**Advance Java Assignment**

Q1. (i) Using Lambda  
interface Interface

{

void abstractFn(int x);

}

public class Main

{

public static void main(String args[])

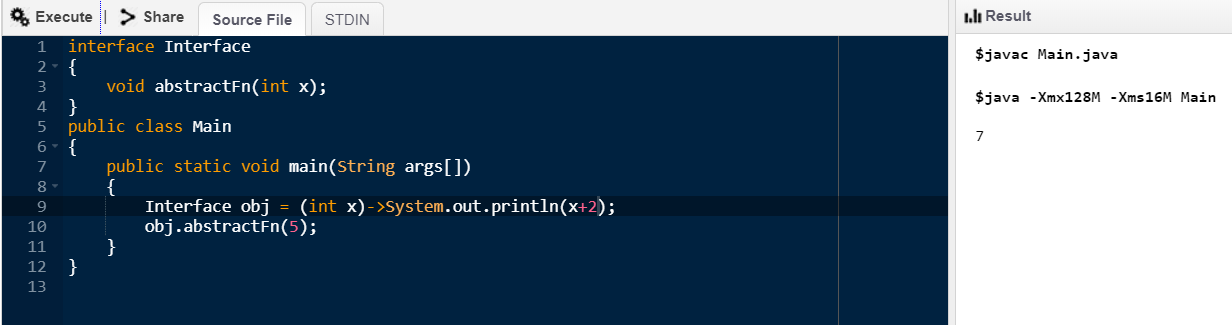
{

Interface obj = (int x)->System.out.println(x+2);

obj.abstractFn(5);

}

}



(ii) Using static method  
interface Interface {

static void first()

{

System.out.println("Static Method");

}

void overrideMethod(String str);

}

public class InterfaceDemo implements Interface {

public static void main(String[] args)

{

InterfaceDemo interfaceDemo = new InterfaceDemo();

Interface.first();

interfaceDemo.overrideMethod("Override Method");

}

@Override

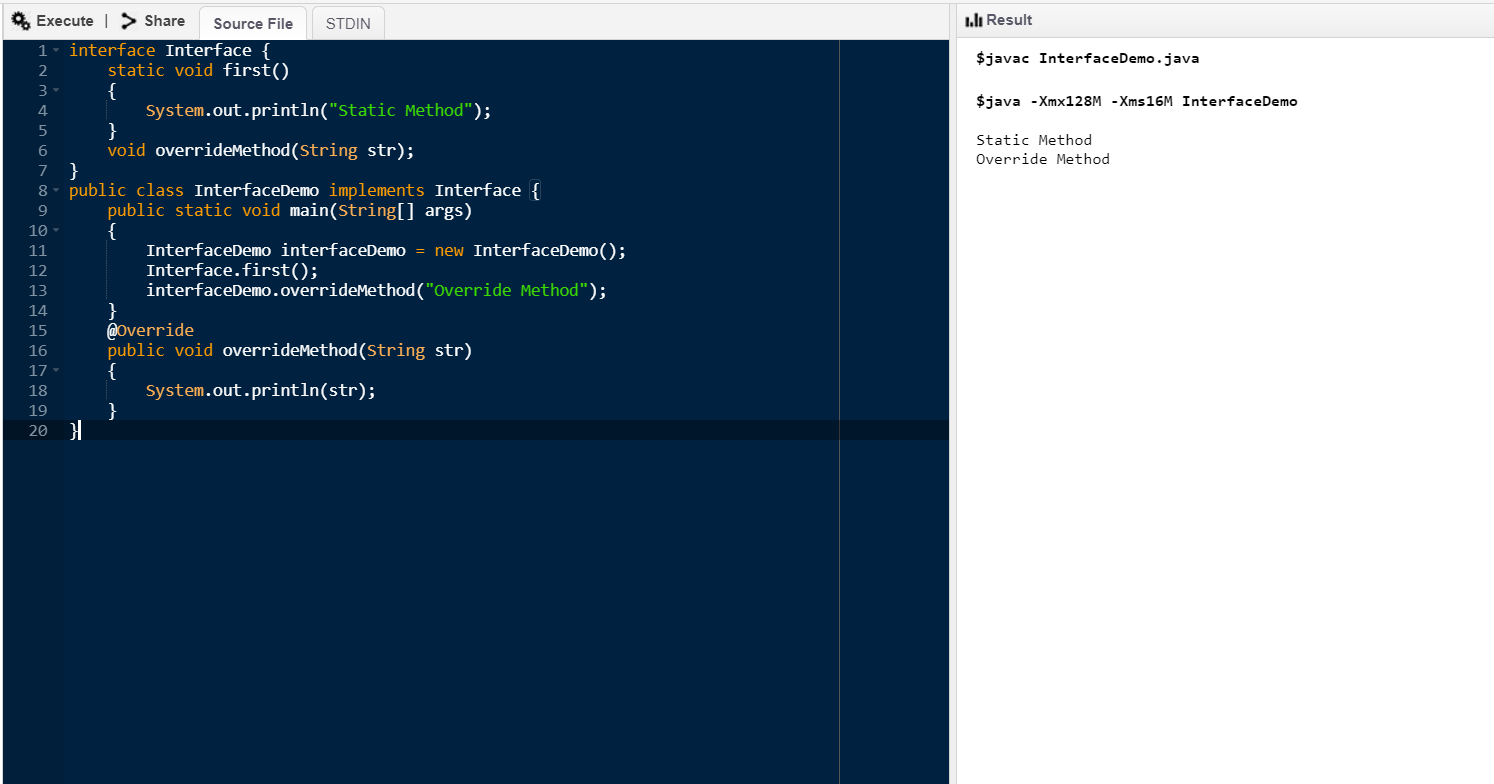
public void overrideMethod(String str)

