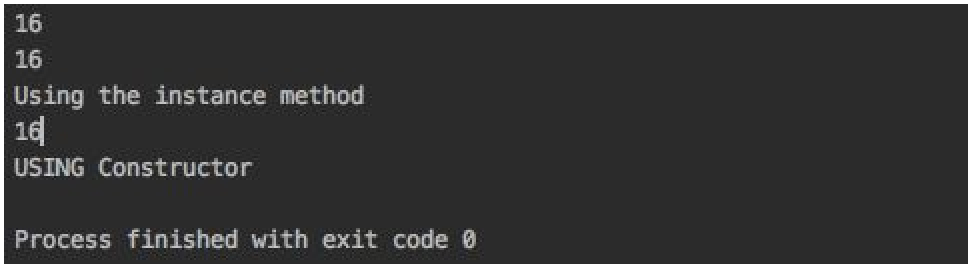
1) Create a functional interface and use - lambda, static method, instance method and contructor references to instantiate it.

CODE:

```
import java.io.*;

class q1
{
  public static void main(String[] args) throws IOException
  {
     int b=4;

     // lambda expression
     Square s = (int x)->x*x;
     int ans = s.calculate(b);
     System.out.println(ans);

     // static method
     Square sq = q1 :: getSquare;
     System.out.println(sq.calculate(b));

     // instance method
     q1 obj=new q1("Using the instance method");
     Square sq1= obj::square;
     System.out.println(sq1.calculate(b));

     // Constructor
     Printer p=q1::new;
     p.display("USING Constructor");

  }
  public static int getSquare(int s)
  {
     return s*s;
  }
  public int square(int m)
  {
     return m*m;
  }
  public q1(String str)
  {
     System.out.println(str);
  }
}

@FunctionalInterface
interface Square{
  int calculate(int x);
}
```

```
@FunctionalInterface
interface Printer{
   void display(String s);
}
```

OUTPUT:



```
16
16
Using the instance method
16
USING Constructor

Process finished with exit code 0
```

Q3. 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.

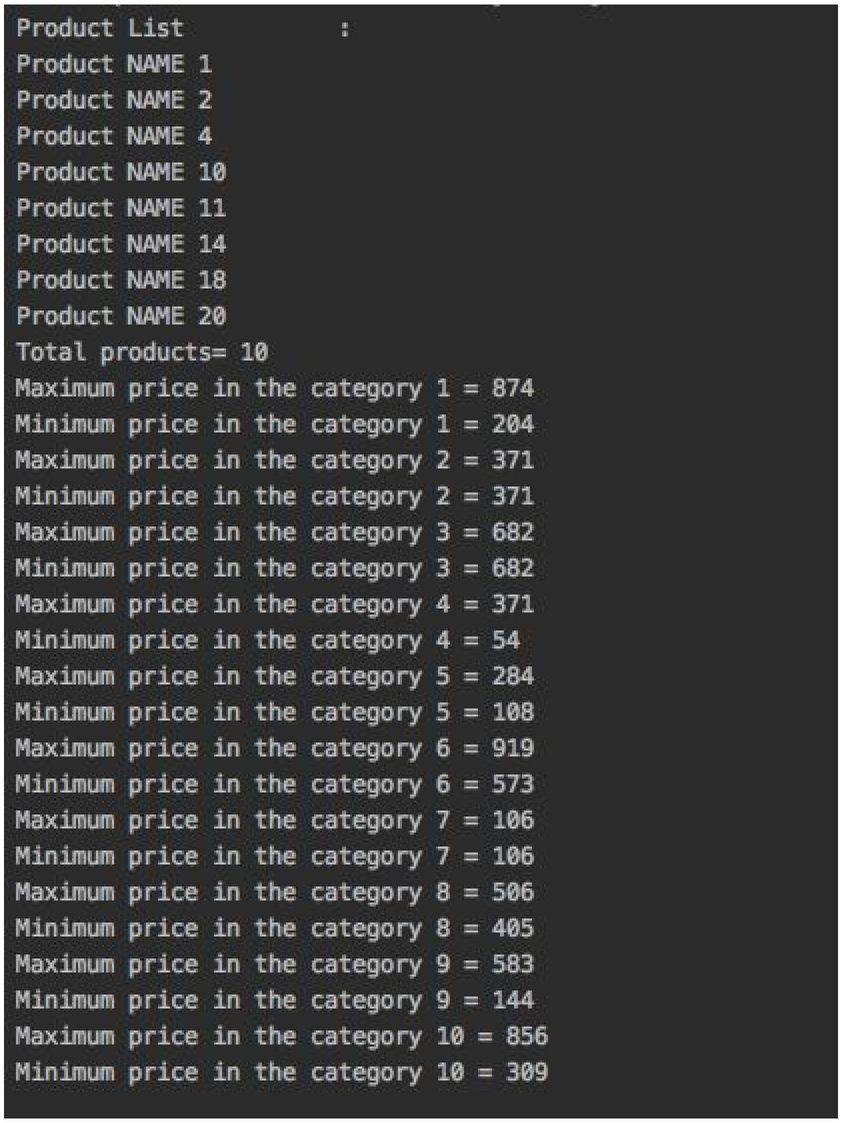
.

