.jar=60974:C:
[Employee{empID=2, empName='B', salary=30000.0}, Employee{empID=5, empName='A', salary=45000.0}, Employee{empID=7, empName='C', salary=80000.0}]

Process finished with exit code 0

```

4. Sort the same list again on basis of salary using Comparator Interface. Show the implementation of Comparator with the help of

a. Anonymous Inner Class

b. Lambda Expression

```

package assignment12;

public class Employee implements Comparable<Employee>{
    int empID;
    String empName;
    double salary;

    public int getEmpID() {
        return empID;
    }

    public void setEmpID(int empID) {
        this.empID = empID;
    }

    public String getEmpName() {
        return empName;
    }

    public void setEmpName(String empName) {
        this.empName = empName;
    }

    public double getSalary() {
        return salary;
    }

    public void setSalary(double salary) {

```

```

        this.salary = salary;
    }

    public Employee(int empID, String empName, double salary) {
        this.empID = empID;
        this.empName = empName;
        this.salary = salary;
    }

    @Override
    public String toString() {
        return "Employee{" +
            "empID=" + empID +
            ", empName='" + empName + '\'' +
            ", salary=" + salary +
            '}';
    }

    @Override
    public int compareTo(Employee other) {
        return Integer.compare(this.empID, other.empID);
    }
}

```

```

public class main {

    public static void main(String[] args) {
        List<Employee> empList = new ArrayList<>();
        empList.add(new Employee(5, "A", 45000));
        empList.add(new Employee(2, "B", 30000));
        empList.add(new Employee(7, "C", 80000));

        // Sort by empID using Comparable
        Collections.sort(empList);

        // Sort by salary using anonymous inner class
        Collections.sort(empList, new Comparator<Employee>() {
            @Override
            public int compare(Employee e1, Employee e2) {
                return Double.compare(e1.getSalary(), e2.getSalary());
            }
        });

        System.out.println("Sorted by salary using anonymous inner
class:");
        System.out.println(empList);

        // Sort by salary using lambda expression
        Collections.sort(empList, (e1, e2) ->
Double.compare(e1.getSalary(), e2.getSalary()));

        System.out.println("Sorted by salary using lambda
expression:");
        System.out.println(empList);
    }
}

```

```
}  
}
```

```
@Override  
public int compare(Employee e1, Employee e2) {  
  
}  
}
```

main

"C:\Program Files\Java\jdk-11\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2022.2.3\lib\idea_rt.jar=53992:C:\Program Files\Java\jdk-11\bin" -Dfile.encoding=UTF-8

Sorted by salary using anonymous inner class:
[Employee{empID=2, empName='B', salary=30000.0}, Employee{empID=5, empName='A', salary=45000.0}, Employee{empID=7, empName='C', salary=80000.0}]

Sorted by salary using lambda expression:
[Employee{empID=2, empName='B', salary=30000.0}, Employee{empID=5, empName='A', salary=45000.0}, Employee{empID=7, empName='C', salary=80000.0}]

Process finished with exit code 0

5. Change the above code to use LinkedList instead of ArrayList and perform the same operations.

```
public class main {  
  
    public static void main(String[] args) {  
        List<Employee> empList = new LinkedList<>();  
        empList.add(new Employee(5, "A", 45000));  
        empList.add(new Employee(2, "B", 30000));  
        empList.add(new Employee(7, "C", 80000));  
  
        // Remove the object at a particular index  
        empList.remove(1);  
  
        // Print the list using for loop  
        System.out.println("Using for loop:");  
        for (int i = 0; i < empList.size(); i++) {  
            System.out.println(empList.get(i));  
        }  
  
        // Print the list using for-each loop  
        System.out.println("Using for-each loop:");  
        for (Employee emp : empList) {  
            System.out.println(emp);  
        }  
  
        // Print the list using Iterator interface  
        System.out.println("Using Iterator interface:");  
        Iterator<Employee> empIterator = empList.iterator();  
        while (empIterator.hasNext()) {  
            System.out.println(empIterator.next());  
        }  
  
        // Print backward using ListIterator interface  
        System.out.println("Print backward using ListIterator interface:");  
    }  
}
```