{

System.out.println(str);

}

}



Q2.

interface Interface

{

void abstractFn(int x);

}

public class Main

{

public static void main(String args[])

{

Interface obj = (int x)->System.out.println(x+2);

obj.abstractFn(5);

}

}

Q3.

class Product {

private Integer Category;

private Double price;

public Product(Integer category, Double price) {

super();

Category = category;

this.price = price;

}

public Integer getCategory() {

return Category;

}

public void setCategory(Integer category) {

Category = category;

}

public Double getPrice() {

return price;

}

public void setPrice(Double price) {

this.price = price;

}

}

class ProductFactory {

public double assignPrice() {

int min = 10, max = 1000;

return Math.random() \* (max - min);

}

public int assignCategory() {

return (int) (Math.random() \* (10) + 1);

}

public List<Product> createList(int size) {

List<Product> productList = new ArrayList<>(size);

for (int i = 0; i < size; i++) {

productList.add(new Product(assignCategory(), assignPrice()));

}

return productList;

}

}

public class Q3 {

public static void main(String[] args) {

System.out.println("Enter size of list");

Scanner s = new Scanner(System.in);

int size = s.nextInt();

ProductFactory obj = new ProductFactory();

List<Product> productList = obj.createList(size);

System.out.println("Enter range of price");

int x = s.nextInt();

int y = s.nextInt();

List<Product> FilteredProduct = productList.stream()

.filter(product -> product.getPrice() >= x && product.getPrice() <= y).collect(Collectors.toList());

System.out.println("Products : ");

for (Product p : FilteredProduct) {

System.out.println("Product Category " + p.getCategory());

System.out.println("Product Name " + p.getPrice());

}

Set<Integer> categoryList = productList.stream().map(Product::getCategory).collect(Collectors.toSet());

int totalCategories = categoryList.size();

System.out.println("Total number of categories = " + totalCategories);

double minPrice = 0;

double maxPrice = 0;

for (Integer i : categoryList) {

maxPrice = productList.stream().filter(product -> product.getCategory() == i).map(Product::getPrice)

.max(Comparator.comparing(Double::valueOf)).get();

minPrice = productList.stream().filter(product -> product.getCategory() == i).map(Product::getPrice)