CODE:

```
import java.io.*;
import java.util.*;
import java.util.stream.Collectors;
class q3
{
   public static void main(String[] args) throws IOException
   {
       ProductFactory productFactory=new ProductFactory();
       List<Product> products=productFactory.getListOfProducts(25);
       int x=500;
       int y=900;

       //(i)
       List<Product> productsFiltered =products.stream().filter(product->product.getPrice()>=x &&
product.getPrice()<=y).collect(Collectors.toList());
       System.out.println("Product List          :");
       productsFiltered.forEach(product->System.out.println(product.getName()));

       //(ii)
```

```
        Set<Integer> categories=products.stream().map(Product::getCategory).collect(Collectors.toSet());
        int totalCategories=categories.size();
        System.out.println("Total products= "+totalCategories);

        //(iii)
        for(Integer i: categories)
        {
            int max,min;

max=products.stream().filter(product->product.getCategory()==i).map(Product::getPrice).max(Comparator.comparing
(Integer::valueOf)).get();

min=products.stream().filter(product->product.getCategory()==i).map(Product::getPrice).min(Comparator.comparing(I
nteger::valueOf)).get();
            System.out.println("Maximum price in the category "+i+" = "+max);
            System.out.println("Minimum price in the category "+i+" = "+min);

        }
    }

}
class ProductFactory {

    public List<Product> getListOfProducts(int numOfProducts) {
        List<Product> products = new ArrayList<>();
        for (int i = 1; i <= numOfProducts; i++) {
            products.add(createProduct(i));
        }
        return products;
    }

    private Product createProduct(int i) {
        Product product = new Product();
        product.setName("Product NAME " + i);
        product.setCategory((int) (Math.random() * (10-1+1) + 1));
        product.setPrice((int) (Math.random() * (1000-100+1) + 20));
        return product;

    }


}


class Product {
    private String name;
    private Integer category;
    private Integer price;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getCategory() {
        return category;
    }

    public void setCategory(Integer category) {
        this.category=category;
    }
    public Integer getPrice() {
        return price;
    }

    public void setPrice(Integer price) {
        this.price=price;
    }
}
```

OUTPUT:



```
Product List           :
Product NAME 1
Product NAME 2
Product NAME 4
Product NAME 10
Product NAME 11
Product NAME 14
Product NAME 18
Product NAME 20
Total products= 10
Maximum price in the category 1 = 874
Minimum price in the category 1 = 204
Maximum price in the category 2 = 371
Minimum price in the category 2 = 371
Maximum price in the category 3 = 682
Minimum price in the category 3 = 682
Maximum price in the category 4 = 371
Minimum price in the category 4 = 54
Maximum price in the category 5 = 284
Minimum price in the category 5 = 108
Maximum price in the category 6 = 919
Minimum price in the category 6 = 573
Maximum price in the category 7 = 106
Minimum price in the category 7 = 106
Maximum price in the category 8 = 506
Minimum price in the category 8 = 405
Maximum price in the category 9 = 583
Minimum price in the category 9 = 144
Maximum price in the category 10 = 856
Minimum price in the category 10 = 309
```

Q4. 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.

CODE:

```
import java.sql.*;
class q5
{

   static final String JDBC_DRIVER = "com.mysql.jdbc.Driver";
   static final String DB_URL = "jdbc:mysql://localhost/lbs";

   static final String USER = "root";
   static final String PASS = "HEllOnitin";
   public static void main(String[] args)
   {
       Connection conn = null;
       Statement stmt = null;
       try{

           Class.forName("com.mysql.jdbc.Driver");


           System.out.println("Connecting to a selected database...");
           conn = DriverManager.getConnection(DB_URL, USER, PASS);
           System.out.println("Connected database successfully...");


           System.out.println("Creating statement...");
           stmt = conn.createStatement();

           String sql = "SELECT TOP * FROM DOCTOR";
           ResultSet rs = stmt.executeQuery(sql);
           while(rs.next()){

               int id  = rs.getInt("id");
               int age = rs.getInt("age");


               System.out.print("ID: " + id);
               System.out.print(", Age: " + age);
           }
           rs.close();
       }
       catch(SQLException se){

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

           e.printStackTrace();
       }finally{

           try{
               if(stmt!=null)
```

```
                conn.close();
         }catch(SQLException se){
         }
         try{
            if(conn!=null)
               conn.close();
         }catch(SQLException se){
            se.printStackTrace();
         }
      }
   }
}
```

Q6. Create a dummy xml and parse and print its data using java program.