```

        ListIterator<Employee> listIterator =
empList.listIterator(empList.size());
        while (listIterator.hasPrevious()) {
            System.out.println(listIterator.previous());
        }

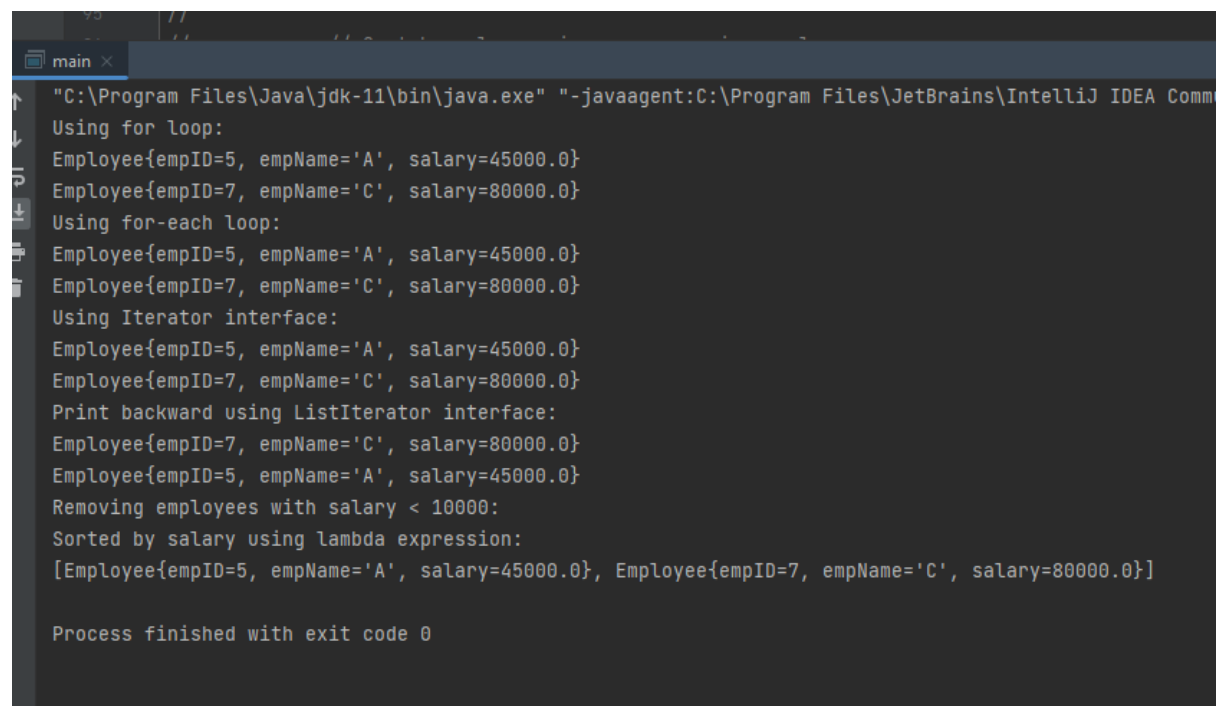
        // Remove all the instances from the list who have salary less than
10000
        System.out.println("Removing employees with salary < 10000:");
        Iterator<Employee> itr = empList.iterator();
        while (itr.hasNext()) {
            Employee emp = itr.next();
            if (emp.getSalary() < 10000) {
                itr.remove();
            }
        }

        // Sort by empID using Comparable
        Collections.sort(empList);

        // Sort by salary using lambda expression
        Collections.sort(empList, (e1, e2) ->
Double.compare(e1.getSalary(), e2.getSalary()));

        System.out.println("Sorted by salary using lambda expression:");
        System.out.println(empList);
    }
}

```