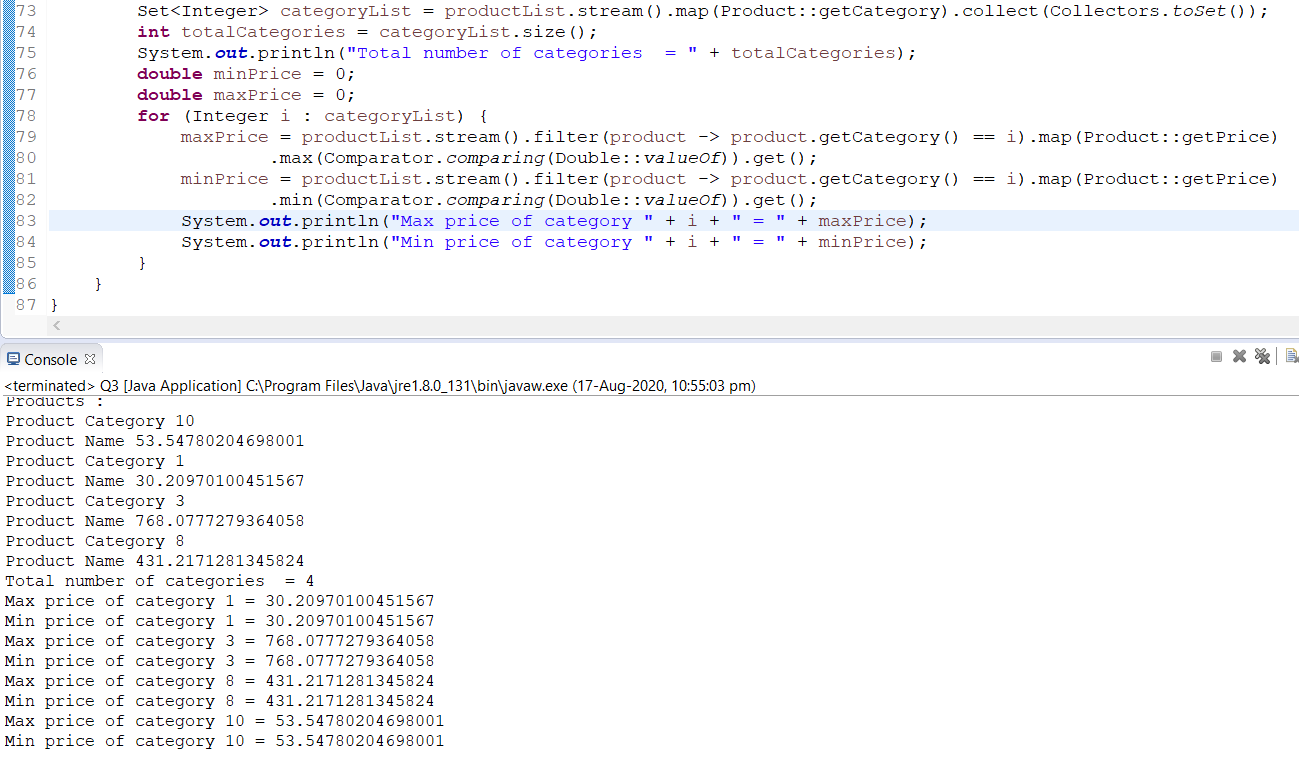
.min(Comparator.comparing(Double::valueOf)).get();

System.out.println("Max price of category " + i + " = " + maxPrice);

System.out.println("Min price of category " + i + " = " + minPrice);

}

}

}

Q4. We can use return statement for conditional rendering instead of continue as shown below.

public class without\_continue

{

public static void main(String[] args) {

List<Integer> num = Arrays.asList (1,2,3,4,5,6);

num.forEach( x-> {

if( x%2 == 0) {

return;

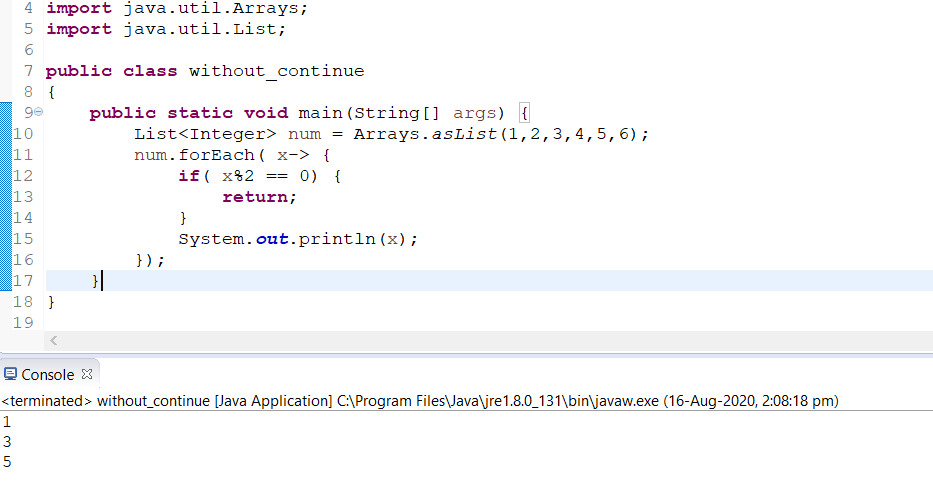
}

System.out.println(x);

});

}

}



Q5.

public class MySqlConnection

{

public static Connection doConnect()

