Java Assesment - II

Provide solutions for the below problem statements.

1. Stream API and collection:

Using below dataset, implement streaming intermediate and terminal operators for the below requirements

sample dataset.

("Messi", 32, Gender.MALE, ("CF","CAM", "RF")),
("Griezmann", 28, Gender.MALE, ("CF", "CAM", "LF")),
("Arthur", 23, Gender.MALE, ("CM", "CAM")),
("Ter Stegen", 27, Gender.MALE, ("GK")),
("Puig", 20, Gender.MALE, ("CM", "CDM")),
("Jennifer", 29, Gender.FEMALE, ("CF", "CAM")),
("Jana", 17, Gender.FEMALE, ("CB")),
("Alexia", 25, Gender.FEMALE, ("CAM", "RF", "LF"))

A) Fecth the data gender value is matching with female and age more than 23.
B) find the count of footballers who are females and age more than 23.
C) get the position of male footballers who are the males and their age greater than 30 years.
D) display the first theree footballers who are male and age greater than 25.
E) find the aggregate values of age of all male and female footbalers.
F) using list of elements (4, 1, 3, 7, 5, 6, 2, 28, 15, 29) find first element which is greater that 5.

Solution:

```
package newww;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.DoubleSummaryStatistics;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;

class dataset {
	String name;
	int age;
	Gender gender ;
	List<String> Football;
  dataset(String name, int age, Gender gender, List<String>
football) {
		this.name = name;
		this.age = age;
		this.gender = gender;
		Football = football;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public int getAge() {
		return age;
	}
	public void setAge(int age) {
		this.age = age;
```

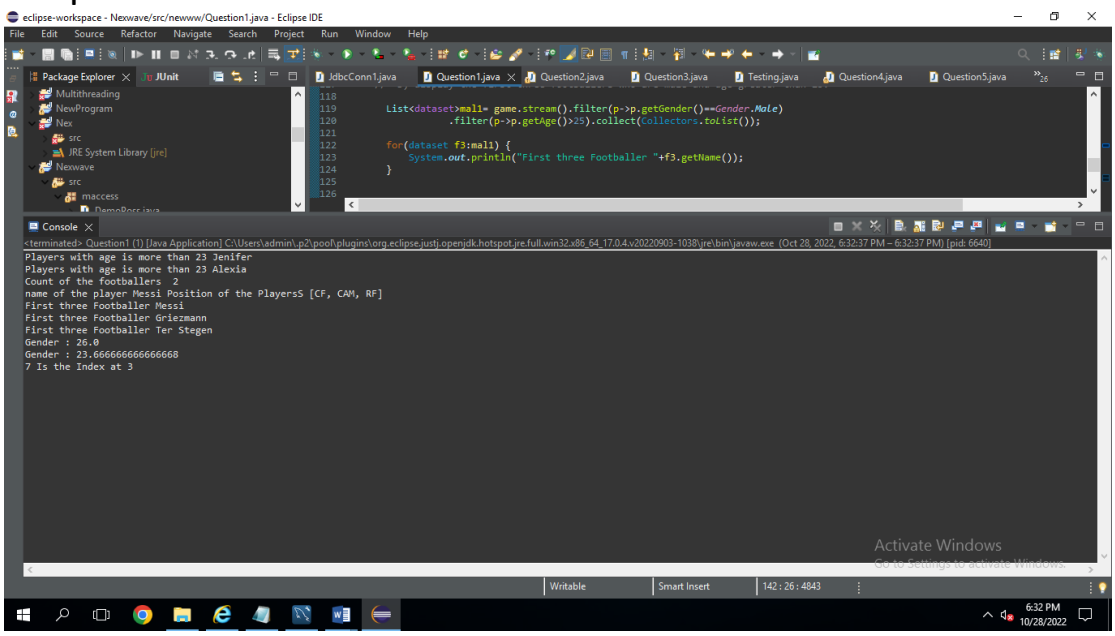
```
	}
	public Gender getGender() {
		return gender;
	}
	public void setGender(Gender gender) {
		this.gender = gender;
	}
	public List<String> getFootball() {
		return Football;
	}
	public void setFootball(ArrayList<String> football) {
		Football = football;
	}



}
enum Gender {
	Male, Female
}

public class Question1 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		ArrayList<dataset>game=new ArrayList<>();

		game.add(new
dataset("Messi",32,Gender.Male ,Arrays.asList("CF","CAM",
"RF")));
		game.add(new dataset("Griezmann", 28,
Gender.Male,Arrays.asList("CF", "CAM", "LF")));
		game.add(new dataset("Arthur", 23,
Gender.Male,Arrays.asList("CM", "CAM")));
```

```
            game.add(new dataset("Ter Stegen", 27,
Gender.Male,Arrays.asList("GK")));
            game.add(new dataset("Puig", 20,
Gender.Male,Arrays.asList ("CM", "CDM")));
