1) Create a functional interface and use - lambda, static method, instance method and contructor references to instantiate it.
Ans.

```
@FunctionalInterface
public interface Addition<T> {
    public T apply(T x, T y);
    static void print(){
        System.out.println("Static method");
    }
    default void print2(){
        System.out.println("Default Method");
    }
}
public class Auadvance1 {
    public static void main(String[] args) {
        Addition<Integer> a = (x, y) -> x+y; // lambda method
        System.out.println(a.apply(1,2));
        Addition.print(); // static method
        a.print2(); // default method
    }
}
```

```
3
Static method
Default Method
```

2) Demonstrate that local variable used in lamda expression must be final or effectively final and instance variable can be changed inside the lambda expression.

Ans.

It can be seen in code below if the variable s is not changed then code throws no error

```
public class Auadvance1 {
    public static void main(String[] args) {
       int s=1;
        Addition<Integer> a = (x, y) -> {
        return x+s;
        }; // lambda method

        System.out.println(a.apply(1,2));
        Addition.print(); // static method
        a.print2(); // default method
    }
}
```

But if we change s then program throws an error

```
public class Auadvance1 {
    public static void main(String[] args) {
       int s=1;
        Addition<Integer> a = (x, y) -> {
        return x+s;   // s should be final or effectively final
        }; // lambda method
        s=s+1;
        System.out.println(a.apply(1,2));
        Addition.print(); // static method
        a.print2(); // default method
    }
}
```

As it could be seen from below code the instance variable can be changed from lambda expression

```
public class Auadvance1 {
    public static void main(String[] args) {
       int s=1;
        Addition<Integer> a = (x, y) -> {
            x=x+1;
        return x+s;
        }; // lambda method
        System.out.println(a.apply(1,2));
        Addition.print(); // static method
        a.print2(); // default method
    }
}
```

3) Create a Product class having properties like name,category, price .Create a ProductFactory class which gives you the List of products when you pass the number of Products.

Now use stream Api -

1) To get the list of products whose price range is between x and y.(You can assume x and y yourself)

2) To get the total categories in the product list.

3) To get the maximum and minimum priced product in each category.

Ans.

```
public class Product {
    String name;
    String category;
    int price;
    Product(String name,int price,String category){
        this.name=name;
        this.price=price;
        this.category=category;
    }
}


import java.util.*;
public class ProductFactory {
    public List<Product> returnList(Product p[]){
     return Arrays.asList(p);
    }
}


import java.util.Collections;
import java.util.List;
import java.util.stream.Collectors;

public class teststreamapi {

    public static void main(String[] args) {
        Product p1= new Product("soap",100,"bath");
        Product p2= new Product("shampoo",200,"bath");
        Product p3= new Product("rice",50,"eatery");
        Product p4= new Product("dal",70,"eatery");
        Product p5= new Product("cheetos",20,"snacks");
        Product p6= new Product("chips",20,"snacks");
        Product p7= new Product("toffee",2,"snacks");
        Product p8= new Product("bread",40,"bakery");
        Product p9= new Product("bun",50,"bakery");
        Product p10= new Product("cake",100,"bakery");
        Product p[]={p1,p2,p3,p4,p5,p6,p7,p8,p9,p10};
        ProductFactory pf= new ProductFactory();
        List<Product> list= pf.returnList(p);
        List<Product> q1= list.stream().filter(s -> s.price>20 &&
s.price<500).collect(Collectors.toList());
        List<String> q2= list.stream().map(s-
>s.category).distinct().collect(Collectors.toList());
        Integer q3a=list.stream().map(s->s.price).max(Integer::compareTo).get();
        Integer q3b=list.stream().map(s->s.price).min(Integer::compareTo).get();
        System.out.println("Q1");
                for (Product pn : q1){
            System.out.println(pn.name+" "+pn.category+" "+pn.price );
        }
        System.out.println("Q2");
        for(String s: q2){
            System.out.println(s);
```

```
        }
        System.out.println("Q3");
        System.out.println(q3a);
        System.out.println(q3b);
    }
}
```

4) Find the replacement of continue keyword when you are iterating over a collection using forEach() method.

eg-List<integer> ints=new ArrayList<>();

ints.forEach(x->System.out.println(x))

Now using this method if we want to skip some object contditionally. Then How are we gonna do. (In for loop we have continue keyword but here how we'll do)

Please write a program for the same.

Ans.

```
import java.util.ArrayList;

public class continuereplace {
    public static void main(String[] args) {
        ArrayList<Integer> a= new ArrayList<>();
        a.add(1);
        a.add(1);
        a.add(1);
        a.add(0);
        a.add(1);
        a.add(1);
        a.add(0);
        a.add(1);
        a.forEach(x->{
            if(x==0){

            }
            else
                System.out.println(x);
        });
    }
```

5) Establish a jdbc connection to a database and print first 10 rows of the table.