CODE:

```
import java.io.*;
import java.util.*;
import org.w3c.dom.*;

import org.xml.sax.SAXException;

import javax.xml.parsers.*;
class q6
{
  public static void main(String[] args) throws ParserConfigurationException, SAXException,IOException
  {

      DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
      DocumentBuilder builder = factory.newDocumentBuilder();


      Document document = builder.parse(new File("employee.xml"));


      document.getDocumentElement().normalize();


      Element root = document.getDocumentElement();
      System.out.println(root.getNodeName());

      NodeList nList = document.getElementsByTagName("employee");
      System.out.println("------------------");
```

```
        for (int temp = 0; temp < nList.getLength(); temp++)
        {
            Node node = nList.item(temp);
            System.out.println("");
            if (node.getNodeType() == Node.ELEMENT_NODE)
            {

                Element eElement = (Element) node;
                System.out.println("Employee id : "    + eElement.getAttribute("id"));
                System.out.println("First Name : "  +
eElement.getElementsByTagName("firstName").item(0).getTextContent());
                System.out.println("Last Name : "   +
eElement.getElementsByTagName("lastName").item(0).getTextContent());
                System.out.println("Location : "    +
eElement.getElementsByTagName("location").item(0).getTextContent());
            }
        }
    }
}
```

Q7. WAP to list all files from a directory recursively with Java.

Code:

```
import java.io.File;
import java.io.IOException;
public class q7 {
   public static void listOfFiles(File dirPath){
      File filesList[] = dirPath.listFiles();
      for(File file : filesList) {
         if(file.isFile()) {
            System.out.println("File path: "+file.getName());
         } else {
            listOfFiles(file);
         }
      }
   }
   public static void main(String args[]) throws IOException {
      //Creating a File object
      File file = new File("desktop\\tempdirectory");
      //List of all files & directories
      listOfFiles(file);
   }
}
```

Q8. WAP to list out all files that are end with ".txt" extension in a folder, and then delete it.

CODE:

```java
import java.io.*;

class q8
{
  public static void main(String[] args) throws IOException
  {
     String folderName="desktop\\nitinmishra";
     File folder =new File(folderName);
     File[] txtFiles=folder.listFiles(new FilenameFilter() {
        public boolean accept(File dir, String filename)
        { return filename.endsWith(".txt"); }});
     for(File file:txtFiles)
     {
        if(file.delete())
           System.out.println("Deleted file    : " + file.getName());
        else
           System.out.println("Failed to delete the file.");

     }
  }
}
```

Q9. WAP to copy one file into another

CODE:

```java
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;

public class q9{
  public static void main(String[] args) {
     FileInputStream ins = null;
     FileOutputStream outs = null;
     try {
        File infile = new File("desktop\\nitin.txt");
        File outfile = new File("desktop\\mishra.txt");
        ins = new FileInputStream(infile);
        outs = new FileOutputStream(outfile);
        byte[] buffer = new byte[1024];
        int length;

        while ((length = ins.read(buffer)) > 0) {
           outs.write(buffer, 0, length);
        }
        ins.close();
        outs.close();
```

```
            System.out.println("File copied successfully!!");
        } catch(IOException ioe) {
            ioe.printStackTrace();
        }
    }
}
```

Q10. WAP to copy a file from one dir to another dir..

```
import java.io.*;



class q10
{
   public static void main(String[] args) throws IOException
   {
       File source = new File("desktop\\temp\\nitin.txt");
       File target = new File("desktop\\nitin.txt");
       InputStream is = null;
       OutputStream os = null;
       try {
           is = new FileInputStream(source);
           os = new FileOutputStream(target);

           byte[] buf = new byte[1024];

           int bytesRead;
           while ((bytesRead = is.read(buf)) > 0) {
               os.write(buf, 0, bytesRead);
           }
       }
       catch(IOException ioe){
           ioe.printStackTrace();
       }
       finally {
           is.close();
           os.close();
       }
   }
}
```