            game.add(new dataset("Jenifer", 29,
Gender.Female,Arrays.asList ("CF", "CAM")));
            game.add(new dataset("Jana", 17,
Gender.Female,Arrays.asList ("CB")));
          game.add(new dataset("Alexia", 25,
Gender.Female,Arrays.asList ("CAM", "RF", "LF")));

         // A) Fetch the data gender value is matching with
female  and age more than 23.
       List<dataset>lis=game.stream().filter(p-
>p.getGender()==Gender.Female)
                       .filter(p-
>p.getAge()>23).collect(Collectors.toList());


       for(dataset lis1 : lis) {
            System.out.println("Players with age is more than 23
"+lis1.getName());
       }

       //B) find  the count of footballers who are females and
age more than 23.

       System.out.println("Count of the footballers  "+lis.size());


       //C) get the position of male footballers who are the
males and their age greater than 30 years.

        List<dataset>mal= game.stream().filter(p-
>p.getGender()==Gender.Male)
```

```
                        .filter(p->p.getAge()>30).collect(Collectors.toList());
        for(dataset males:mal) {
                System.out.println("name of the player "+males.getName()+" Position of the PlayersS "+males.getFootball());
        }


       //  D) display the first three footballers who are male and age greater than 25.

        List<dataset>mal1= game.stream().filter(p->p.getGender()==Gender.Male)
                                .filter(p->p.getAge()>25).collect(Collectors.toList());

        for(dataset f3:mal1) {
                System.out.println("First three Footballer "+f3.getName());
        }


        //E) find the aggregate values of age of all male and female footballers.

        Map<Gender, DoubleSummaryStatistics>val=game.stream().collect(Collectors.groupingBy(p->p.getGender(),Collectors.summarizingDouble(p->p.getAge())));

        Set<Gender>gen=val.keySet();
        for(Gender gens:gen) {
                System.out.println("Gender : "+ val.get(gens).getAverage());
```

```
        }

        // F) using list of elements (4, 1, 3, 7, 5, 6, 2, 28, 15, 29)
find first element which is greater that 5.
                List<Integer>ints=List.of(4, 1, 3, 7, 5, 6, 2, 28, 15,
29);

                for(int i=0;i<ints.size();i++) {
                        if(ints.get(i)>5) {
                                System.out.println(ints.get(i)+" Is the
Index at "+i);
                                break;
                        }
                }

        }
    }
```