Ans.

```
import java.sql.*;
public class JDBCExample {
   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/hospital";

   static final String USER = "arpit";
   static final String PASS = "123456";

   public static void main(String[] args) {
   Connection conn = null;
   Statement stmt = null;

   try{
      Class.forName("com.mysql.jdbc.Driver");
      System.out.println("Connecting to database...");
      conn = DriverManager.getConnection(DB_URL,USER,PASS);
      System.out.println("Creating statement...");
      stmt = conn.createStatement();
      String sql;
      sql = "select * from patient limit 10";
      ResultSet rs = stmt.executeQuery(sql);
      while(rs.next()){
         int id  = rs.getInt("id");
         int age = rs.getInt("age");
         String name = rs.getString("name");
         String sex = rs.getString("sex");

         System.out.print("ID: " + id);
         System.out.print(", Name: " + name);
         System.out.print(", Age: " + age);
         System.out.println(", Sex: " + sex);
      }
      System.out.println("Closing");
      rs.close();
      stmt.close();
      conn.close();
   }catch(SQLException se){
      se.printStackTrace();
   }catch(Exception e){
      e.printStackTrace();
   }finally{
      try{
         if(stmt!=null)
            stmt.close();
      }catch(SQLException se2){
      }// nothing we can do
      try{
         if(conn!=null)
            conn.close();
      }catch(SQLException se){
         se.printStackTrace();
      }//end finally try
   }//end try
}
}
```

6) Create a dummy xml and parse and print its data using java program.

Ans.

Xml file

```
<breakfast_menu>
<food>
<name>Belgian Waffles</name>
<price>$5.95</price>
<description>Two of our famous Belgian Waffles with plenty of real maple syrup</
description>
<calories>650</calories>
</food>
<food>
<name>Strawberry Belgian Waffles</name>
<price>$7.95</price>
<description>Light Belgian waffles covered with strawberries and whipped cream</
description>
<calories>900</calories>
</food>
<food>
<name>Berry-Berry Belgian Waffles</name>
<price>$8.95</price>
<description>Light Belgian waffles covered with an assortment of fresh berries
and whipped cream</description>
<calories>900</calories>
</food>
<food>
<name>French Toast</name>
<price>$4.50</price>
<description>Thick slices made from our homemade sourdough bread</description>
<calories>600</calories>
</food>
<food>
<name>Homestyle Breakfast</name>
<price>$6.95</price>
<description>Two eggs, bacon or sausage, toast, and our ever-popular hash
browns</description>
<calories>950</calories>
</food>
</breakfast_menu>
```

Program

```
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import javax.xml.parsers.*;
import java.io.*;
public class xmltest {
    public static void main(String[] args) {
      try {
          File inputFile= new File("/home/arpit/Desktop/test.xml");
          DocumentBuilderFactory dbFactory=
DocumentBuilderFactory.newInstance();
          DocumentBuilder dBuilder = dbFactory.newDocumentBuilder();
          Document doc= dBuilder.parse(inputFile);
          doc.getDocumentElement().normalize();
          System.out.println(doc.getDocumentElement().getNodeName());
          NodeList nList= doc.getElementsByTagName("food");
          for(int temp=0; temp<nList.getLength();temp++){
              Node nNode= nList.item(temp);
              System.out.println(nNode.getNodeName());
              if(nNode.getNodeType()==Node.ELEMENT_NODE){
                  Element eElement = (Element) nNode;
                  System.out.println("name" +
eElement.getElementsByTagName("name").item(0).getTextContent());
```

```
                    System.out.println("name" +
eElement.getElementsByTagName("price").item(0).getTextContent());
                    System.out.println("name" +
eElement.getElementsByTagName("description").item(0).getTextContent());
                    System.out.println("name" +
eElement.getElementsByTagName("calories").item(0).getTextContent());

                }
            }
        }
        catch (Exception e){
            e.printStackTrace();
        }
    }
}
```

7) WAP to list all files from a directory recursively with Java.

Ans.

```
import java.io.*;
public class fileio1{
public static void main(String[]args){


         File dir=new File("/home/arpit/Desktop");
         String[]contents=dir.list();
         for(int i=0;i<contents.length;i++){
         System.out.println(contents[i]);
         }
         }
}
```

8) WAP to list out all files that are end with ".txt" extension in a folder, and then delete it.

Ans.

```
import java.io.*;
public class fileio1{
public static void main(String[]args){


         File dir=new File("/home/arpit/Desktop/abx");
         String[]contents=dir.list(new FilenameFilter() {
             @Override
             public boolean accept(File file, String s) {
                 return s.endsWith(".txt");
             }
         });
         for(int i=0;i<contents.length;i++){
             File f= new File("/home/arpit/Dekstop/abx/"+contents[i]);
             f.delete();
         System.out.println(contents[i]+" deleted");
         }
         }
}
```

9) WAP to copy one file into another

Ans.

```
import java.io.*;
public class fileio1{
public static void main(String[]args){
```

```
FileInputStream instream =null;
FileOutputStream outstream=null;
try{
    File infile= new File("/home/arpit/Desktop/test.xml");
    File outfile= new File("/home/arpit/Desktop/output.xml");
instream= new FileInputStream(infile);
outstream= new FileOutputStream(outfile);
byte[] buffer = new byte [1024];
int length;
while((length=instream.read(buffer))>0){
    outstream.write(buffer,0,length);
}
instream.close();
outstream.close();
    System.out.println("file copied");
}
catch (Exception e){
    e.printStackTrace();
}
        }
}
```

10)WAP to copy a file from one dir to another dir..

Ans.

```
import java.io.*;
import java.nio.file.Files;
import java.nio.file.StandardCopyOption;

public class fileio1{
public static void main(String[]args) throws IOException {
File f= new File("/home/arpit/Desktop/papers/kapoor2011.pdf");
    Files.copy(f.toPath(),new
File("/home/arpit/Desktop/abcd/kapoor.pdf").toPath(),
StandardCopyOption.REPLACE_EXISTING);
        }
}
```