{

Connection con=null;

try {

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

System.out.println("Loaded...");

con=DriverManager.getConnection("jdbc:mysql://localhost/java\_project","root","bce");

System.out.println("Connected");

}

catch (ClassNotFoundException | SQLException e)

{

e.printStackTrace();

}

return con;

}

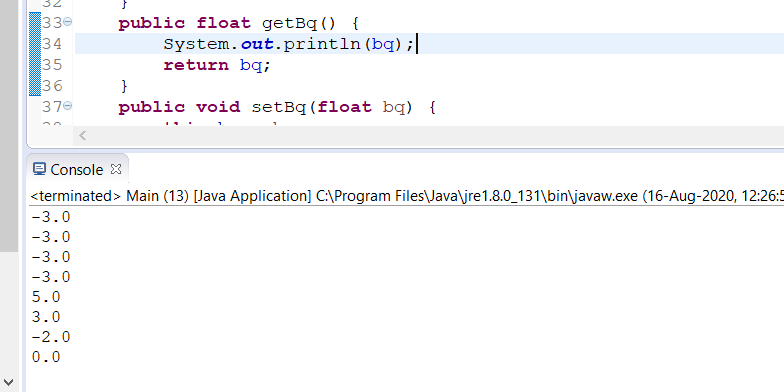
public static void main(String args[])

{

doConnect();

}

}



Q7. public class list\_files

{

static void Print(File[] arr, int level)

{

for (File f : arr)

{

for (int i = 0; i < level; i++)

System.out.print("\t");

if(f.isFile())

System.out.println(f.getName());

else if(f.isDirectory())

{

System.out.println("[" + f.getName() + "]");

Print(f.listFiles(), level + 1);

}

}

}

public static void main(String[] args)

{

String path = "C:\\Users\\Vivek\\Desktop\\Pranvi";

File directory = new File(path);

if(directory.exists() && directory.isDirectory())

{

File arr[] = directory.listFiles();

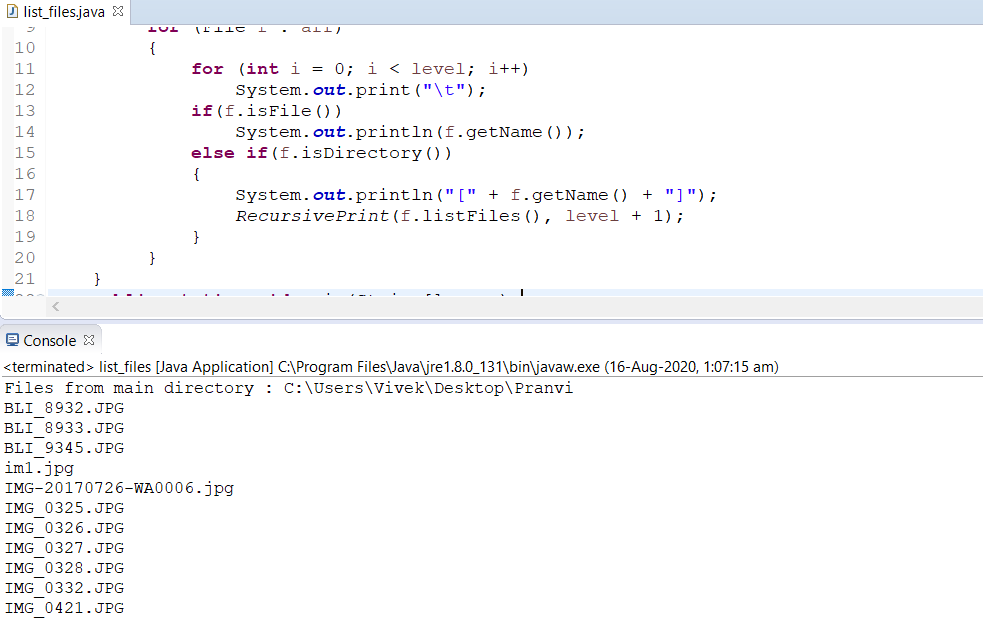
System.out.println("Files from main directory : " + directory);

RecursivePrint(arr, 0);

}

}

}



Q8.

public class txt\_files {

public static void main(String[] args) {

File directoryPath = new File("F:\\Linux files\\Documents\\docs\\Final\\Accolite\\Sessions\\Java");

File[] files=directoryPath.listFiles(new FilenameFilter() {

public boolean accept(File dir, String name) {

return name.endsWith(".txt");

}

});