Output:



2. **JDBC**

In Mysql db create the table with below structure

Tbl Name: Orders
Field names: OrderId,Name,size,price,quantity,date

Using JDBC implement the following

1. Feed the data at least 5 records .
2. Fetch the total amount of orders delivered and minimum quantity for each name.

Once the above 2 completed

1.Feed the additional data with 3 records and having 1 duplicate.

In case of possible exceptions in both scenarios implement exception handling.

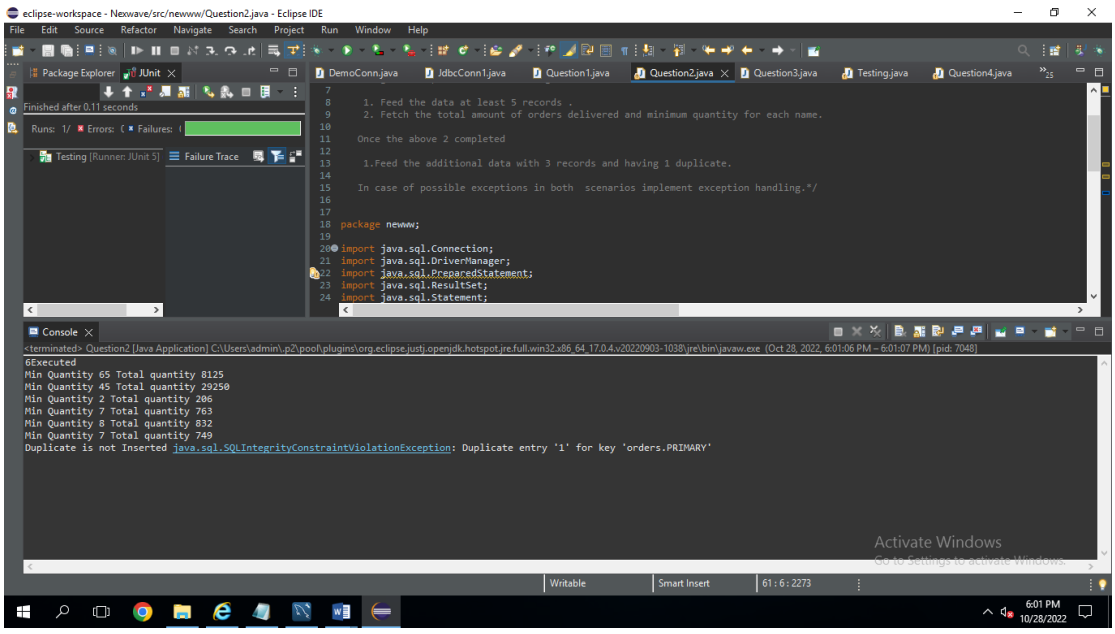
Solution:

```
package newww;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

import com.mysql.cj.protocol.Resultset;

public class Question2 {
```

```
public static void main(String[] args) throws Exception{
        // TODO Auto-generated method stub
        String url="jdbc:mysql://localhost:3306/nexwave";
        String uname="root";
        String pass="admin";
        String Query="insert into orders
values(1,'praveen','medium',125,65,'2022-08-
13'),(15,'Sridhar','small',650,45,'2021-08-
17'),(02,'Srihari','big',103,2,'2022-08-
25'),(3,'Hari','medium',109,7,'2022-08-
31'),(6,'Sriii','big',107,7,'2021-08-
28'),(5,'vishnu','small',104,8,'2022-07-28')";
        String Min="select
min(quantity),sum(quantity)*price from orders group by name";
        String Dub="insert into orders
values(1,'praveen','medium',125,65,'2022-08-
13'),(15,'Sridhar','small',650,45,'2021-08-
17'),(02,'Srihari','big',103,2,'2022-08-
25'),(3,'Hari','medium',109,7,'2022-08-
31'),(6,'Sriii','big',107,7,'2021-08-
28'),(5,'vishnu','small',104,8,'2022-07-28')";


            Class.forName("com.mysql.cj.jdbc.Driver");
            Connection
conn=DriverManager.getConnection(url,uname,pass);
            Statement stmt=conn.createStatement();
              int sm=stmt.executeUpdate(Query);


              //feed 5 data to the table
          System.out.println(sm+"Executed");

          ResultSet rs=stmt.executeQuery(Min);
              while(rs.next()) {
```

```
                    System.out.println("Min Quantity
"+rs.getInt(1)+" Total quantity "+rs.getInt(2));
                }
                try {
                int du=stmt.executeUpdate(Dub);
                System.out.println(du+"duplicate Inserted");}
                catch(Exception e) {
                    System.out.println("Duplicate is not
Inserted "+e);
                }
        }

        }
```

Output:



III) **JUNIT5**

a. create class Stringmerge and method to implement combining two strings and return value.

b.create class for employees with method to search for name of employee in the given list of employees if exists or contains in the arraylist created.

If name found then display as FOUND else NOT FOUND and return the result.

Create JUNIT Test cases for above both scenarios, using Junit annotations @display,@disabled,@Test,@Beforeall,@afterall and assertion assertequals().

Solution:

```
package newww;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

class Stringmerge{
	String getValue(String firstname,String lastname) {
		return firstname+lastname;
	}
}
class employee{
	List<String>names= new
ArrayList<>(Arrays.asList("Srihari","praveen","rajasekhar"));
```

```
	String emp( String name) {
		if(names.contains(name)) {
			return "Found";
		}
		else {
			return "Not Found";
		}
	}
}
public class Question3 {



}
```