```

main x
"C:\Program Files\Java\jdk-11\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Comm
Using for loop:
Employee{empID=5, empName='A', salary=45000.0}
Employee{empID=7, empName='C', salary=80000.0}
Using for-each loop:
Employee{empID=5, empName='A', salary=45000.0}
Employee{empID=7, empName='C', salary=80000.0}
Using Iterator interface:
Employee{empID=5, empName='A', salary=45000.0}
Employee{empID=7, empName='C', salary=80000.0}
Print backward using ListIterator interface:
Employee{empID=7, empName='C', salary=80000.0}
Employee{empID=5, empName='A', salary=45000.0}
Removing employees with salary < 10000:
Sorted by salary using lambda expression:
[Employee{empID=5, empName='A', salary=45000.0}, Employee{empID=7, empName='C', salary=80000.0}]

Process finished with exit code 0

```


6. Create a Java program that creates a Set using a HashSet implementation. The Set stores fruit names and the fruits must be added in the following sequence:

Pear

Banana

Tangerine

Strawberry

Blackberry

The functionalities which must be exhibited by the program are:

- a. Display the contents of set.**
- b. Display the number of elements in the set.**
- c. Remove blackberry and strawberry, display contents of set again.**
- d. Remove the remaining fruits using a single method invocation.**
- e. Show the set is now empty.**

→ While displaying the set contents, explain why the fruits don't get listed in the order in which they were added.

```
package assignment12;

import java.util.HashSet;
import java.util.Set;

public class FruitSet {
    public static void main(String[] args) {
        Set<String> fruitSet = new HashSet<>();
        fruitSet.add("Pear");
        fruitSet.add("Banana");
        fruitSet.add("Tangerine");
        fruitSet.add("Strawberry");
        fruitSet.add("Blackberry");

        System.out.println("Contents of set: " + fruitSet);

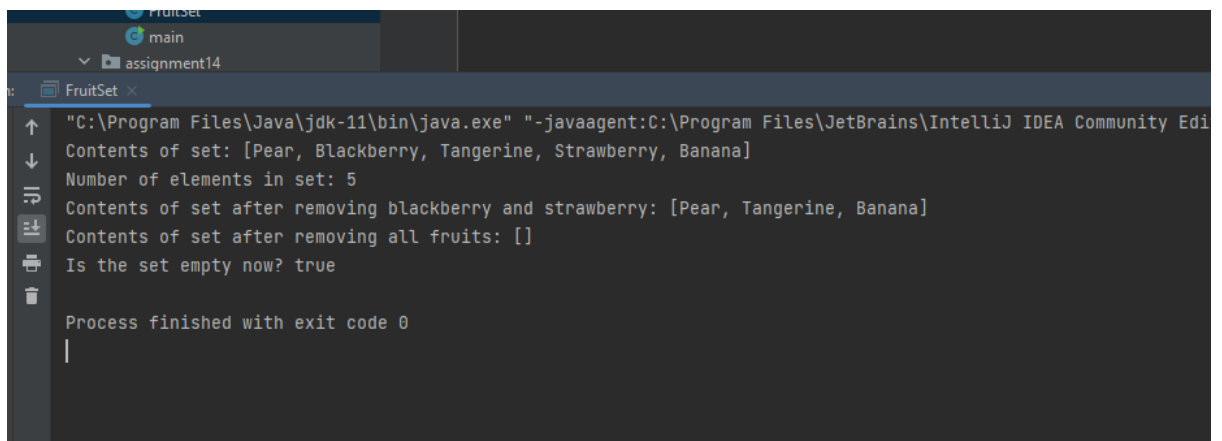
        System.out.println("Number of elements in set: " +
            fruitSet.size());

        // Remove blackberry and strawberry
        fruitSet.remove("Blackberry");
        fruitSet.remove("Strawberry");
        System.out.println("Contents of set after removing blackberry and
```

```
strawberry: " + fruitSet);

    // Remove remaining fruits
    fruitSet.clear();
    System.out.println("Contents of set after removing all fruits: " +
fruitSet);

    // empty
    System.out.println("Is the set empty now? " + fruitSet.isEmpty());
}
}
```



```
"C:\Program Files\Java\jdk-11\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edi
Contents of set: [Pear, Blackberry, Tangerine, Strawberry, Banana]
Number of elements in set: 5
Contents of set after removing blackberry and strawberry: [Pear, Tangerine, Banana]
Contents of set after removing all fruits: []
Is the set empty now? true

Process finished with exit code 0
|
```