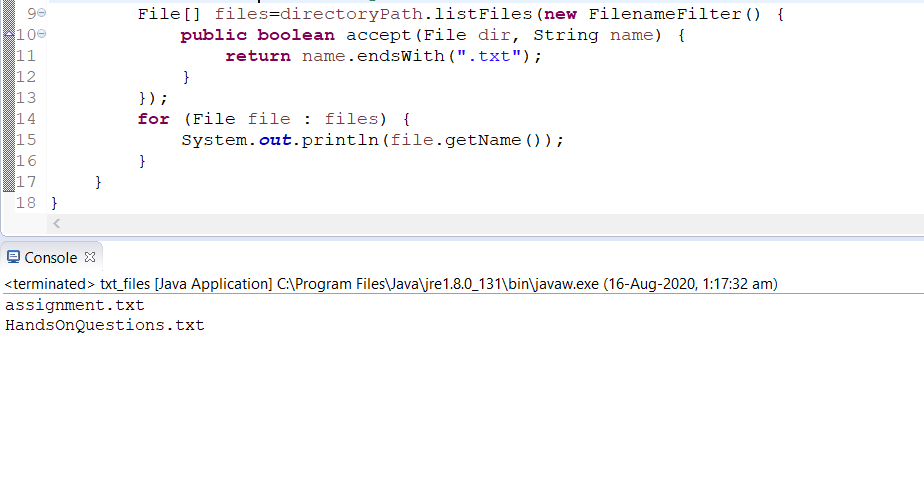
for (File file : files) {

System.out.println(file.getName());

}

}

}



Q9. public class file\_copy {

public static void main(String[] args) {

try {

FileReader fread = new FileReader("E:\\Real Java\\programs\\Exceptions\\Accolite\\src\\adv\_java\_assignment\\input.txt");

BufferedReader bread = new BufferedReader(fread);

FileWriter fwrite = new FileWriter("E:\\Real Java\\programs\\Exceptions\\Accolite\\src\\adv\_java\_assignment\\output.txt", true);

String s;

while ((s = bread.readLine()) != null) {

fwrite.write(s);

fwrite.flush();

}

bread.close();

fwrite.close();

System.out.println("file copied");

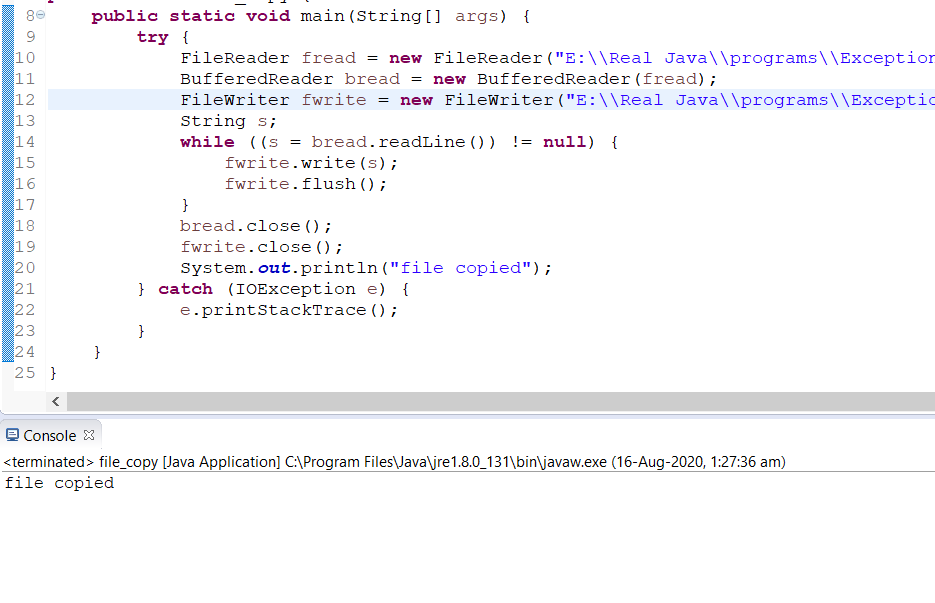
} catch (IOException e) {

e.printStackTrace();

}

}

}



Q10.

public class file\_transfer

{

public static void main(String[] args) throws IOException

{

Path temp = Files.move

(Paths.get("E:\\Real Java\\programs\\Exceptions\\Accolite\\src\\adv\_java\_assignment\\input.txt"),

Paths.get("E:\\Real Java\\programs\\Exceptions\\String\_class\\src\\inp.txt"));

if(temp != null)

{

System.out.println("File moved successfully");

}

else

{

System.out.println("Failed");

}

}

}