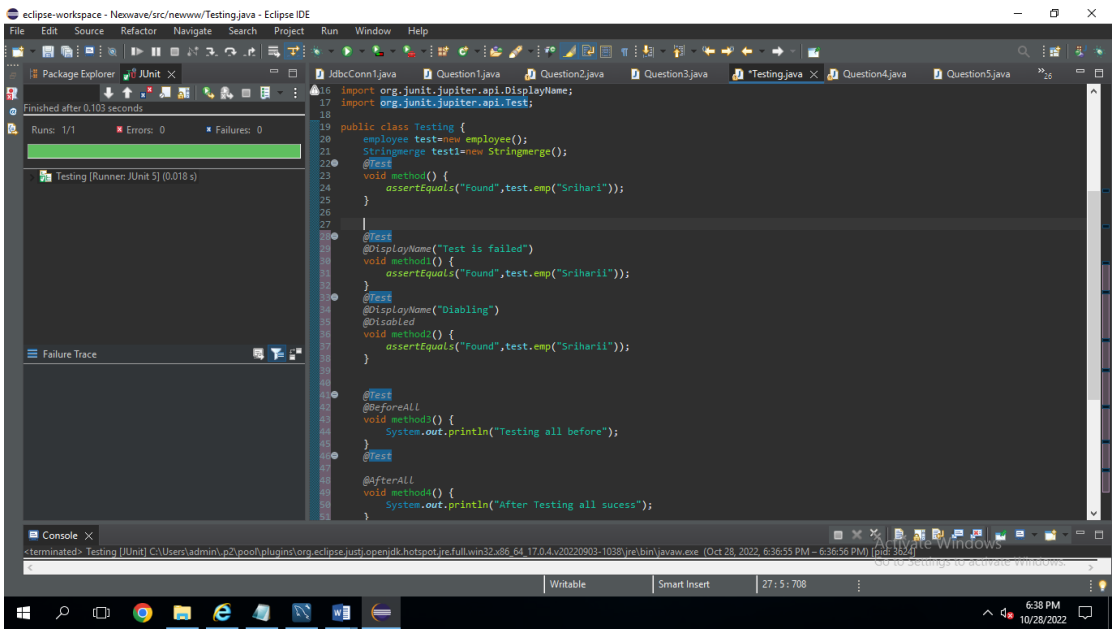
Testing method solution:

```
package newww;


import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Disabled;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;

public class Testing {
	employee test=new employee();
	Stringmerge test1=new Stringmerge();
	@Test
	void method() {
		assertEquals("Found",test.emp("Srihari"));
```

```
}


@Test
@DisplayName("Test is failed")
void method1() {
	assertEquals("Found",test.emp("Sriharii"));
}
@Test
@DisplayName("Diabling")
@Disabled
void method2() {
	assertEquals("Found",test.emp("Sriharii"));
}


@Test
@BeforeAll
void method3() {
	System.out.println("Testing all before");
}
@Test

@AfterAll
void method4() {
	System.out.println("After Testing all sucess");
}
@Test

@AfterEach
void method5() {
	System.out.println("After Each tested Sucess");
}
@Test
@BeforeEach
```

```
        void method6() {
                System.out.println("Before Each tested Sucess");
        }
        @Test

        void method7() {
                assertEquals("Sriharivishnu
janakiraman",test1.getValue("Sriharivishnu", "janakiraman"));
        }
}
```

Output:



IV) USING Collection framework , derive solution for the below problem statement:

create Seat and Theatre classes with below fields information.

Seat --> SeatNo, Price and Reserved

Theatre--> name and ArrayList of seats
--> constructor (name,noOfRows,SeatsperRow)

Business requirement to implement:

Build solution to fill the Theatre with seats of 10 rows and 15 seats per row

The seats name must be A01, A02, A03, ..., H15 as 'A' is the row and '01' as the seat number

The front seats as A01.....A05 are more expensive than the others seats.

1. **Create method that makes you reserve a seat**
2. **Create a method that makes you cancel seat reservation**

Solution:

```
package newww;

import java.util.ArrayList;

class seat{
	int seatno;
	double price;
	boolean reserved;
	public int getSeatno() {
		return seatno;
```

```
	}
	public void setSeatno(int seatno) {
		this.seatno = seatno;
	}
	public double getPrice() {
		return price;
	}
	public void setPrice(double price) {
		this.price = price;
	}
	public boolean isReserved() {
		return reserved;
	}
	public void setReserved(boolean reserved) {
		this.reserved = reserved;
	}


}
class theater{
	String name;
	int row;
	int seatPerRow;
	seat web;
	ArrayList<String>seats;

	theater(String name,int row,int seatPerRow){
		this.name=name;
		this.row=row;
		this.seatPerRow=seatPerRow;
		seats=new ArrayList<>();
		for(int i=1;i<row;i++) {
			for(int j=1;j<seatPerRow;j++) {
				web=new seat();
```

```
                        char[]seatname=
{'A','B','C','D','D','E','F','G','H','I'};
                        String names=seatname[i]+" "+i;
                        web.setSeatno(5);
                        web.setPrice(200);
                        web.setReserved(false);


                }
            }
        }

        public void book(String seatno) {
            if(web.isReserved()) {
                System.out.println(seatno+"  Already booked");
            }else{
                web.setReserved(true);
                System.out.println("Booked Seat sucessfully
"+seatno);
            }

        }


        public void cancel(String seatno) {
            if(web.isReserved()) {
                System.out.println(seatno+"  Sucessfully
cancelled");

            }else{
                web.setReserved(false);
                System.out.println(" Already Cancelled
"+seatno);
            }
```

```
	}
}

public class Question4 {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		theater newthe =new theater("srihari",2,2);

		newthe.book("B5");
		newthe.book("B5");

		newthe.cancel("B5");
	}

}
```

Output: